الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Nº Réf : ……

Centre Universitaire
Abd Elhafid Boussouf Mila

Institut des Sciences et Technologie
Département de Mathématiques et Informatique

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique
Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Proposing and implementing a solution to protect Internet of Things environments

Préparé par : Berche Imane
Moulakmim Chaima

Soutenue devant le jury

Encadré par	Lalouci Ali	Grade MAA
Président	Guetiche Mourad	Grade MCB
Examinateur	Bessouf Hakim	Grade MAA

Année Universitaire : 2021/2022

Remerciement

En préambule à ce PFE nous remerciant **ALLAH** *qui nous a aidé et nous a donné la patience et le courage durant ces longues années d'étude.*

Nous souhaitant adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce PFE ainsi qu'à la réussite de cette formidable année universitaire.

Ces remerciements vont tout d'abord au corps professoral et administratif du département de MI pour la richesse et la qualité de leurs enseignements et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Nous tenons à remercier sincèrement Monsieur **A. Lalouci** *qui en tant qu'encadrant de PFE, elle a toujours montré l'écoute et la disponibilité tout au long de la réalisation de ce PFE, ainsi pour l'inspiration, l'aide et le temps qu'elle a bien voulu nous consacrer pour que ce mémoire voit le jour*

Chaima et Imane

Dédicaces

A mes très chers parents, source de vie, d'amour et d'affection

A mes chers frères, source de joie et de bonheur

A toute ma famille, source d'espoir et de motivation

A tous mes amis, tout particulièrement Rayane,Ryane, Houssna, Nihad, Chaima,Houda, Assia et Soufia

A Imen, chère amie et sœur avant d'être binôme

A vous cher lecteur

CHAIMA

Dédicaces

Je dédie cet ouvrage

A ma très chère mère

A la femme qui a souffert sans me laisser souffrir, qui n'a jamais dit non à mes exigences et qui n'a épargné aucun effort pour me rendre heureuse

A mon très cher père

Tu as toujours été à mes côtés pour me soutenir et m'encourager. Que ce travail traduit ma gratitude et mon affection.

A mes chères sœurs, et mon frère, Que Dieu les protège et leurs offre la chance et le bonheur.

A toute ma famille Berche et Bouakouche, Que Dieu leur donne une longue et joyeuse vie.

A tous mes amis, tout particulièrement Ryane, Chahinez, Houssna, Chaima,Nihad, Houda, Sofia,

A Chaïma & Khouloud, chère amies et sœurs savant d'être binôme

Imene

Abstract:

Internet of Things (IoT) technology has attracted a lot of attention in recent years. It connects things to the internet, enabling insights that have never been available before. IoT is a large network of devices that includes various smart devices and sensors. These objects collect and exchange data. Their physical constraints as well as the hostile environments in which they could be deployed make the IoT vulnerable and require effective and inexpensive security mechanisms.

In security, there are several methods used to protect things like a firewall.

A firewall is a network security mechanism that monitors and controls incoming and outgoing network traffic (applications and data) based on predetermined security rules. A firewall generally establishes a barrier between a trusted network and an untrusted network, such as the Internet. However, the installation and maintenance of these require a very high cost.

The objective of this project is to design and implement a distributed algorithm for finding nodes forming a dominating set in IOT environments. Thus, we propose an effective solution allowing us to reduce the cost of installing firewalls in the nodes of the system.

Résumé :

La technologie de l'Internet des objets (IoT) a attiré beaucoup d'attention ces dernières années. Il connecte des objets à Internet, permettant d'obtenir des idées qui n'ont jamais été disponibles auparavant. L'IoT est un vaste réseau d'appareils qui comprend divers appareils et capteurs intelligents. Ces objets collectent et échangent des données. Leurs contraintes physiques ainsi que les environnements hostiles dans lesquels ils pourraient être déployés rendent l'IoT vulnérable et nécessitent des mécanismes de sécurité efficaces et peu coûteux.

En sécurité, il existe plusieurs méthodes utilisées pour protéger les choses comme un pare-feu.

Un pare-feu est un mécanisme de sécurité réseau qui surveille et contrôle le trafic réseau entrant et sortant (applications et données) en fonction de règles de sécurité prédéterminées. Un pare-feu établit généralement une barrière entre un réseau de confiance et un réseau non fiable, tel qu'Internet. Cependant, l'installation et la maintenance de ceux-ci nécessitent un coût très élevé.

L'objectif de ce projet est de concevoir et d'implémenter un algorithme distribué pour la recherche des nœuds formant un ensemble dominant dans les environnements IOT. Ainsi, nous proposons une solution efficace nous permettant de réduire le coût d'installation des pare-feux dans les nœuds du système.

الملخص:

جذبت تقنية إنترنت الأشياء (IoT) الكثير من الاهتمام في السنوات الأخيرة. يربط الأشياء بالإنترنت ، مما يتيح رؤى لم تكن متاحة من قبل. إنترنت الأشياء هو عبارة عن شبكة كبيرة من الأجهزة التي تضم العديد من الأجهزة الذكية وأجهزة الاستشعار. تقوم هذه الكائنات بجمع البيانات وتبادلها. القيود المادية بالإضافة إلى البيئات المعادية التي يمكن نشرها فيها تجعل إنترنت الأشياء ضعيفًا وتتطلب آليات أمنية فعالة وغير مكلفة.

في مجال الأمان ، هناك العديد من الطرق المستخدمة لحماية أشياء مثل جدار الحماية.

جدار الحماية عبارة عن آلية أمان للشبكة تراقب وتتحكم في حركة مرور الشبكة الواردة والصادرة (التطبيقات والبيانات) بناءً على قواعد أمان محددة مسبقًا. يُنشئ جدار الحماية بشكل عام حاجزًا بين شبكة موثوقة وشبكة غير موثوق بها ، مثل الإنترنت. ومع ذلك ، فإن تركيبها وصيانتها يتطلب تكلفة عالية للغاية.

الهدف من هذا المشروع هو تصميم وتنفيذ خوارزمية موزعة لإيجاد العقد التي تشكل مجموعة مسيطرة في بيئات إنترنت الأشياء. وبالتالي ، فإننا نقترح حلاً فعالاً يسمح لنا بتقليل تكلفة تثبيت جدران الحماية في عقد النظام.

List of Figures

List of Tables

Contents

General introduction

The Internet of Things (IoT) refers to a concept of connected objects and devices of all types over the Internet wired or wireless. The popularity of IoT or the Internet of Things has increased rapidly, as these technologies are used for various purposes, including communication, transportation, education, and business development. IoT introduced the hyper-connectivity concept, which means organizations and individuals can communicate with each other from remote locations effortlessly.

Where there is 26.66 billion IoT devices exist in the current world. The mass explosion started in 2011 with the introduction of home automation, wearable devices, and smart energy meters. The rapid explosion of IoT has benefitted organizations and in various ways improved market research and business strategies. Similarly, IoT has improved the lifestyle of individuals by introducing automated services. However, such an uncontrolled explosion has increased privacy and security challenges.

Thus, unintended use, failure to change passwords, and lack of device updates lead to increased risks of cybersecurity and malicious applications that access sensitive data of IoT systems. Inappropriate security practices increase the chances of data breaches and other threats. Most security professionals consider the Internet of Things to be a weak point for cyber attacks due to weak security policies and protocols. Although many electronic security mechanisms have been developed to protect IoT devices from cyber attacks, security guidelines have not been sufficiently documented. Thus, end users cannot use preventive measures to avoid data attacks. Hackers have developed different types of malware to infect IoT devices.

To solve the problem posed, we used graphs, which are powerful tools used to model and solve real-life networks problems in different domains such as transportation, communication and computer science. In such networks, we find direct applications of graphs since a network can be represented as a set of nodes connected between them using physical or logical links. This strength is due principally to two main factors:

- The simplicity a graph provides to model real-life problems while entities are represented by nodes and the interaction between them by edges

- Graphs have many properties, known as graph parameters, very helpful in problem solving. The common approach for problems solving using graphs consists of:

- Modeling the problem by a graph

- Searching the appropriate property (parameter) corresponding to the studied problem

- Finding the parameter value using suitable algorithms .

The objective of this project is to propose an effective solution allowing us to reduce the cost of installing firewalls in the system, to balance the loads between the different firewalls installed, and to ensure the control of data flow in the system. All while respecting certain inherent properties such as mobility and heterogeneity.This thesis is composed of forth chapters :

• The first chapter:Domination Problem in Graphs

In this chapter, we present the different definitions and concepts of graph theory, as well as the different optimization problems and their methods , in this thesis we used dominating set as a methode to solve the probleme posed.

• The second chapter: Security in IOT

This chapter, is dedicated to talking about the main concepts and definitions of the Internet of Things (IoT) and how to protect it. Moreover, we will talk about smart city which is our application environment.

• The third chapter: Problem modeling and Simulation

In this chapter, we will present some basic consepts of simulation and more precisely the cupcarbon Similator .and we will present our conceptual and programming model that we will use to simulate the studied environment.

• The fourth chapter: Experimental results

In this chapter, we will present the environement of our experiment, which is the city of Mila, and the result that we obtained after applying our algorithm to the cupcarbon simulator .

Chapter 1

Domination Problem in Graphs

Introduction

In this chapter, we will start by giving an overlook of some basic concepts of graphs such as definition and notations that we will use in this thesis. After that, we will define briefly combinatorial optimization problems. Moreover, we will present a brief survey of graph domination problems.

1.1 Graph Basics

This section is devoted to introduce the basic concepts of graphs theory. Firstly, we give a short overview of standard graph terminology used throughout this thesis. Some others will be given later when necessary. Then, we talk about techniques used to store graphs in a computer. We will end this section by presenting the existing well-known graph classes and graph parameters in the literature.

1.1.1 Definition and notation

Definition 1.1 (Graph)

A graph-generally denoted G(V,E) or G= (V,E) is a pair consists of a finite non-empty set V of Vertices (nodes) and a set $\left(E \subseteq VxV\right)$ of Edges (links) that connect these vertices.

Example 1.1

The graph given in the figure as vertex set V=1,2,3,4,5,6 and edge set=(1,2),(1,5), (2,3),(2,5),(3,4),(4,5),(6,6).

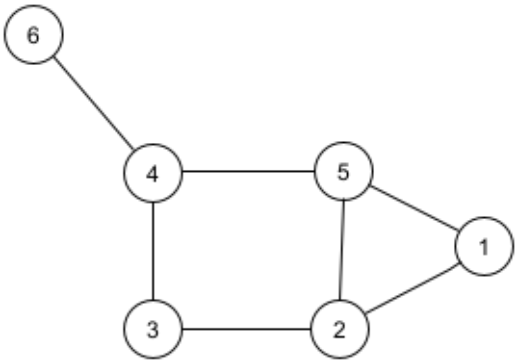

Figure 1.1: Simple undirected graph

Definition 1.2 (Order and size)

The cardinality n of the nodes set denoted by $(|\ V\ |)$ is called the order of G (n = $(|\ V\ |)$) and the cardinality m of the edges set denoted by $(|\ E\ |)$ is called the size of G (m = $(|\ E\ |)$).

Example 1.2

Order of the graph in the figure 1.1 is: $(|\ V\ |)$ =6
Size of the graph in the figure 1.1 is: $(|\ E\ |)$ =7

Definition 1.3 (Neighbors)

Two nodes u and v are adjacent or neighbors if they are connected by an edge, in other words, (u,v) is an edge, or $((u,v) \in E)$. The set of neighbors of a node v is denoted by N(v), i.e.$(N(v) = u \in V \mid (u,v) \in E)$. The closed neighborhood of a node v is denoted by $(N[v] = N(v) \cup v)$. And also, we notice that [12]:

- A vertex v is isolated if N(v) = Ø.
- A vertex v is universal if N[v] = V.
- Two distinct vertices u, v are twins if N(v) = N(u), or are false twins if N[v] = N[u].

Example 1.3

Neighbors of edge 1 of the graph in the figure 1.1 are: N (1) = 2,5
Neighbors of edge 4 of the graph in the figure 1.1 are: N (4) = 3,5,6

Definition 1.4 (Degree of a node)

The degree of a node v denoted by d (v) is the cardinality of its open neighborhood $(i.e.d(v) =|\ N(v)\ |)$, is represent the number of edges with v as an end vertex. And $(\triangle(G))$ represent the maximum node degree in the graph. By convention, we count a loop twice and parallel edges contribute separately :[7]

- A pendant vertex is a vertex whose degree is 1.
- An edge that has a pendant vertex as an end vertex is a pendant edge.
- An isolated vertex is a vertex whose degree is 0.

Example 1.4

Degree of edge 1 of the graph in the figure 1.1 is: d (1) = 2.
Degree of edge 4 of the graph in the figure 1.1 is: d (4) = 3.

Definition 1.5 (Path)

A path is a simple graph whose vertices can be ordered so that two vertices are adjacent if and only if they are consecutive in the list. A Simple path is a path in which all the vertices are distinct. The maximum length of the shortest path between any nodes is called diameter of G and it is denoted by D .[7]

Example 1.5

Path from edge 1 to edge 6 in the figure 1.1 is : 1→2→5→4→6

Definition 1.6 (Complete graph)

A simple graph G= (V, E) with n mutually adjacent vertices is called a complete graph G and it is denoted by (K_n) or A simple graph G= (V, E) in which every vertex in mutually adjacent to all other vertices is called a complete graph G .[7]

Example 1.6

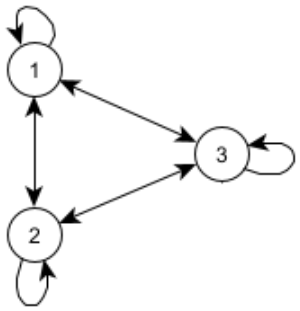

Figure 1.2: Oriented and simple graph (K_3)

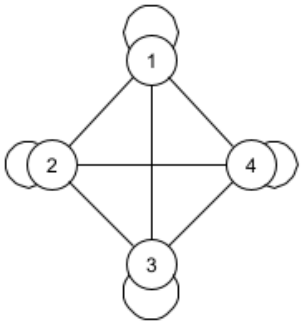

Figure 1.3: Not oriented and simple graph (K_4)

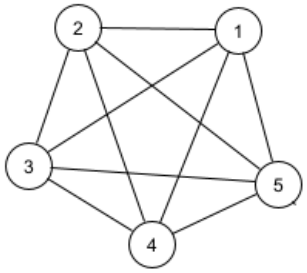

Figure 1.4: Not oriented and simple graph (K_5)

Definition 1.7 (Subgraphs and induced subgraphs)

In a simple graph G = (V;E), we say that G' = (V', E') is a subgraph of G if $(V' \subseteq V)$ and $(E' \subseteq (E \cap V' \times V'))$, if $(V' = V)$ we say that G is a spanning subgraph of G [9]. An induced subgraph of G denoted by $(G[S] or <S>)$ is the maximal subgraph of G with nodes set S $(\forall (u,v) \in S \times S)$, u, v are adjacent in G[S] if and only if they are adjacent in G.

Example 1.7

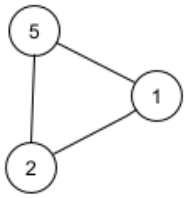

Figure 1.5: Subgraph G' of graph G in the figure 1.1

Definition 1.8 (Clique)

A subgraph induced by $(S_i = (v_i, \ldots, v_j))$ forms a clique Qi if and only if $(\forall v_j, v_k \in S_i : v_j v_k \in E)$, i.e, the number of edges in the clique. $(\frac{i(i-1)}{2})$ .if $(S_i \equiv V)$ then we say that G is a complete graph [9].

Example 1.8

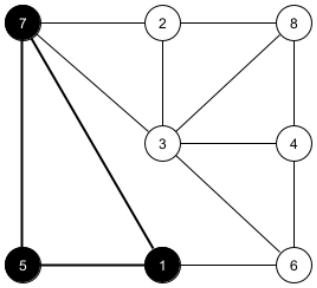

Figure 1.6: Clique

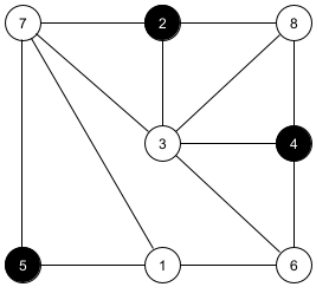

Figure 1.7: Stable

Definition 1.9 (Simple graph and Multigraph)

Multigraph is a graph which is permitted to have multiple edges (a.k.a. parallel edges) that have the same extremities. In other words, two vertices may be connected by more than one edge. A graph is simple if there is at most one edge between every two vertices.

Example 1.9

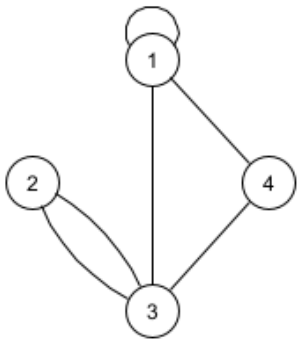

Figure 1.8:

Multigraph

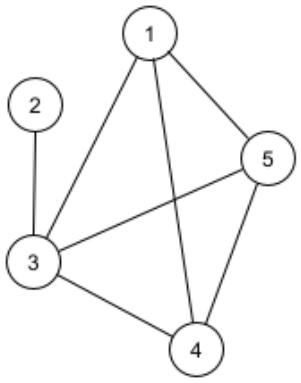

Figure 1.9: Simple graph

Definition 1.10 (Graph complement)

The complement of the graph G = (V,E) has V as nodes set $\left(\tilde{E}\right)$, but the edges set defined by $\left(\tilde{E} = \{uv\,\varepsilon\,E, uv/\varepsilon\,E\}\right)$.

Example 1.10

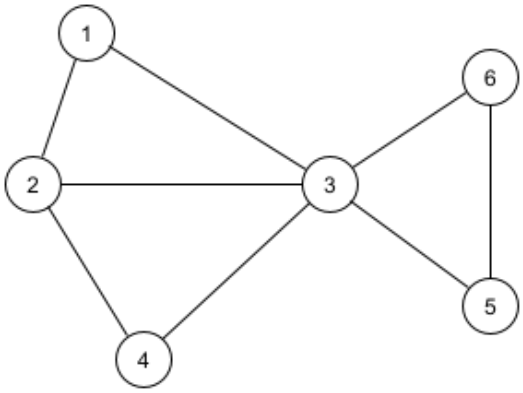

Figure 1.10: Graph complement

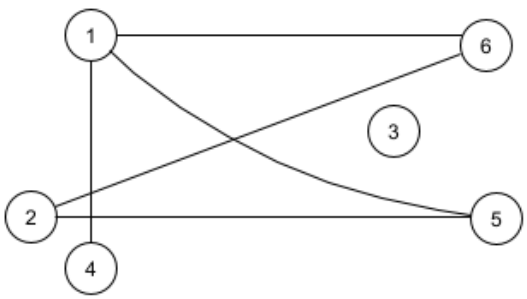

Figure 1.11: Graph not complement

1.1.2 Graph representation

Graph representation is a technique used to store graphs in a computer memory. To represent a graph, we just need a set of vertices, and for each vertex, the neighbors of the vertex. If the graph is weighted, the weight will be assigned to each edge. There are different ways to optimally represent a graph, depending on the density of its edges, type of operations to be performed and ease of use[8] .

1.2.1 Adjacency Matrix

Adjacency Matrix A is a binary matrix of size V×V where V is the number of vertices in a graph. If there is an edge from a vertex i to vertex j, then the corresponding element of A,$\left(a_{i,j}=1\right)$, otherwise$\left(a_{i,j}=0\right)$, Figure. If there is a weighted graph then instead of 1s and 0s, we can store the weight of the edge.

Example 1.11

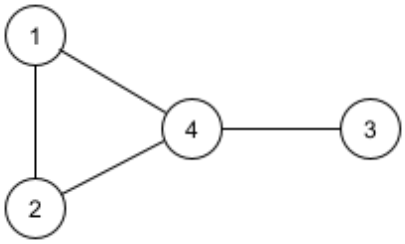

Figure 1.12: Adjacency matrix Undirected graph.

$$A_u = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

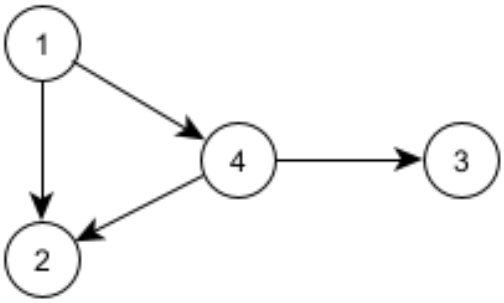

Figure 1.13: Adjacency matrix Directed graph.

$$A_d = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

1.2.2 Incidence Matrix

Incidence matrix B is a matrix of size $\left(|\,V\,| \times |\,E\,|\right)$. For undirected graphs, the entries of B are: $\left(b_{i,j} = 1\right)$, if vertex i incident to edge j otherwise 0. For directed graphs: $\left(b_{i,j} = 1\right)$, if edge j is (k, i). $\left(b_{i,j} = -1\right)$, if edge j is (i,k) otherwise 0, Figure.

Example 1.12

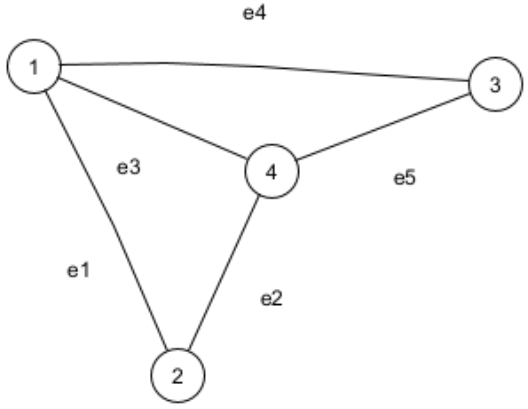

Figure 1.14: Incidence Matrix Bu

$$B_u = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

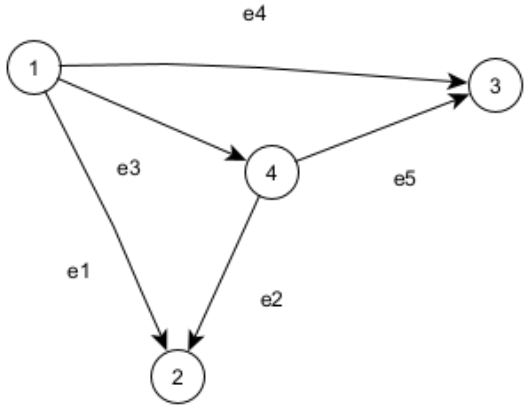

Figure 1.15: Incidence Matrix Bd

$$B_d = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & -1 & 0 & 1 \end{pmatrix}$$

1.2.3 Adjacency List

Adjacency list is an array of size $\left(|\,V\,|\right)$ where for each vertex in the graph, we maintain the list of its neighbors. It means, every vertex of the graph contains a list of its

adjacent vertices (The i-th array element is a list of the vertices adjacent to i), Figure 4.3. Similarly, an edge list stores the vertex pairs incident to each edge.

Example 1.13

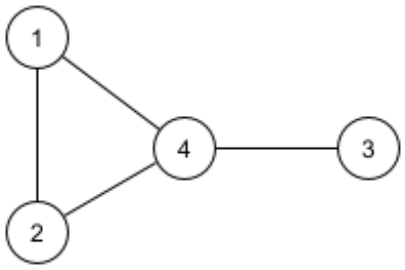

Figure 1.16: Adjacency List

$$L_a[1] = \{2,4\}$$
$$L_a[2] = \{1,4\}$$
$$L_a[3] = \{4\}$$
$$L_a[4] = \{1,2,3\}$$

1.1.3 Graph classes

A class of graph, which we can also sometimes call a family of graphs, is a set of graphs defined by one or more properties on the structure of the graph. Any graph G possessing the required property or properties belongs in fact to the corresponding class. Conversely, any graph G for which at least one property defining the class of graph does not correspond will in fact be outside this class of graph[32].

1.3.1. Bipartite graph

A bipartite graph (or bigraph), often denoted by G = (U, V, E), is a graph whose nodes set can be divided into two independent (disjoining) sets U and V , such that every edge connects a node of U to another one of V (there are no edges between U and V).

A bipartite graph is a graph does not contain any odd-length cycles. If each node of U is adjacent to each other of V the graph is called a complete bipartite graph.

Example 1.14

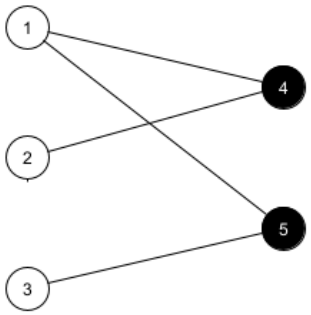

Figure 1.17: Bipartite graph

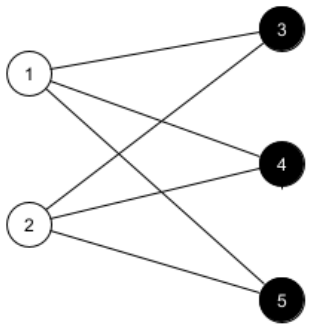

Figure 1.18: Complete Bipartite graph

1.3.2.Tree

An undirected connected graph is called tree if there are no cycles in it as induced sub graph. There is exactly one simple path between any vertices u and v. A tree with n nodes has exactly n-1 edges.

A forest is a disconnected graph whose components are trees. In some problems, we need to select a node from the rest called the root. The acyclic structure of trees makes them the most important data structures in computer science, the file system tree representation is an example of using such a data structure.

Example 1.15

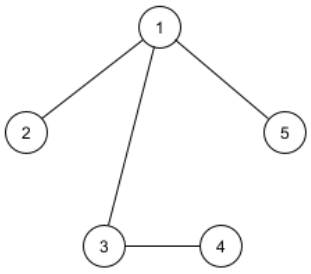

Figure 1.19: First Tree

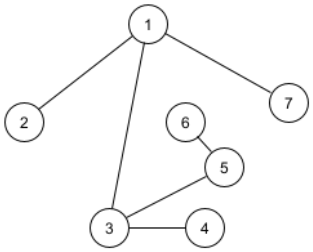

Figure 1.20: Second Tree

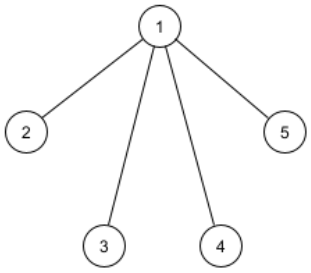

Figure 1.21: Third Tree

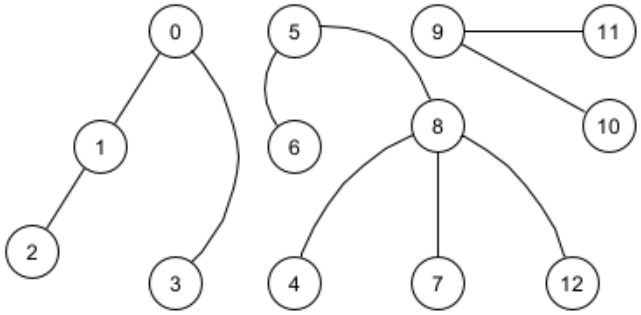

Figure 1.22: Forest graph

1.3.3.Planar graph

A graph is planar if it can be drawn in a plane without edges crossing, i.e., it can be drawn in such a way that no edges cross each other, and the only intersections of edges are at their endpoints. In other words, a plane graph is the graph that can be embedded in the plane. Planar graph of n nodes contains at most 3n - 6 edges. Embedded in plan property enables planar graphs to be the natural model in several fields including electrical networks, transport networks and electronic circuits.[9]

Example 1.16

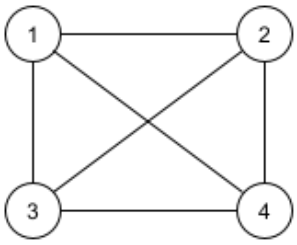

Figure 1.23: Un graphe G

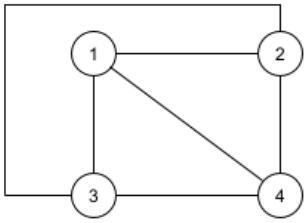

Figure 1.24: Representation planaire de G

1.3.4.Chordal graph

A graph G is said Chordal (triangulated) if every induced cycle in G should have atmost three vertices. It does not contain any induced cycle of length greater than three. In other words, it s a graph in which all cycles of four vertices or more have a chord, which is an edge that is not part of the cycle but connects two vertices of the cycle.

Example 1.17

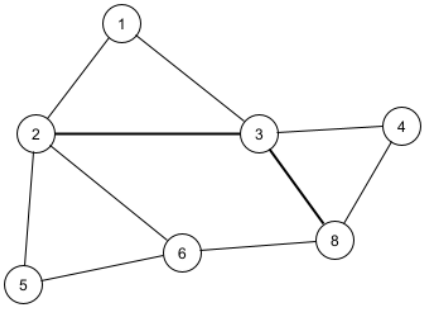

Figure 1.25: Chordal graph

1.3.5.Split graph

A split graph is a graph G(V, E) for which the set of nodes V can be divided into two subsets V1 and V2, i.e.,, such that V1 is a clique and V2 is an independent set.

Example 1.18

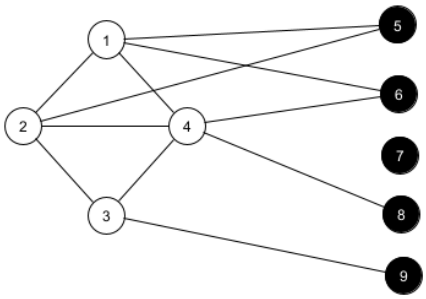

Figure 1.26: Split graph

1.3.6.Interval graph

An interval graph is an undirected graph formed by a set of intervals. Each interval represents a vertex and each edge that connects two vertices corresponds to the intersection of two intervals. An interval graph is said to be proper if and only if no interval is properly contained in another . Figure 2.7 represents an example of interval graph.

Example 1.19

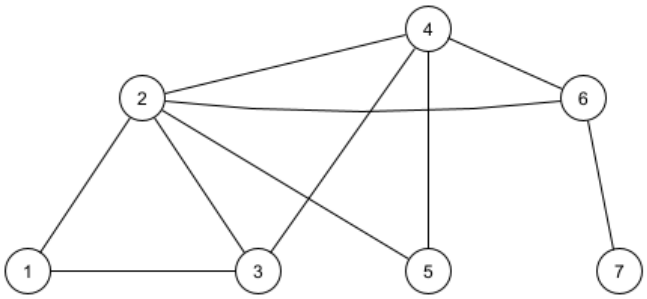

Figure 1.27: Interval graph

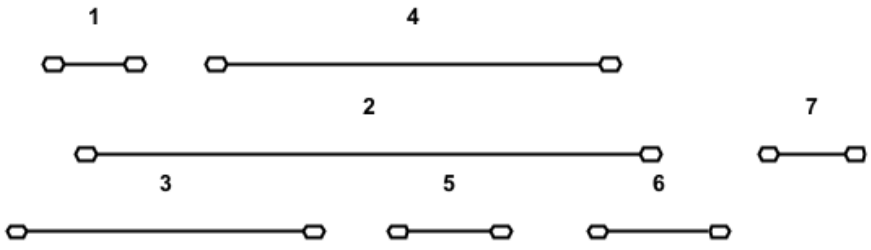

Figure 1.28: Interval graph

1.1.4 Graph parameters

Graph parameters are very useful in graph characterization and problems resolution. Generally, a real-life problem corresponds to a parameter in the graph modeling such one. A graph parameter is an invariant that depends only on the abstract structure of the graph (eg. number of nodes or edges) and not on the graph representation (eg. graph drawing) [9]. Determining a graph parameter leads clustering or decomposition of the graph nodes which facilitates the graph analysis. Determining the minimum number of color needed for node coloring in graph, the maximum independent set and minimum dominating set are examples of such a fact.

1.4.1.Vertex coloring

An assignment of colors to the vertices of a graph G so that no two adjacent vertices of G have the same color is called vertex coloring of a graph G. The minimum number of color needed for node coloration is called the chromatic number of G and is denoted by X(G).

Example 1.20

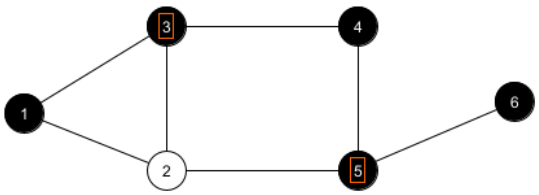

Figure 1.29: Vertex coloring

1.4.2.Spanning tree

Given an undirected and connected graph G= (V, E), a spanning tree of the graph G is a tree that spans G (that is, it includes every vertex of G) and is a sub-graph of G (every edge in the tree belongs to G). Sometimes, we need to restrict the graph to a spanning subgraph without cycles. Spanning tree is very useful, in communication networks to find the best path and in the electrical networks to minimize the wirings cost. Kruskal and Sollin are the best well known algorithms for determining minimal spanning trees .[9]

Example 1.21

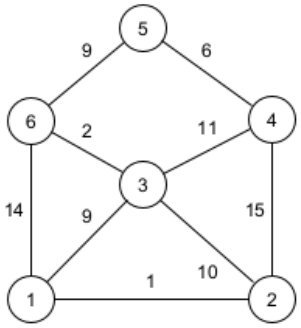

Figure 1.30: Undirected graph G=(V,E)

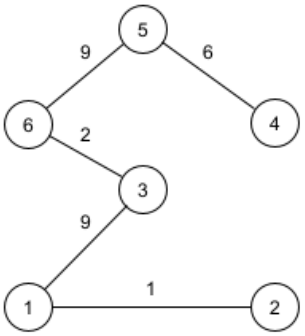

Figure 1.31: The minimal spanning tree the tree cost is 27

1.4.3.Shortest path

The shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. It can be defined for graphs whether undirected, directed, or mixed. The most important algorithms for solving this problem are Dijkstra's algorithm and Bellman–Ford algorithm .

Example 1.22

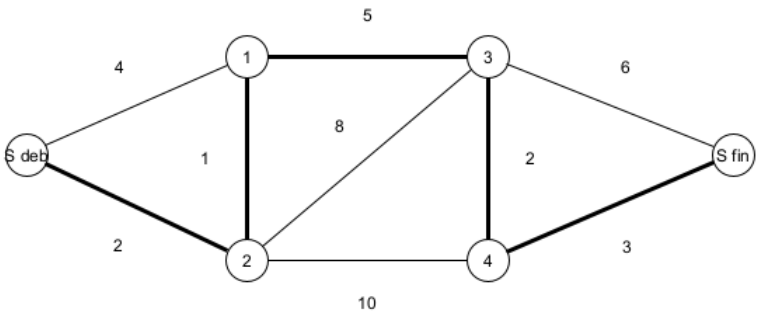

Figure 1.32: Shortest path

1.4.4.Independent set

A set of nodes $(I \leq V)$ is independent set or stable set if no two nodes in S are adjacent. A maximal independent set (MIS) is an independent set that is not properly contained in any independent set.However, as we have already seen, graph parameters are very useful in problem resolution but their determination is not always an easy task, and their related optimization problems are usually hard to solve [9]. For better understanding, before detailed the most interesting parameters or graph problems like domination and decomposition, we first introduce the different complexity classes of problems.

Example 1.23

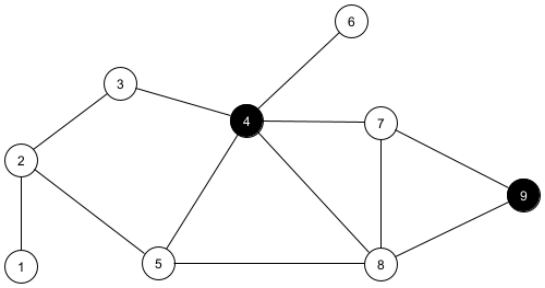

Figure 1.33: Independent set

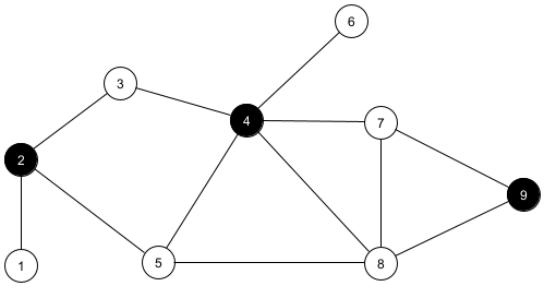

Figure 1.34: Maximal independent set

1.2 Optimization Problems

In this section we will give a brief overview of combinatorial optimization problems with their different types and some existing resolution methods.

1.2.1 Combinatorial problems

A Combinatorial problem is based on finding the best combination of solutions (there are an exponential number of possible combination). A combinatorial problem can be a decision problem, a research problem or an optimization problem .[1]

2.1.1. Decision problem

A decision problem is to check if some property is true or false. It is a problem that can be solved in terms of yes/no to know if there is a solution to the problem. Therefore, there is no need to find the actual solution because the two possible answers are yes or no .[2]

2.1.2.Research problem

The resolution of the problem (a.k.a. construction problem) is no longer base on knowing whether it admits a solution or not. The algorithm must be able to provide the solution if it exists. Therefore, any decision problem can be extended to a research problem.[3]

2.1.3.Optimization problem

An optimization problem is obtained from a research problem by associating for each solution a value. It depends on finding among a set of potential solutions the best that meets certain criteria, which is described as an objective function to maximize or minimize (it depends on the objective).

1.2.2 Complexity

Complexity is concerned with the formal study of the difficulty of problems in computer science. It differs from the theory of computability which focuses on whether a problem can be solved by a computer. Complexity theory therefore focuses on problems that can actually be solved, the question being whether or not they can be solved efficiently or not based on (theoretical) estimation of computation times and computer memory requirements.

2.2.1 Time complexity

The time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input. Note that the time to run is a function of the length of the input and not the actual execution time of the machine on which the algorithm is running on.

In order to calculate time complexity on an algorithm, it is assumed that a constant time c is taken to execute one operation, and then the total operations for an input length on N are calculated.

2.2.2 Space complexity

Space Complexity of an algorithm is the total space taken by the algorithm with respect to the input size. Space complexity includes both Auxiliary space and space used by input.

For example, if we want to compare standard sorting algorithms on the basis of space, then Auxiliary Space would be a better criterion than Space Complexity. Merge Sort uses O(n) auxiliary space, Insertion sort, and Heap Sort use O(1) auxiliary space. The space complexity of all these sorting algorithms is O(n) though.

Space complexity is a parallel concept to time complexity. If we need to create an array of size n, this will require O(n) space. If we create a two-dimensional array of size n*n, this will require O(n2) space.

2.2.3 Complexity classes

In computer science, there exist some problems whose solutions are not yet found, the problems are divided into classes known as Complexity Classes. In complexity theory, a Complexity Class is a set of problems with related complexity. These classes help scientists to groups problems based on how much time and space they require to solve problems and verify the solutions. It is the branch of the theory of computation that deals with the resources required to solve a problem.

2.3.1 Types of Complexity Classes

• P Class

The P in the P class stands for Polynomial Time. It is the collection of decision problems(problems with a "yes" or "no" answer) that can be solved by a deterministic machine in polynomial time.

• NP Class

The NP in NP class stands for Non-deterministic Polynomial Time. It is the collection of decision problems that can be solved by a non-deterministic machine in polynomial time.

• Co-NP Class

Co-NP stands for the complement of NP Class. It means if the answer to a problem in Co-NP is No, then there is proof that can be checked in polynomial time.

• NP-hard class

An NP-hard problem is at least as hard as the hardest problem in NP and it is the class of the problems such that every problem in NP reduces to NP-hard.

• NP-complete class

A problem is NP-complete if it is both NP and NP-hard. NP-complete problems are the hardest problems in NP.

1.2.3 Types of optimization problems

An essential step to optimization technique is to categorize the optimization model since the algorithms used for solving optimization problems are customized as per the nature of the problem. Let us walk through the various optimization problem types:

2.2.1. Objectives number classification

Optimization problems can be divided into two main classes depending on the number of the objectives, and hence the complexity of the problem. The two classes are the so-called: mono-objective optimization problems and the multi-objective optimization problems[1].

• Mono-Objective optimization problems

We say that a given problem P is a mono-objective optimization problem if there is only one objective in P to be optimized. In this case the optimal solution is clearly defined, it is the one that has the optimal cost, whether the problem is to be minimized or maximized.

Example of the mono-objective optimization problem is the following. If we are buying a car where the objective is the price lower than 3000dolars, we found two cars on the market: one of them costs 2900dolars and the other is 2800 dolars, we see that is a clear solution we just have one objective, the solution can be exact. Note that here we are considering only one objective, and that other objectives can be considered for the same problem (see the multiobjective example).

• Multi-objective optimization problems

Optimization problems involve more than one objective function to be optimized simultaneously. This kind of optimization problems arise in many fields, such as engineering, economics, and logistics, when optimal decisions need to be taken in the presence of tradeoffs between two or more conflicting objectives.[5]

The main difficulty of multi-objective problems is the no definition of the optimal solution. The decision maker can express that a solution is better than another but, there is no best solution exists. Solving a multi-objective problem does not consist in searching for an optimal solution but the set of satisfactory solutions. The resolution methods for multiobjective problems are just to help in decision because the final decision is always to the decision maker.[6]

Solving multi-objective problems can be done either by transforming them into monoobjective problems or dealing with them as a multi-objective problem.

2.2.2. Variables types classification

Optimization problems can be divided into two categories, depending on whether the variables are continuous or discrete :[4]

• Discrete optimization problems :

An optimization problem with discrete variables is known as a discrete optimization, in which an object such as an integer, permutation or graph must be found from a countable set

• Continuous optimization problems :

A problem with continuous variables is known as a continuous optimization, in which an optimal value from a continuous function must be found. They can include constrained problems and multimodal problems

1.2.4 Resolution methods

Generally, the resolution of an optimization problem goes through three main steps:

• Problem analysis: grouping the characterization of the problem and modeling of the problem.

• The expression of the objective to be optimized:This requires a good knowledge of the problem.

• The choice of the resolution method: This allows from the representation of the problem in a search space to obtain a solution or a set of optimal solutions with respect to the evaluation function[11]

To define the suitable approach for solving a given problem, we have first to set the problem complexity. Generally, the optimization problems can be solved with an exact algorithm or an approximate algorithm (see Figure). In exact algorithms, the solutions are optimal and their optimality is guaranteed. In case of NP-complete or NP-hard problems, exact algorithms run in non-polynomial time, which means it would take both important time and space to find a solution. However, approximate algorithms generate high quality solutions in a reasonable time, but there is no guarantee of finding an optimal solution.[10]

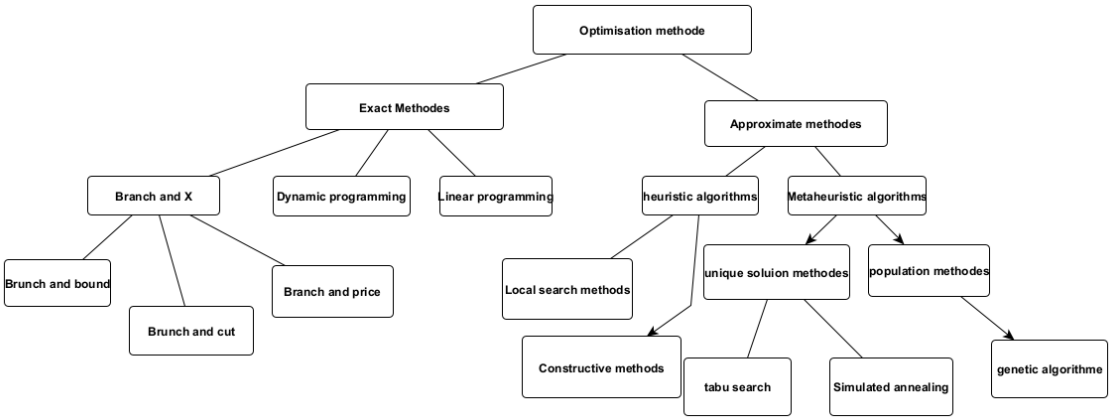

Figure 1.35: Resolution methods

2.3.1. Exact methods

In exact methods, all solutions in the search space are listed implicitly using mechanisms that detect failures like the calculation of bounds. Using these methods we can find optimal solutions. Nevertheless, they take a lot of time when it comes to a complicate problem (too much cases to verify), which is the case for NP-complete and NP-Hard problems. The most known of exact methods are the Branch and Bound algorithm, dynamic programming, branch and cut, branch and price, linear programming, etc.[1]

2.3.2. Approximate methods

They are also called incomplete methods due to the fact that they explore only a part of the solution space. Therefore, they may not find the optimal solution in some cases. However, there are some approximate methods that give a guaranteed solution close to the optimal, and give a ratio of how far it is from the optimal solution. Generally, we use approximate algorithms to solve big size problems. They can be divided into two subclasses, namely Heuristics and Metaheuristics [10].

1.3 Domination problems

The Domination problem is one of the most famous and known problems in graph theory. It is an NP-Complete problem, which means that there is no algorithm that can

resolve it in a polynomial time. It was studied from the 1950s onwards, but the rate of research on domination significantly increased in the mid-1970s until now [2].

1.3.1 Overview and definitions

Definition 1 (domination)

In a graph G = (V, E), a subset of vertices $\left(D \subseteq V\right)$ A is called a dominating set of G, if every node $\left(v \in V\right)$ is either a member of D or is adjacent to a member of D.

Definition 2 (minimal dominating set)

A dominating set D is minimal (minimal dominating set: MDS) if no proper subset of D is a dominating set. The dominating number $\left(\gamma(G)\right)$ is the minimum cardinality of a dominating set of G.

Example 1.24 $\left(\gamma(G) = 3\right)$

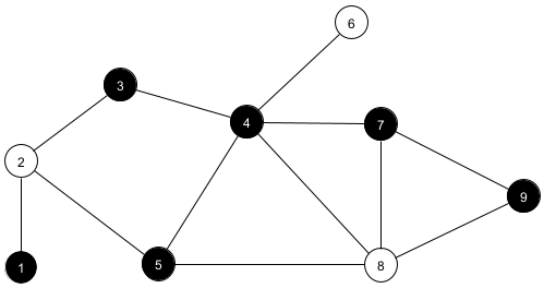

Figure 1.36: Dominating set

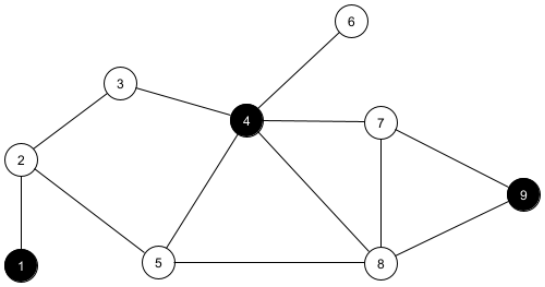

Figure 1.37: Minimal dominating set

1.3.2 Variants of dominating set problem

There are many variants of domination in graph according to, the dominating set D is connected, independent, a clique, a path etc. the following are the most popular variants:

• Total Dominating set

A set D is called total dominating set (TDS) if every node of the graph has a neighbor in D, i.e. $\left(\forall v \in V : N(v) \cap D \neq \emptyset\right)$.

The total dominating set differs from the ordinary dominating set in that in a total dominating set $\left(S^t\right)$, the members of $\left(S^t\right)$ are required to themselves be adjacent to a vertex in $\left(S^t\right)$, whereas is an ordinary dominating set S, the members of S may be either in S itself or adjacent to vertices in S.

Example 1.25

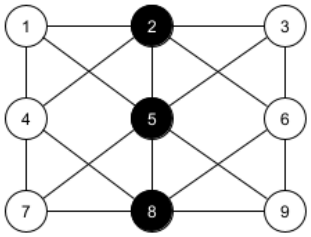

Figure 1.38: Total Dominating set

• K-dominating set

The set D is called k-dominating set (KDS) if every node outside of D has at least k neighbors inside D.

Example 1.26

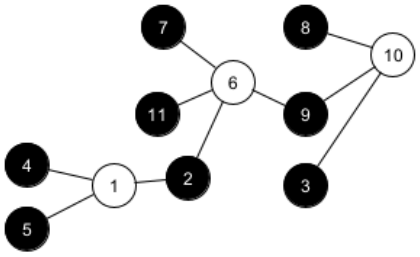

Figure 1.39: Dominating set (black vertices) for k = 3

• Connected dominating set

A dominating set is said connected dominating set (CDS) if D is connected.

Example 1.27

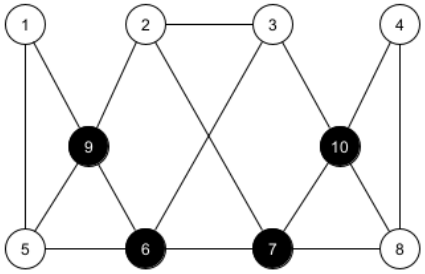

Figure 1.40: Connected dominating set

• Independent Dominating Set

A set is independent (or stable) if no two vertices in it are adjacent. An independent dominating set of G is a set that is bo

A lot of attention is given to this problem by studying the domination, a very interesting variant in the graph security field

Example 1.28

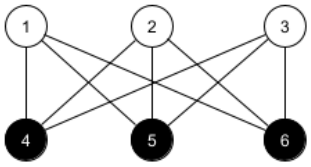

Figure 1.41: Independent Dominating Set

1.3.3 Aims and Motivation

The Dominating set is very important class of problem with several theoretical and practical applications. In practical side, the structure of dominating set can be useful as overlays in computer networks. These structures are usually used for designing efficient protocols in wireless sensor and ad-hoc networks. It is used also for optimization and designing protocols in social networks and many other fields [2]. For these reasons, different variants of dominating set have been identified by the graph community.

1.3.4 Algorithms

In the literature, many algorithms are known for the domination problem and its variants. But, there is no algorithm that can resolve it in a polynomial time. Therefore, we will present a greedy algorithm which was proposed in [15]:

Algorithm 1 centralized algorithm for dominating set in a graph

Input: G=(V, E) a non oriented graph.
Output: A dominating set $D \subseteq V$

```
Begin
  D ← ∅;
  while (V≠∅) do
        x ← argmax_{v∈V} (N_{G[V]}(v)) ;
      D ← D ∪ {x} ;
      V ← V–N [x] ;
  end while
End.
```

1.3.5 Secure Domination problem

The concept of secure domination was conceived by E. J. Cockayne and presented in a keynote lecture during a workshop at the University of South Africa in Pretoria, in 2002. And was first discussed by Cockayne et al [29]. After few years, it becomes a very interesting filed in different researches.

3.3.1. Overview and definitions

Definition 1(secure dominating)

A dominating set X of a graph G is said to be a secure dominating set (SDS) if each vertex $(u \in V/X)$ is adjacent to a vertex $(v \in X)$ such that $(X - \{v\} \cup \{u\})$ is a dominating set of G.

Definition 2(minimal secure dominating)

A secure dominating set X is minimal, if no proper subset of D is a secure dominating set. The secure domination number, denoted by $(\gamma_s(G))$, is the minimum cardinality of an SDS of G.

Exemple 1.29

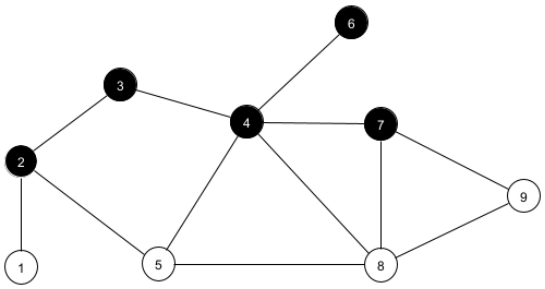

Figure 1.42: SDS

Conclusion

Graphs are powerful and useful tools used to model and solve real-life networks problems in different domains such as transportation, communication and computer science. In this chapter, we introduced the basic concepts required for better understanding of graph applications. Then, we presented the concepts of optimization problems and their resolution methods. We also provided a brief survey of algorithms proposed for dominating problems, dominating sets and secure dominating problems.

Chapter 2

Security in IOT

Introduction

In this chapter, we will start by giving an overlook of some basic concepts of IOT . After that, we will define briefly IOT security. Moreover, we introduce the basic concepts of smart cities.

2.1 IoT basic concepts

This section is devoted to introduce the basic concepts of IOT. Firstly, we give a short definition of IOT . Then we talk about how its works and its advantages. After that, we will present architecture of IOT , stages and components of IOT . We will end this section by presenting a reference architecture of IOT.

2.1.1 Definition of IOT (Internet of Things)

Internet of Things (IoT) is a network of physical objects or people called “things” that are embedded with software, electronics, network, and sensors that allows these objects to collect and exchange data. The goal of IoT is to extend to internet connectivity from standard devices like computer, mobile, tablet to relatively dumb devices like a toaster. IoT makes virtually everything “smart,” by improving aspects of our life with the power of data collection, AI algorithm, and networks. The thing in IoT can also be a person with a diabetes monitor implant, an animal with tracking devices, etc. This IoT tutorial for beginners covers all the Basics of IoT.[16]

Figure 2.1: IOT paradigm

2.1.2 IoT architecture

Iot architecture is the flow of information or data from the sensors to the large server clouds. The sensors are attached to the “things” and they take in information from the surroundings. Large cloud servers perceive, store and process the incoming data to generate necessary outputs. Data is sent back through the clouds, to the “things” to generate a chain reaction[17].

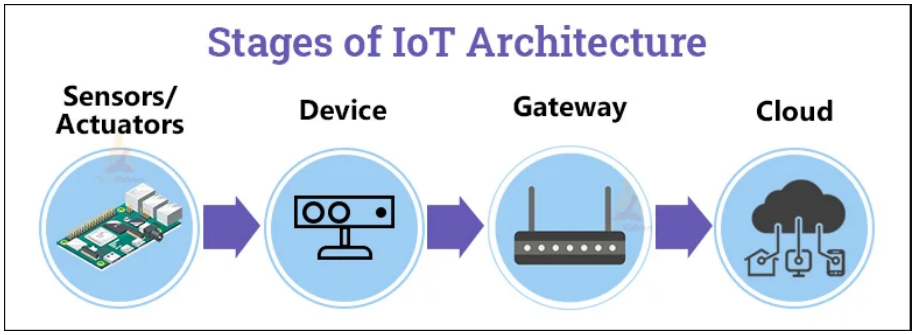

Figure 2.2: IoT architecture

2.1.3 IoT Works

The entire IoT process starts with the devices themselves like smartphones, smartwatches, electronic appliances like TV, Washing Machine which helps you to communicate with the IoT platform.[16]

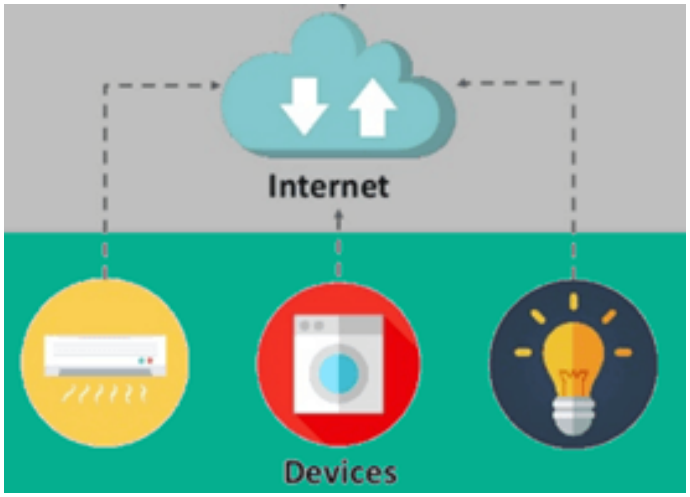

Figure 2.3: IOT works

• Sensors/Devices :

Sensors or devices are a key component that helps you to collect live data from the surrounding environment. All this data may have various levels of complexities. It could be a simple temperature monitoring sensor, or it may be in the form of the video feed. A device may have various types of sensors which performs multiple tasks apart from sensing. Example, A mobile phone is a device which has multiple sensors like GPS, camera but your smartphone is not able to sense these things.

• Connectivity :

All the collected data is sent to a cloud infrastructure. The sensors should be connected to the cloud using various mediums of communications. These communication mediums include mobile or satellite networks, Bluetooth, WI-FI, WAN, etc.

• Data Processing :

Once that data is collected, and it gets to the cloud, the software performs processing on the gathered data. This process can be just checking the temperature, reading on devices like AC or heaters. However, it can sometimes also be very complex like identifying objects, using computer vision on video.

• User Interface:

The information needs to be available to the end-user in some way which can be achieved by triggering alarms on their phones or sending them notification through email or text message. The user sometimes might need an interface which actively checks their IoT system. For example, the user has a camera installed in his home. He wants to access video recording and all the feeds with the help of a web server. However, it's not always one-way communication. Depending on the IoT application and complexity of the system, the user may also be able to perform an action which may create cascading effects. For example, if a user detects any changes in the temperature of the refrigerator, with the help of IoT technology the user should able to adjust the temperature with the help of their mobile phone.

2.1.4 Stages of IoT Architecture

The designing of the contents of an Iot system is known as the IoT architect. It works to deliver services through a network of Iot devices and servers[17].

The following are the stages of an IoT architecture:

Stage 1: Perception Layer

First, we have the perception layer that contains Iot devices such as sensors, actuators and machines that have the capacity to sense, calculate and connect to other devices. Sensors sense physical changes in the surroundings and gather information. This layer can be called the client side as this layer fits into the clients location or address.

The collected information passes on to the aggregation layer or the Iot gateway. This layer combines and computes the data. Operators control this layer and this layer involves servers.

Stage 2: Transport Layer or connectivity

Basically, this layer transports information. It transports physical data from sensors and IoT devices to clouds and servers. It then transports back generated responses back to the appliances. The transportation takes plays through a network or via gateways.

The use of various technologies helps in the transportation of data. The most common ones are WiFi, ethernet, Zigbee, Bluetooth, LoRa and cellular networks.

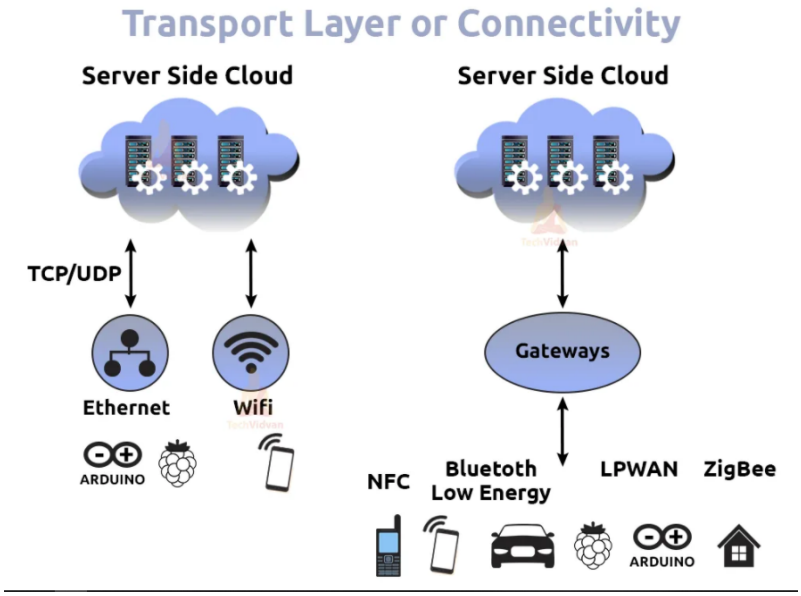

Figure 2.4: Transport Layer or connectivity

Stage 3: Event Processing Layer

The next layer is established on cloud and it is called the event processing layer. It has various algorithms and data calculation code written in it and it processes the information obtained from the sensor layer. It is responsible for collecting all data and information coming from various IoT devices.

Stage 4: API Management Layer

The final layer is the API management layer or the application layer. This layer is the interface between the third party applications and infrastructure or users. Device managers, identity and access managers which provide safety and security back the entire system at every stage.

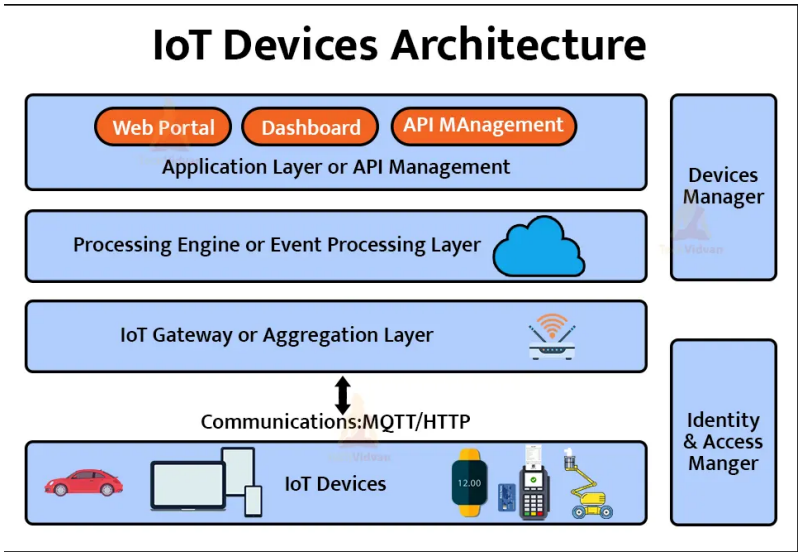

Figure 2.5: API Management Layer

2.1.5 Advantages of IoT

• Technical Optimization:

IoT technology helps a lot in improving technologies and making them better. Example, with IoT, a manufacturer is able to collect data from various car sensors. The manufacturer analyzes them to improve its design and make them more efficient.

• Improved Data Collection:

Traditional data collection has its limitations and its design for passive use. IoT facilitates immediate action on data.

• Reduced Waste

IoT offers real-time information leading to effective decision making and management of resources. For example, if a manufacturer finds an issue in multiple car engines, he can track the manufacturing plan of those engines and solves this issue with the manufacturing belt.

• Improved Customer Engagement:

IoT allows you to improve customer experience by detecting problems and improving the process.[16]

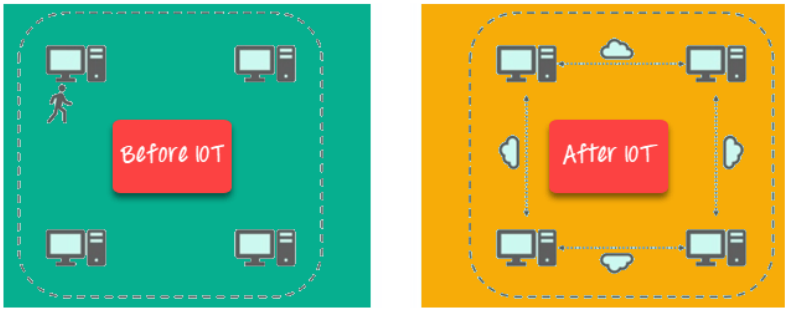

Figure 2.6: IOT advantages

2.1.6 Components of IoT architecture

An IoT architecture mainly consists of the following components [17]:

• Sensing, embedded components:

This layer provides accurate and credible data. It collects information from the surroundings.Sensors sense or detect even the slightest changes in the environment. Whereas, actuators respond or act on the signals they receive. For example, temperature control in smart thermostats.

• Connectivity:

Networking, communication and connectivity are the fundanets of any Iot ecosystem. Without device communication and connectivity, there is in fact no IoT. IoT protocols transfer data from one place to another. The most common wireless protocols are WiFi, Zigbee, LoRa and cellular etc.

Gateways are a mode through which the data passes to reach the cloud or servers. Gateways provide security by limiting unauthorized access.

• IoT cloud:

Cloud stores all the incoming data. Here data processing takes place with the help of data analysis and actions are performed on the data to generate a response in the system. Edge computing is done when there is large amounts of incoming data from the user.

• Data management:

This is a proper mechanism that stores the data and remembers information for future responses. Regardless of any IoT project, IoT uses some common components:

1. Devices
2. Connectivity
3. Platform
4.Data analytics
5. Applications

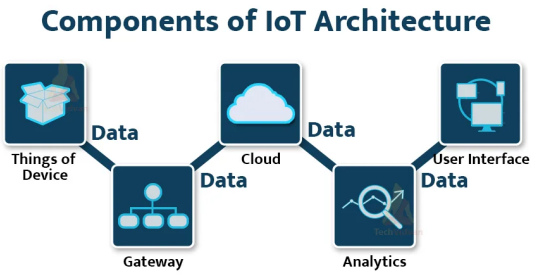

Figure 2.7: Components of IoT architecture

2.1.7 A Reference Architecture

The following reference architecture can provide a detailed explanation[17]:

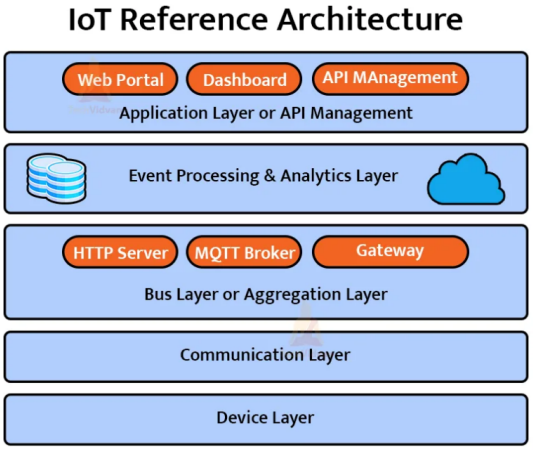

Figure 2.8: A Reference Architecture

The device layer is the main section which contains various applications that contain sensors such as the Bluetooth via mobile phones, zigbee via zigbee gateway or the raspberry pie that is connected to Ethernet. These devices are together connected to the communication layer. Both layers generate a large amount of data and information.

Next, the bus layer or the aggregation layer forms a bridge between the data and the communication layer. This layer contains HTTPS server, MQTT broker and it aggregates and combines information via gateway.The event processing layer drives the data and transforms the data generated. The data gets stored in the databases.

We create a web based engine to interact with other APIs with the help of client layers. This layer contains the dashboard which provides a view of the functions that can be performed. This layer communicates outside the network.

2.2 IoT Security

This section is devoted to introduce the basic concepts of IoT Security. Firstly, we give a short definition of IoT security . Then we talk about its uses . After that, we will present security features of IOT . We will end this section by presenting Security Threats in Each layer.

2.2.1 Defintion of IOT security

IoT security covers both physical device security and network security, and impacts the processes, technologies, and measures necessary to protect IoT devices and networks. It spans industrial machines, smart energy grids, building automation systems, entertainment devices, and more, including devices that often aren't designed for network security. IoT device security must protect systems, networks, and data from a broad spectrum of IoT security attacks, which target four types of vulnerabilities[18]:

- Communication attacks on the data transmitted between IoT devices and servers.
- Lifecycle attacks on the IoT device as it changes hands from user to maintenance.
- Attacks on the device software.
- Physical attacks, which directly target the chip in the device.

2.2.2 Uses of IoT security

A robust IoT security portfolio allows developers to protect devices from all types of vulnerabilities while deploying the security level that best matches their application needs. Cryptography technologies help combat communication attacks, while security services can protect against lifecycle attacks. Isolation measures can be implemented to fend off software attacks, and tamper mitigation and side-channel attack mitigation technologies are essential against physical attacks on the chip[18].

2.2.3 Firewall

2.3.1 Definition

A firewall is a network security device that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules.

Firewalls have been a first line of defense in network security for over 25 years. They establish a barrier between secured and controlled internal networks that can be trusted and untrusted outside networks, such as the Internet.

A firewall can be hardware, software, or both[34].

2.3.2 Firewall work

A firewall acts as a gatekeeper. It monitors attempts to gain access to your operating system and blocks unwanted traffic or unrecognized sources.

A firewall acts as a barrier or filter between your computer and another network such as the internet. You could think of a firewall as a traffic controller. It helps to protect your network and information by managing your network traffic. This includes blocking unsolicited incoming network traffic and validating access by assessing network traffic for anything malicious like hackers and malware[35].

2.3.3 Types of firewalls

There are several different types of firewalls based on their structure and functionality. Here are the various firewalls can implement, depending on the size of network and the level of security need[35].

- Packet-filtering firewalls.
- Proxy service firewalls.

- Stateful multi-layer inspection (SMLI) firewalls.
- Unified threat management (UTM) firewalls.
- Next-generation firewalls (NGFW).
- Network address translation (NAT) firewalls.
- Virtual firewalls.

2.2.4 Security features of IoT

The security challenges of IoT can be broadly divided into two classes; Technological and Security objection . The technological challenges come due to the different and pervasive nature of IoT devices, while the security provocation is related to the ethics and usefulness that should be implemented to attain a secure network. Security should be included in IoT throughout the growth and running lifecycle of all IoT devices and hubs.Given below are the security principles that should be followed to achieve a secure interaction framework for the people, software, processes, and things in an IoT[19].

• Confidentiality

It is important to ensure that data is secure and only available to approved users. Integrity. The IoT is based on interchanging data and information between many various types of devices, which is why it is important to confirm accuracy of the data; that data is being comes from the right sender as well as to make sure that the data is not modified with during the process of transmitting due to intended or unintended interference.

• Availability

The vision of IoT is to join as many smart devices as possible. The users of the IoT should have all the data visible whenever they need it. However data is not the only modules that is used in the IoT; devices and services must also be approachable and accessible when needed in a timely fashion in order to achieve the predictions of IoT.

• Authentication

Each object in the IoT must be clever to clearly identify and authenticate other objects. However, this process can be testing because of the nature of the IoT; many entities are mixed up (devices, people, services, service providers and processing units. In addition, sometimes objects may need to communicate with other objects for the first time (objects they do not know) [8]. Because of all this, a methods to mutually authenticate entities in every communication in the IoT is required.

• Lightweight Solutions

All of the security intentions considered earlier is not peculiar to IoT, although it may add special characteristics and constraints to each of them. However, in general confidentiality, integrity, availability and authentication are treated as basic intention in every computer or network security.

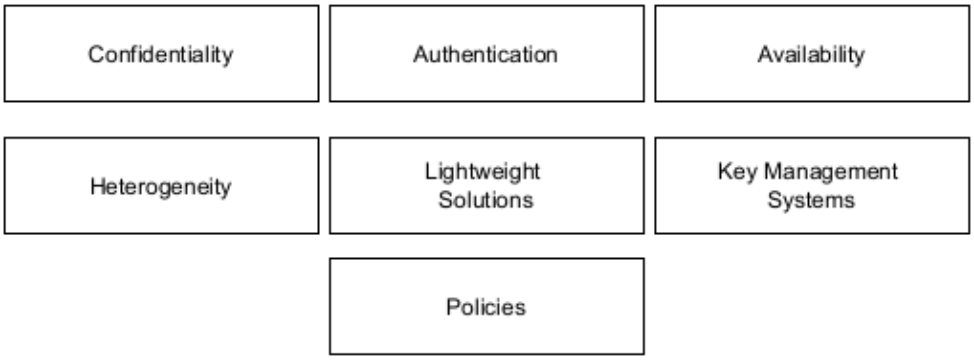

Figure 2.9: IOT secuirity principles

• Heterogeneity

The IoT connects different entities with contrasting potential, complexity, and vendors. The devices even have desperate dates and discharge versions, use desperate technical interfaces and bitrates, and are designed for an altogether different functions. Therefore obligations must be outlined to work in a variety of devices as well as in distinctive situations . The IoT aims at connecting device to device, human to device, and human to human, thus it implements connection between different things and networks . One more challenge that must be considered in IoT is that the environment is always changing (dynamics), at one time a device might be linked to a completely distinctive set of devices than in another time. And to ensure security optimal cryptography system is needed with an adequate key management and security protocols.

• Policies

There must be policies and standards to ensure that data will be managed, protected, and transmitted in an efficient way, but more importantly a mechanism to accomplish such plan is needed to assure that every entity is implementing the standards.

Service Level Agreements (SLO) must be clearly identified in every service involved. The enforcement of such guidelines will recommend trust by human users in the IoT model which will hereafter result in its growth and scalability.

• Key Management Systems

In IoT, the devices and IoT sensors need to interchange some encryption materials to achieve confidentiality of the data. For this intention, there needs to be a lightweight key management system for all structures that can enable trust between different things, and can deliver keys by consuming devices' minimum capacity.

2.2.5 Security Threats in Each layer

Each IoT layer is manageable to security danger and attacks. These can be active, or passive, and can derive from external sources or internal network possess to an attack by the Insider . The active attack through stops the service while the differential kindobserves IoT network information without inhibit its service. At each layer, IoT devices and services are sensitive to Denial of Service attacks (DoS), which make the device, resource

or network unavailable to approved users. The security problems at each layer are stated in Table 2, and given below is a brief examine of these problems with respect to each layer[19].

• Perception Layer

There are three security issues in IoT perception layer. First is the influence of wireless signals. Mostly the signals are sent between sensor nodes of IoT using wireless technologies whose effectiveness can be weaken by convulsing waves. Secondly, the sensor node in IoT devices can be attacked not only by the owner but also by the attackers because the IoT nodes usually run in external and outdoor environments, leading to physical attacks on IoT sensors and devices in which an attacker can tinkle the hardware components of the device. Third is the characteristic nature of network model which is dynamic as the IoT nodes are often moved around different places. The IoT perception layer mostly consists of sensors and RFIDs, due to which their storage kill, power utilization, and computation potential are very limited making them responsive to many kinds of dangers and attacks

The confidentiality of this layer can easily be abuse by Replay Attack which can be made by spoofing, modifying or replaying the identity information of one of the devices in IoT. Or the attacker might receive the encryption key by review the required time to perform the encryption what is known as Timing Attack. Another confidentiality dangering attack is when the attacker takes over the node and seizures all information and data which is basically Node Capture attack. Attacker can add another node to the network that threatens the integrity of the data in this layer by sending Malicious Data. This can also lead to a DoS attack, by consuming the energy of the nodes in the system and depriving it from the sleep mode that the nodes use to save the energy

The above listed security issues at perception layer can be coped with by using encryption (which can be point-to-point o end-to-end), authentication (to verify true identity of sender) and access control . Further security measures and protocols to address this issue are given in Section 4.

• Network Layer

As mentioned before, the network layer of IoT is also manageable to DoS attacks. Apart from the DoS attacks, the opponent can also attack the confidentiality and privacy at network layer by traffic examining, eavesdropping, and passive monitoring .

These attacks have a high likelihood of occurrence because of the remote access mechanisms and data interchange of devices.

The network layer is highly manageable to Man-in-the-Middle attack, which can be followed by eavesdropping. If the keying material of the devices is eavesdropped, the secure interacting channel will be completely weakened. The key exchange mechanism in IoT must be secure enough to prevent any intruder from eavesdropping, and then accomplish identity theft.

• Application Layer

Since the IoT still does not have exhaustive policies and standards that supervise the communication and the enlargement of applications, there are many problems related to the security. Different software and applications have different authentication mechanisms, which makes unification of all of them very hard to guarantee data privacy and identity authentication. The large amounts of associated devices that share data will cause large elevated on applications that analyze the data, which can have big influence on the availability of the services.

2.3 Smart City

This section is devoted to introduce the basic concepts of smart city. Firstly, we give a short definition . Then we talk about key components of a smart city . After that, we will present its IoT technology and wireless technology for smart cities . We will end this section by giving an overview of success factors and technology solutions of smart city .

2.3.1 Definition:

A smart city is an economically vibrant city that provides its citizens a good quality of life using Information Communication and Technology (ICT) solutions. The Smart Cities Council 1 defines a smart city as one that uses digital technology for all city functions. Similarly, World Bank2 defines a smart city as a technology-intensive city that has sensors installed everywhere and offers highly efficient public services using information gathered in real time by thousands of interconnected devices. Further, a smart city cultivates a better relationship between citizens and governments using the available technology. It relies on feedback from citizens to help improve service delivery. It puts in place mechanisms to gather this information[20].

Figure 2.10: Smart city

2.3.2 Key Components of a Smart City

- Core Infrastructure.
- High Quality of Life.
- Smart Solutions overlying Basic amenities.
- Clean and sustainable environment[20].

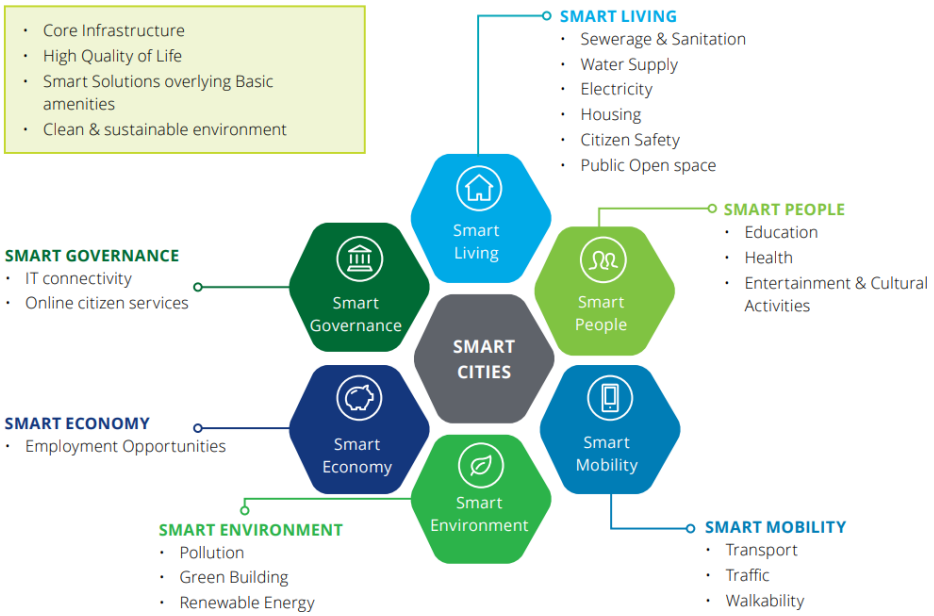

Figure 2.11: Components of a Smart City

2.3.3 IoT technology of smart city

Secure wireless connectivity and IoT technology are transforming traditional elements of city life - like streetlights - into next-generation intelligent lighting platforms with expanded capabilities.The scope includes integrating solar power and connecting to a cloud-based central control system that connects to other ecosystem assets.These solutions shine far beyond simple lighting needs[21]:

- High-power embedded LEDs alert commuters about traffic issues, provide severe weather warnings, and provide heads up when fires arise, for example.
- Streetlights can also detect free parking spaces and EV charging docks and alert drivers where to find an open spot via a mobile app. Charging might even be able from the lamppost itself in some locations!

2.3.4 Wireless technology for smart cities

The first building block of any smart city application is reliable, pervasive wireless connectivity. While there's no one-size-fits-all, evolving Low Power Wide Area Network (LPWAN) technologies are well suited to most smart city applications for their cost efficiency and ubiquity.These technologies include LTE Cat M, NB-IoT, LoRa, Bluetooth, and a few others that all contribute to the fabric of connected cities. The advent of 5G technology is expected to be a watershed event that propels smart city technology into the mainstream and accelerates new deployments[21].

2.3.5 Smart city success factors

In addition to people, dwellings, commerce, and traditional urban infrastructure, there are four essential elements necessary for thriving smart cities [21]:

- Pervasive wireless connectivity
- Open data
- Security you can trust in
- Flexible monetization schemes

2.3.6 Smart city technology solutions

When governments embark on a smart city project, they often invest in a combination of intelligent digital solutions. These smart city technologies are designed to work together to connect the community, enhance the lives of city residents, drive sustainability efforts, improve infrastructure, and support economic growth.

Here are a few examples of innovative smart city solutions [31]:

- Cloud computing
- Artificial intelligence (AI)
- Internet of Things (IoT)
- Blockchain technology
- Edge
- Augmented reality (AR)

Conclusion

Security is one of the most important things that all networks, especially the Internet of Things, should have due to its great importance in protecting things from being hacked.In this chapter, we introduced the basic concepts required for better understanding of IOT. Then, we presented the security of IOT.We also provided smart cities.

Chapter 3

Problem modeling and Simulation

Introduction

In this chapter, we will start by giving an overlook of some basic concepts of simulation such as General Principles Of simulation. After that, we will define briefly cupcarbon .Moreover,we will present our conceptual and programming model.

3.1 Basic Concepts of simulation

This section is devoted to introduce the basic concepts of simulation. Firstly, we give a short overview of General principles of simulation. We will end this section by presenting the Model and how modeling.

3.1.1 General principles of simulation

Etymologically, the term ”simulation” is derived from the Latin word ”SIMULARE” which means: to copy, to pretend, to make appear as real something that is not. Thus, the simulation can be defined as follows:

1.1.1 What is simulation

Simulation is experimentation on a model. It is a procedure of scientific research which consists in carrying out an artificial reproduction (model) of the phenomenon which one wishes to study, in observing the behavior of this reproduction when one experimentally varies the actions which one can exert on this, and to infer from it what would happen in reality under the influence of analogous actions[22].

In our project, we simulated an iot envirement, which we applied to the city of Mila, and to protect it from penetration by installing the least number of firewalls. To achieve this, we placed sensors in multiple areas of the city connected to each other by wifi or bluetooth, and each sensor contains a An algorithm whose job is to determine where to put the firewall

1.1.2 When to simulate

This question is related to the characteristics of the problem itself and the choice of a method for solving it. In general, there are two ways to solve a problem: analytically and experimentally. Analytical methods often lead to an optimal solution, while experimental methods often lead to a good solution, but not necessarily to the optimal solution. Thus,

simulation can be seen as the conduct of an indirect experiment (on the model and not on the system) [24]. It is often characterized as the method of last resort,thus if one has the possibility of solving the problem posed with analytical methods, it would be preferable to use them because they lead to optimal solutions.

The simulation of a real system becomes necessary as soon as the analytical models become either too complex in terms of calculation and resolution time, or too simplified with respect to reality, thus rendering the results obtained are not representative of the behavior of the system in a real environment. [24]:

1.1.3 Why simulate

In our project we will simulate for the following reasons:

- Experiments on the real system are very expensive in terms of material resources (the cost of sensors) and human resources.
- Real system experiments are not repeatable and do not represent all possible environments.

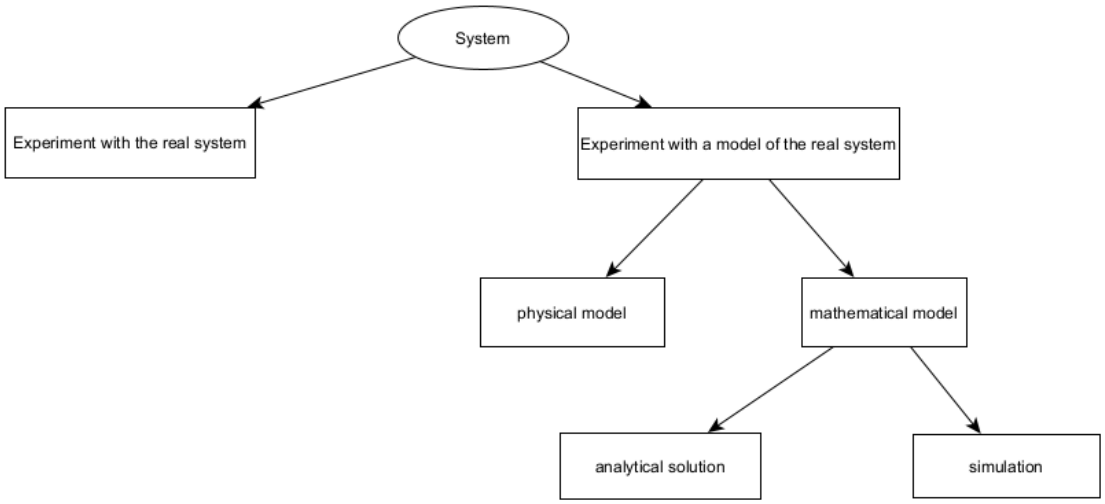

Figure 3.1: Approaches to study a system

1.1.4 Simulation steps

The steps that make up any simulation project can be schematized by the flowchart below [24]:

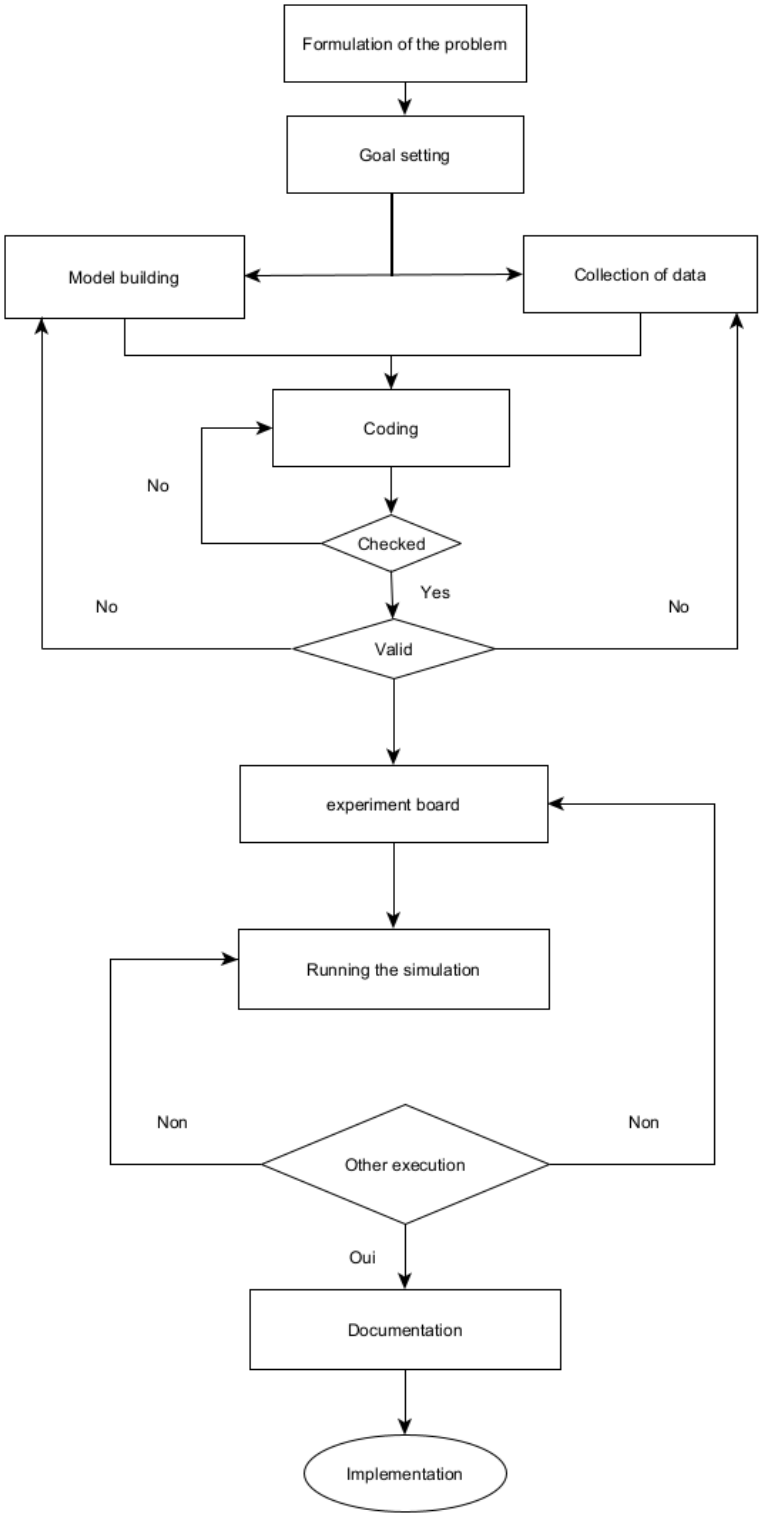

Figure 3.2: Simulation steps

• **Formulation of the problem**

Any simulation project begins with the statement of the problem to be solved.

• Goal setting

Once the problem has been formulated, the objectives targeted by the simulation project must be defined. This includes: the questions that the simulation study that we want to conduct will have to answer, the personnel, hardware and computer software required, the various scenarios that we want to investigate, the costs of the study as well as the time required, etc..

• Model building

It is a question of building a conceptual model which is an abstraction of the real system. This model can be seen as a set of mathematical and logical relations concerning the components and the structure of the system .

• Collection of data

Once the problem has been formulated and the targeted objectives have been identified, it will be necessary to establish an inventory of data needs on the real system .

• Coding

This involves translating the conceptual model obtained in step 3 into a form acceptable to the computer (i.e. program, also called operational model) .

• Verification

The verification step is extremely important in any simulation project. It concerns the operating model (the program). This is to ensure that the model runs without errors. Verification is essential even for small models because they can also contain errors .

• Validation

Validation consists of ensuring that the conceptual model is an accurate representation of the real system. The question is whether the model can be substituted for the real system for the purpose of experimentation. In the case where the system exists, the ideal way to validate the conceptual model is to compare its outputs with those of the system. Unfortunately, we do not always have this possibility, especially in new system design projects [34].

• Design of an experimental framework

This involves defining for each scenario to be simulated or experimented with a certain number of parameters such as: the duration of the simulation, the number of simulations to be carried out (replications), the initial state of the model and the rules for managing Waiting lines. The program can provide the possibility to separate the experiment framework which contains all the data and information to run the simulation. Thus, it will be possible to carry out different experiments on the same model by only changing the experimental framework .

• Execution of the simulation and analysis of the results

The operational model (or the program) is the main support for carrying out a computer simulation. It outputs purely statistical results (average, minimum, maximum, etc.).

The analysis of these results will aim to estimate the performance measures scenarios that we have experienced .

• Additional executions

At this level, we have a set of results from the various simulations that we have carried out as well as an analysis of these results. It will be a question of determining on the basis of this analysis if other simulations must be made, if other unforeseen scenarios must be tested in order to ensure that the model meets the objectives targeted in stage 2 .

• Documentation

Documentation is necessary, and it concerns both the model and the simulation results. If the model will one day be reused by other people, the documentation will help them understand how the model works and facilitate any modifications or updates to the model .

• Implementation

The objective of any simulation is to propose several solutions for a problem. The choice of the best solution must be made by the analyst who will justify it in the documentation and propose it to the client. The decision to retain this solution for a possible implementation therefore remains the responsibility of the customer .

3.1.2 Model and modeling

Modeling consists in building a simplified representation of a system generally called: model.

1.2.1 Definition

A definition of a model set out by AFCET is as follows:

”A model is a diagram, i.e. a mental (internalized) or figurative (diagrams, mathematical formulas, etc.) description which, for a field of questions, is taken as an abstract representation of a class of phenomena, more or less skilfully taken out of their context by an observer to serve as a support for investigation and/or communication”.

A model is therefore a representation of a system (real or imagined) whose purpose is to explain and predict certain aspects of the behavior of this system.

This representation is more or less faithful because on the one hand the model must be quite complete in order to be able to answer the various questions that can be asked about the system it represents and on the other hand it must not be too complex. so that it can be easily handled. This immediately implies that there is interest in clearly defining the limits or boundaries of the model that is supposed to represent the system[23].

1.2.2 Modeling process

The process of building a simulation model can be schematized as follows[23]:

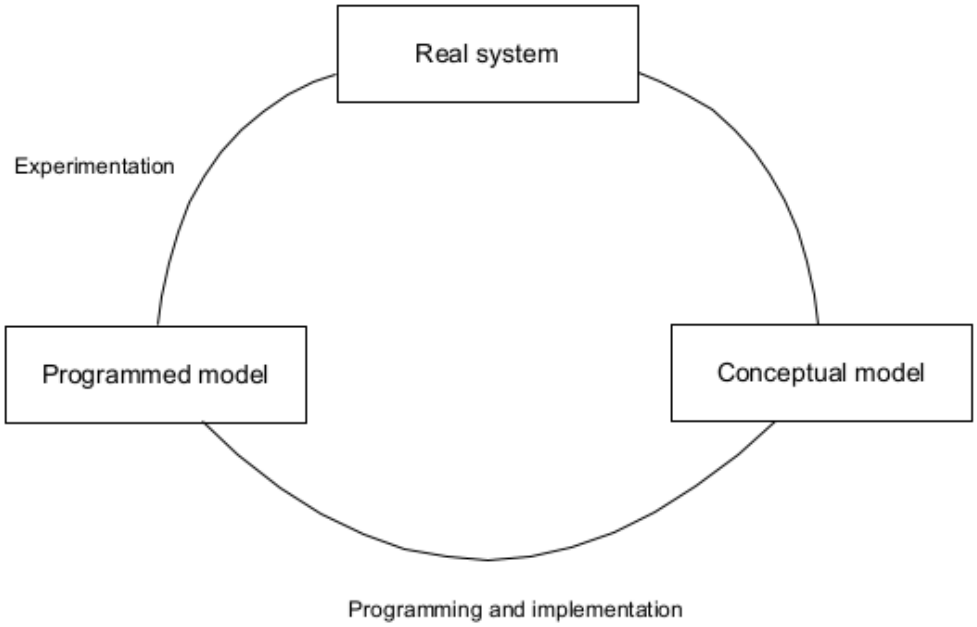

Figure 3.3: Simplified Modeling Process

3.2 CupCarbon Simulator

This section is devoted to define the simulator What we will be working on. Firstly, we give a short overview of the CupCarbon platform. Then, we talk about its architecture and presentation of the platform. We will end this section by presenting the 3D environment of cupcarbon.

Figure 3.4: Cupcarbon simulator

3.2.1 The CupCarbon platform

CupCarbon is a Smart City and Internet of Things Wireless Sensor Network (SCI-WSN) simulator .Its objective is to design, visualize, debug and validate distributed algorithms for monitoring, collecting environmental data, etc., and to create environmental scenarios,generally within educational and scientific projects.

CupCarbon offers two simulation environments. The first simulation environment enables the design of mobility scenarios and the generation of natural events such as fires and gas as well as the simulation of mobiles such as vehicles and flying objects (e.g. UAVs, insects, etc.). The second simulation environment represents a discrete event simulation of wireless sensor networks which takes into account the scenario designed on the basis of the first environment.

Networks can be designed and prototyped by an ergonomic and easy to use interface using the OpenStreetMap (OSM) framework to deploy sensors directly on the map. It includes a script language called SenScript , which allows to program and configure each sensor node individually.

CupCarbon offers the possibility to simulate algorithms and scenarios in several steps. The energy consumption can be calculated and displayed as a function of the simulation time. This allows to clarify the structure,feasibility and realistic implementation of a network before its real deployment[26]

3.2.2 The CupCarbon architecture:

CupCarbon is developed in Java. Its architecture consists of two layers: The first layer concerns the modules used to build the simulation. The second layer concerns the simulation it self. Figure 5 shows the different modules of CupCarbon[27]

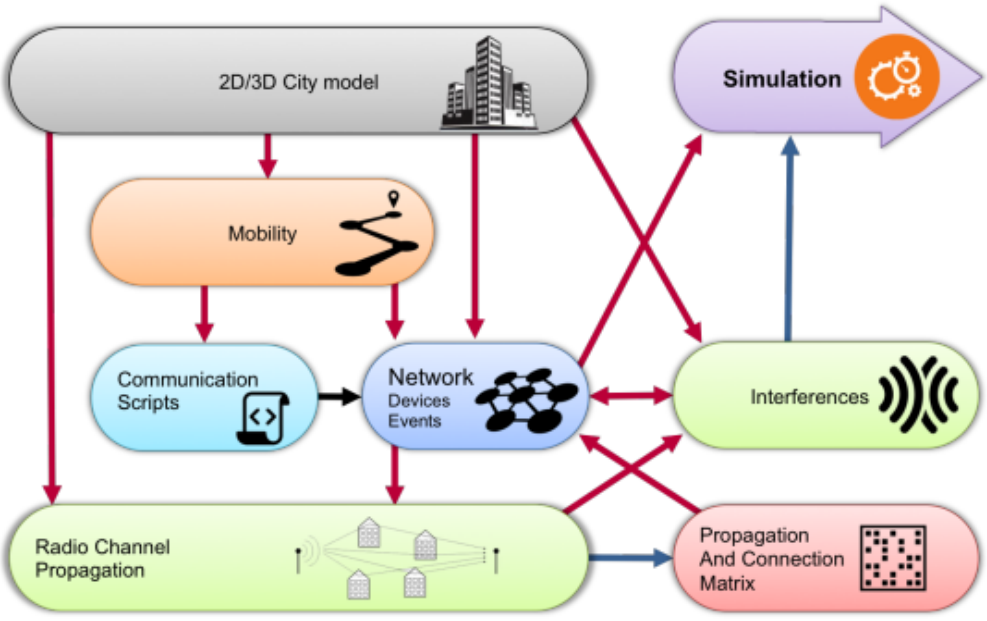

Figure 3.5: The CupCarbon architecture

3.1.3 The presentation of the platform:

The proposed platform is composed of 4 main parts: a 2D/3D environment block, an interference block, a radio channel block and an implementation block[28].

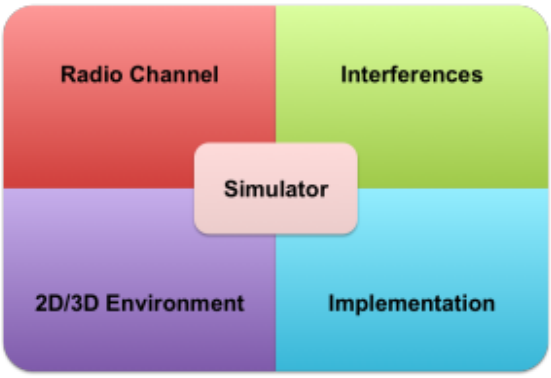

Figure 3.6: Main parts of the CupCarbon platform.

3.1.4 The 3D environment:

The 3D environment of CupCarbon is composed of ground elevation, buildings and various objects like sensor nodes. The ground elevation model can be imported into a CubCarbon project using heightmaps, which are simple grayscale images and meta data, providing GPS coordinates and bounds (min and max altitude) of the heightmaps. The ground elevation can also be obtained using external web services such as Google Elevation API.Once the elevation is imported, a bilinear interpolation is performed to compute a triangle mesh. Top shapes of buildings can be imported from the Overpass API provided by OSM. Then these shapes will be extruded on the ground elevation. Sensor nodes are simply represented by spheres of various colors and sizes, depending on their respective states and types. Links between sensors are shown by lines. Ground, buildings and sensor node meshes are then sent to the graphic card using the OpenGL API. Since CupCarbon is programmed in Java,we have used the JOGL (Java OpenGL) library in order to create and use the OpenGL context[33].

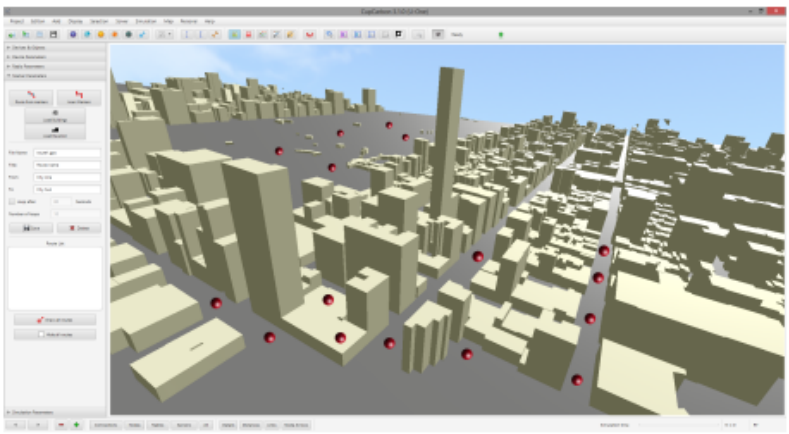

Figure 3.7: 3D environment in CupCarbon.

3.2.3 Objective of cupcarbon

Its objective is to design, visualize, debug and validate distributed algorithms dedicated to monitoring, data collection, and to create environmental scenarios such as fires, gas and mobiles.

It allows to dynamically configure the different nodes (networks with several channels, identifiers, etc.)[36]

3.2.4 SenScript

SenScript is an interpreted language used especially to program nodes of a wireless sensor network[36]

- Variables are not declared
- A variable is declared when it is used for the first time
- A variable can be initialized using the set command
- Instructions: command [result] [option] [args]

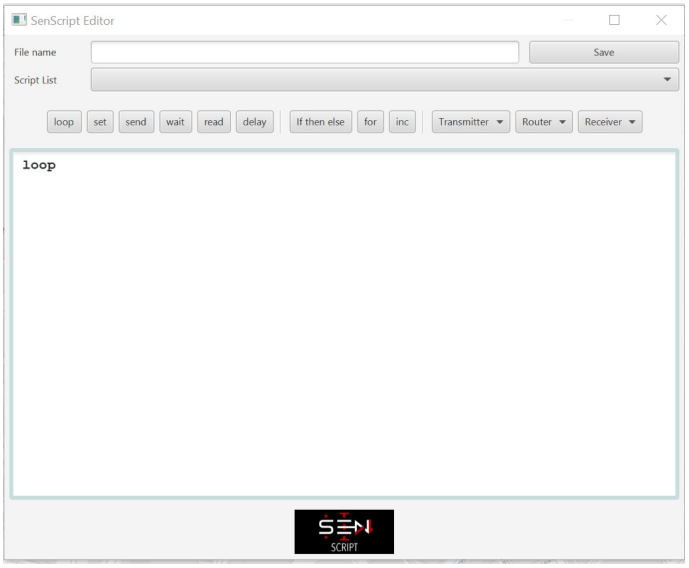

Figure 3.8: Senscript

3.2.5 Wireless sensor network

2.5.1 Sensor Node

A sensor node is a device that possesses the capacity to gather sensor information from the environment, process the information and communicate with other nodes [30].

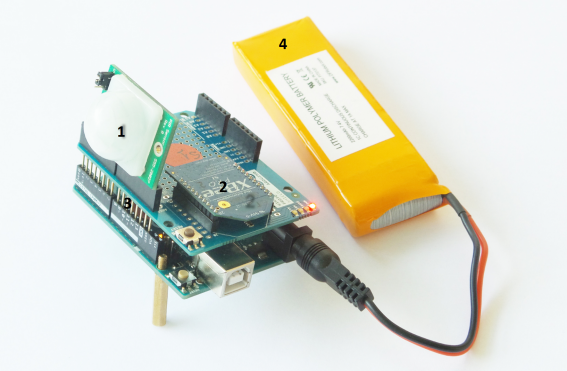

Figure 3.9: Sensor Node

1. Sensor (Capture unit)
2. Radio module (communication)
3. Microcontroller ß Script (program)
4. Battery

Options:

5. GPS
6. Mobility (Robot, chassis, . . .)

2.5.2 Wireless Sensor Network

Sensor network based on Arduino/Xbee modules

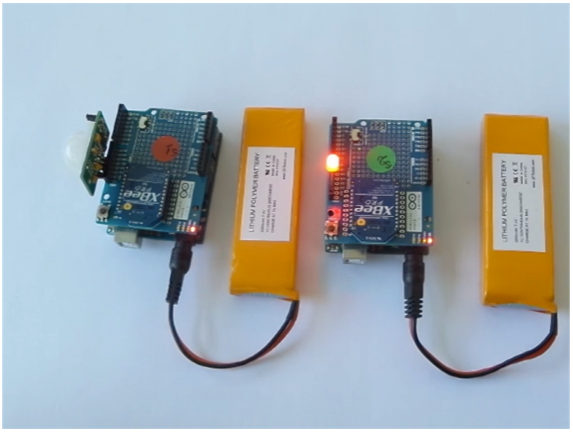

Figure 3.10: ArduinoXbee

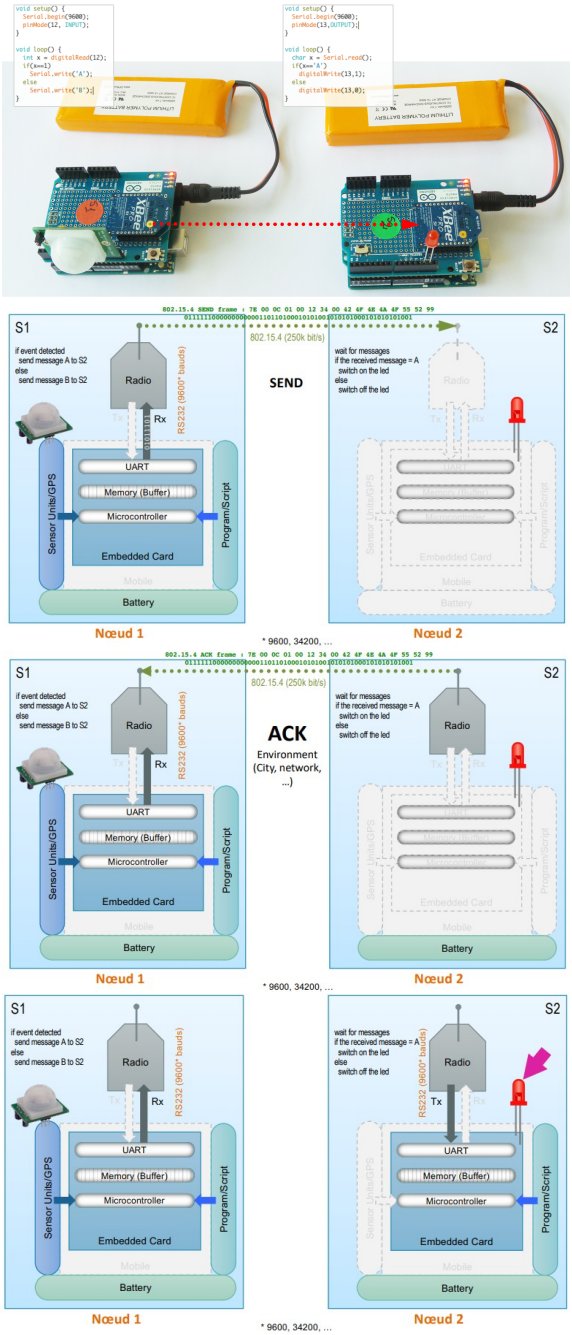

Figure 3.11: Wireless Sensor Network

3.3 Problem Modeling

The main issue of this section is to study the problem of domination in smart cities using modeling and simulation technique, basing on the domination algorithm which was presented in the first chapter.

3.3.1 System model

In this step, we will present our conceptual model. So, we consider the topology of a Smart City as an undirected connected graph G = (V,E) where V is a set of nodes (sonsors) and E is the set of bidirectional asynchronous communication links(wifi,zegbee.) .

We state that n is the size of $\left(G(|\ V\ |=n)\right)$ and m is the number of edges $\left(|\ E\ |=m\right)$. We assume that the graph G = (V,E) is a simple connected graph. In the system, u and v are neighbors if and only if a communication link (u,v) exists (i.e., $\left((u,v)\in E\right)$. Each node $\left(v\in V\right)$ has a unique identifier in the network, noted IDv.

3.3.2 Distributed construction

In this step, we will present our programmed model. So, we will construct a distributed version of the graph algorithm for dominating set in smart cities that will run on each node of the system.

In this algorithm we select the nodes those have the maximum degree of neighbors and then we calculate the initial waiting time, where the node that has the largest number of neighbors we put it in the dominating set and we inform its neighbors to be removed from the initial list.

Algorithm2 Distributed algorithm for Dominating set graphs

```
1: input: G (V, E), V
2: output: dominant
3: id = getId ()
4: n = getNeig
5: b = rand (1,100)
6: w=(50-n) *100+b
7: dominant= false
8: while(true) do
9:    wait w
10:   read message
11:    if (message== null) then
12:    send(T1)
13:    dominant = true
14:    else
15:       if (message == T1) then
16:         send w
17:       end
18:       if (message == T2) then
19:         send
20:       end
21:    end
22:    message == T2
30:  end while
```

We have to note that the algorithm is uniform, a distributed algorithm is uniform if and only if all nodes execute the same algorithm.

Conclusion

Simulation is now widely recognized as a powerful technique for system analysis and design. It can be applied in various fields such as computer systems.In this chapter, we introduced the basic concepts required for better understanding of simulation. Then, we presented the cupcarbon simulator platform,Finally we presented our conceptual and programming model .

Chapter 4

Experimental results

Introduction

In this chapter, We will start by explaining why we chose Smart city as the environment for our project. After that, we will present our experimentation .Moreover,We will make a simple comparison between our work and the work of last year.

4.1 Choice of Smart cities

In our work, we chose the smart city because it contains many facilities that are always connected to the Internet (sensors), and therefore it is vulnerable to hacking frequently, so it must be protected from piracy and all its information is secured.

Figure 4.1: Choice of Smart cities

4.1.1 Mila city presentation

Mila is located in the north-east of Algeria and the main city is Mila. The city center contains several facilities, including the municipality, hospitals, pharmacies, commercial centers, cafes ... etc.

In order to make Mila a smart city, we have placed a set of sensors at the level of several prominent facilities, where the latter communicate with each other via a wireless network and transmit information freely and to protect the information from being hacked using the firewall.

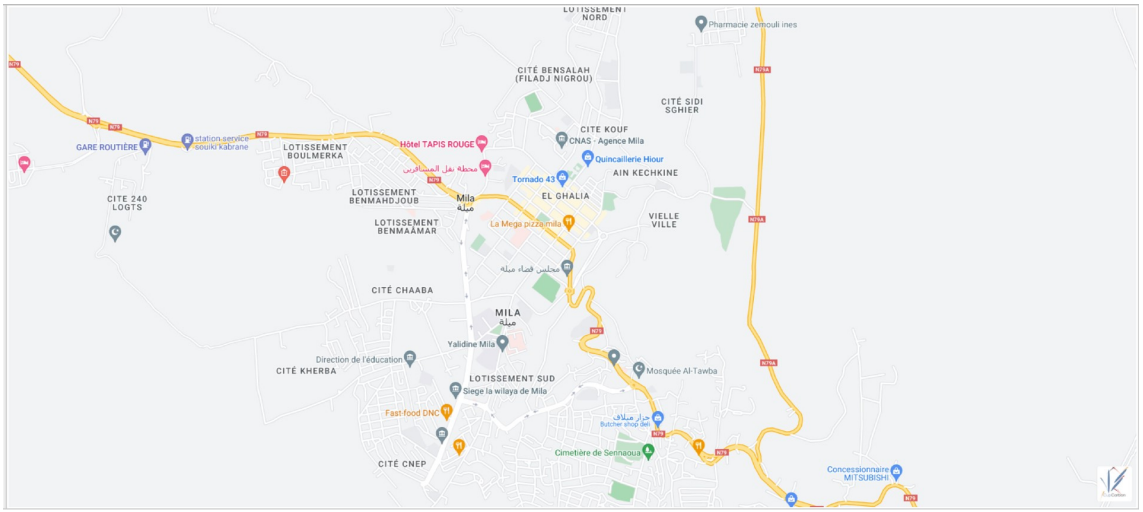

Figure 4.2: Mila Map

4.2 Experimentation

To test the proposed algorithm we used an Intel I5 (2.50GHZ) computer with 8 G of RAM under the Windows 10 professional operating system.

The detail of our experimental results can be found in the following stapes.

4.2.1 Traces for experimentation

1.1 Centralized construction

• Step 1:We choose one node that have the maximum degree in the graph (11) ,which the number of its neighbors equals 4 ,and put it in the list of the dominating set (marking).

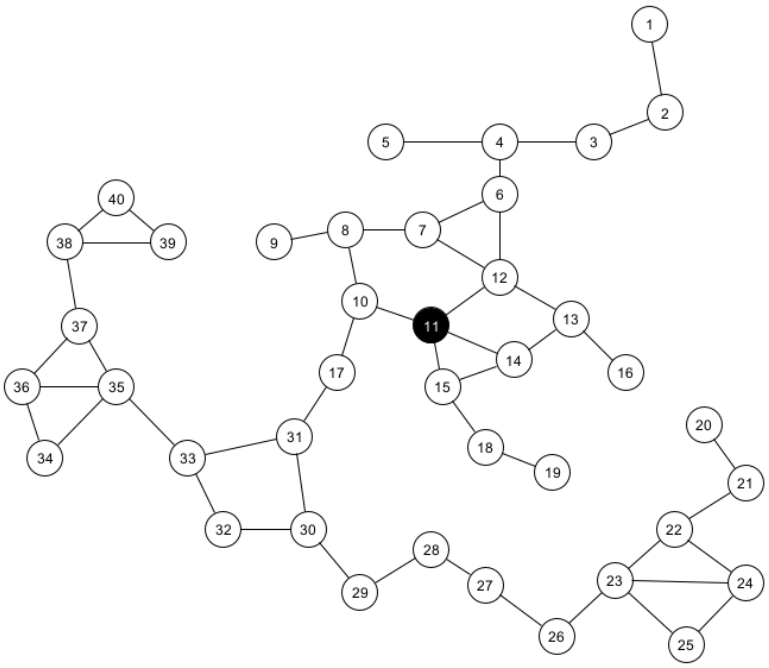

Figure 4.3: Step 1

• Step 2: Remove its neighbors from the list of original nodes of the graph.

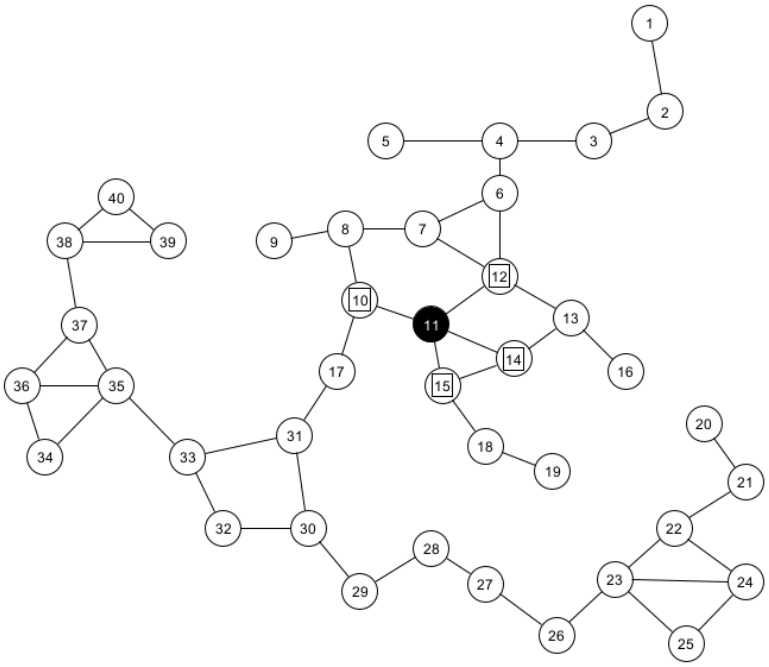

Figure 4.4: Step 2

• Step 3:We do the same work on the new graph, by choosing one node that have the maximum degree (23) , which the number of its neighbors equals 4 ,and put it in the list of the dominating set (marking).

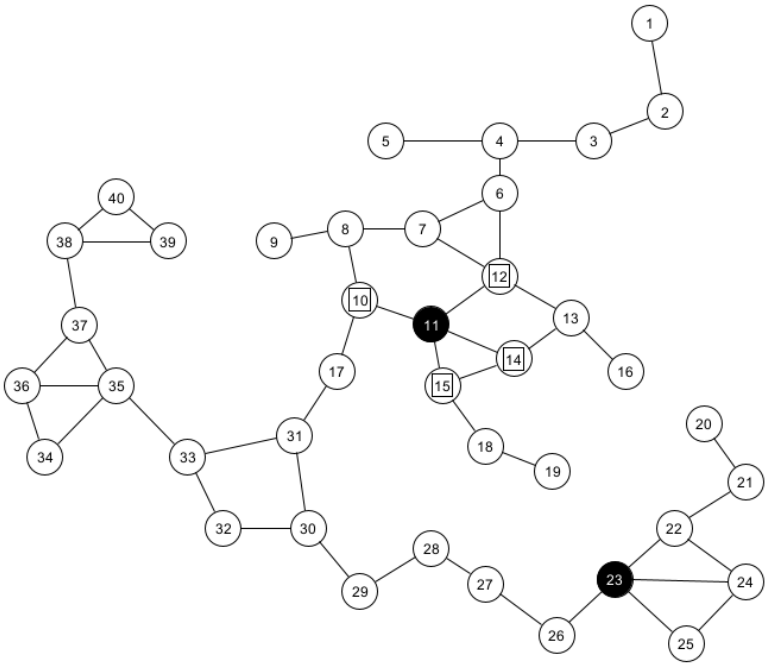

Figure 4.5: Step 3

• Step 4:Remove its neighbors from the list of original nodes of the graph.

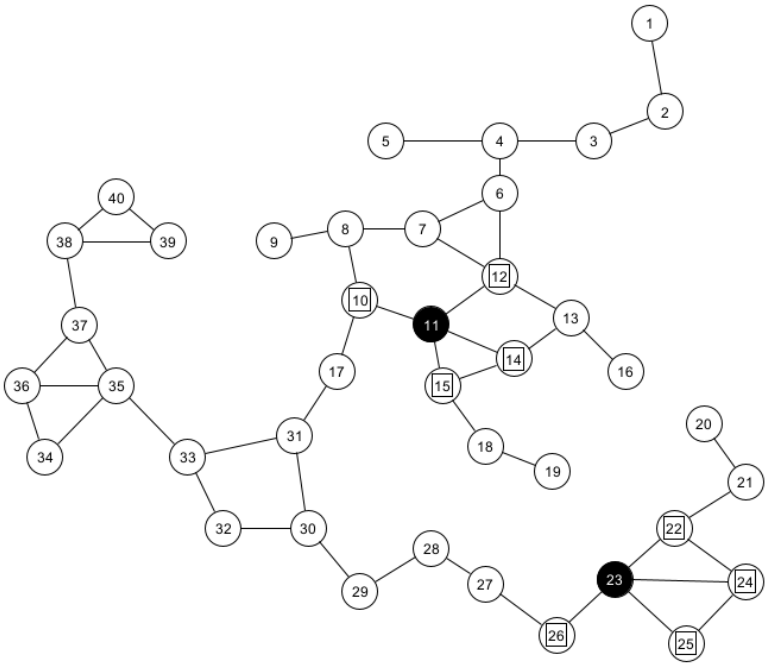

Figure 4.6: Step 4

• Step 5:We do the same work on the new graph, by choosing one node that have the maximum degree (35) , which the number of its neighbors equals 4 ,and put it in the list of the dominating set (marking).

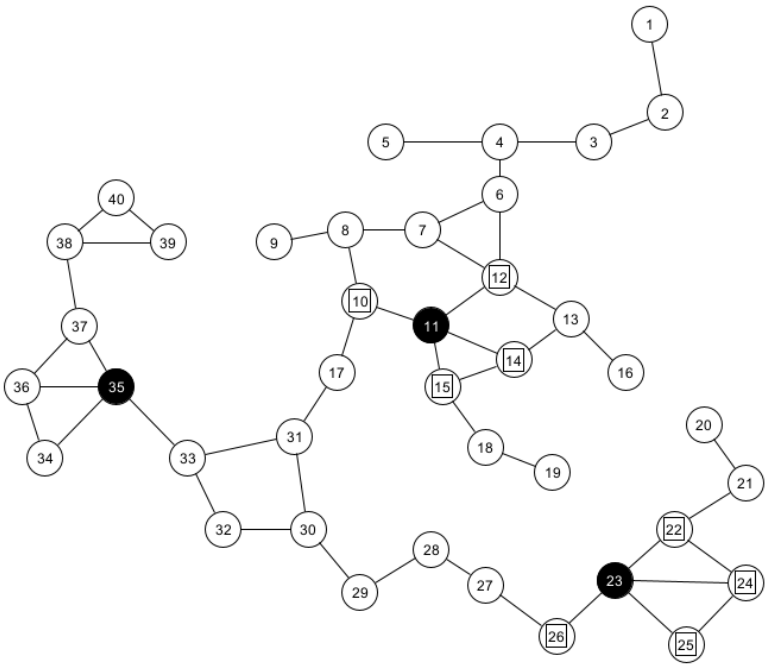

Figure 4.7: Step 5

• Step 6:Remove its neighbors from the list of original nodes of the graph.

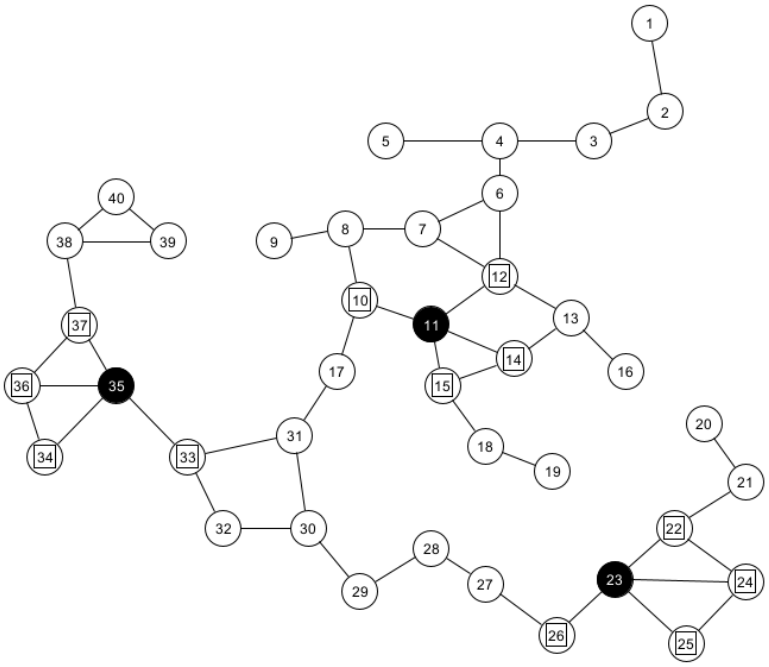

Figure 4.8: Step 6

• Step 7:We do the same work on the new graph, by choosing one node that have the maximum degree (4) , which the number of its neighbors equals 3 ,and put it in the list of the dominating set (marking).

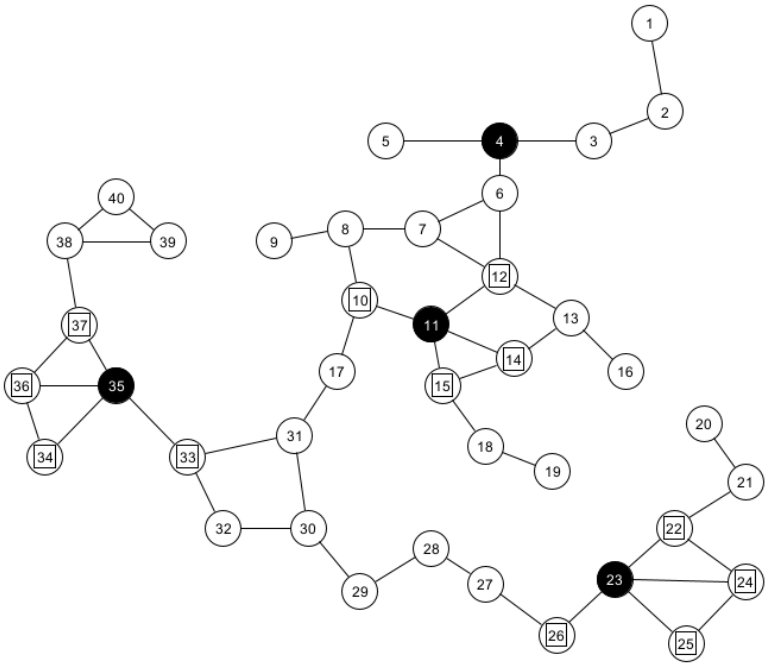

Figure 4.9: Step 7

• Step 8:Remove its neighbors from the list of original nodes of the graph.

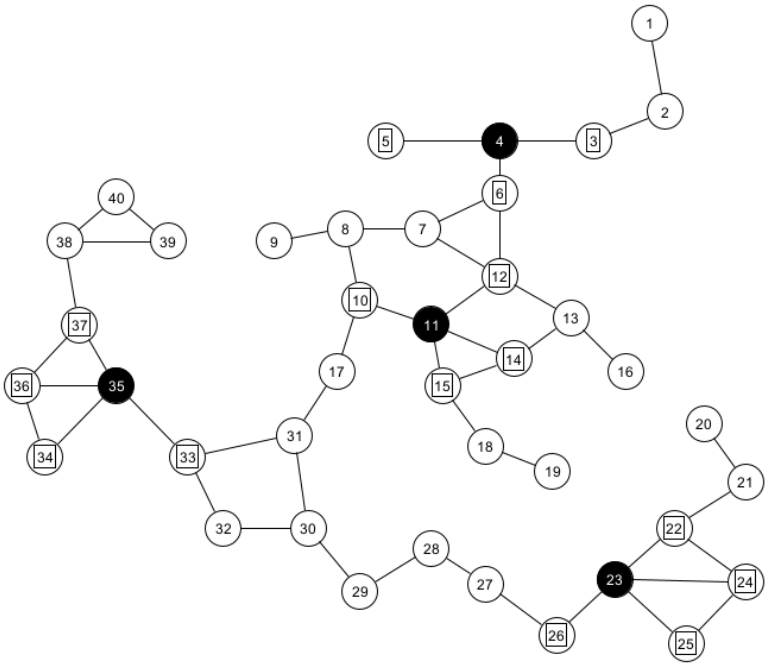

Figure 4.10: Step 8

• Step 9:We do the same work on the new graph, by choosing one node that have the maximum degree (7) , which the number of its neighbors equals 3 ,and put it in the list of the dominating set (marking).

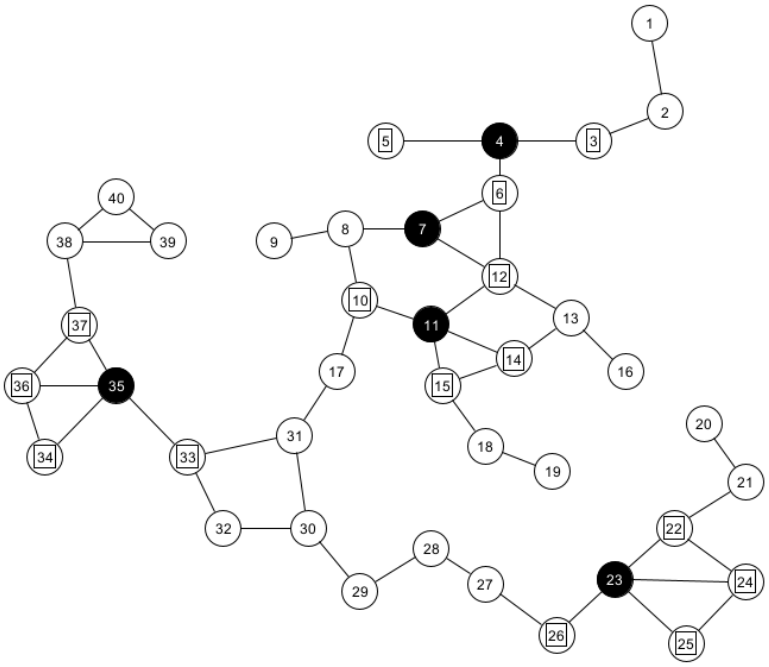

Figure 4.11: Step 9

• Step 10:Remove its neighbors from the list of original nodes of the graph.

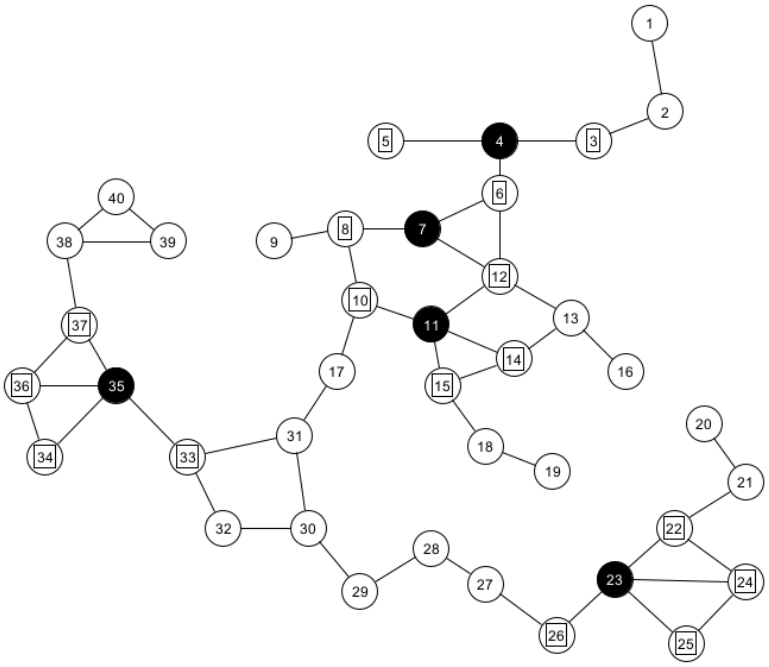

Figure 4.12: Step 10

• Step 11:We do the same work on the new graph, by choosing one node that have the maximum degree (30) , which the number of its neighbors equals 3 ,and put it in the list of the dominating set (marking) .

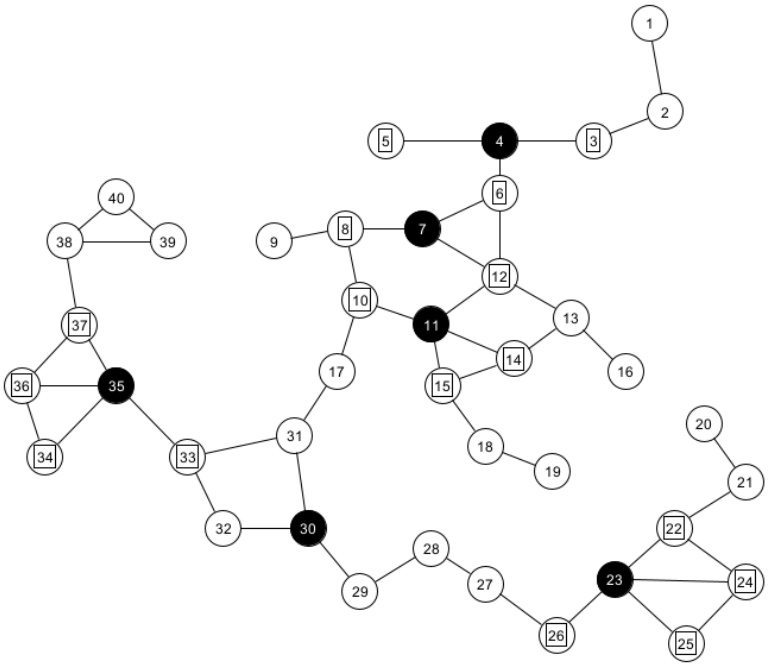

Figure 4.13: Step 11

• Step 12:Remove its neighbors from the list of original nodes of the graph.

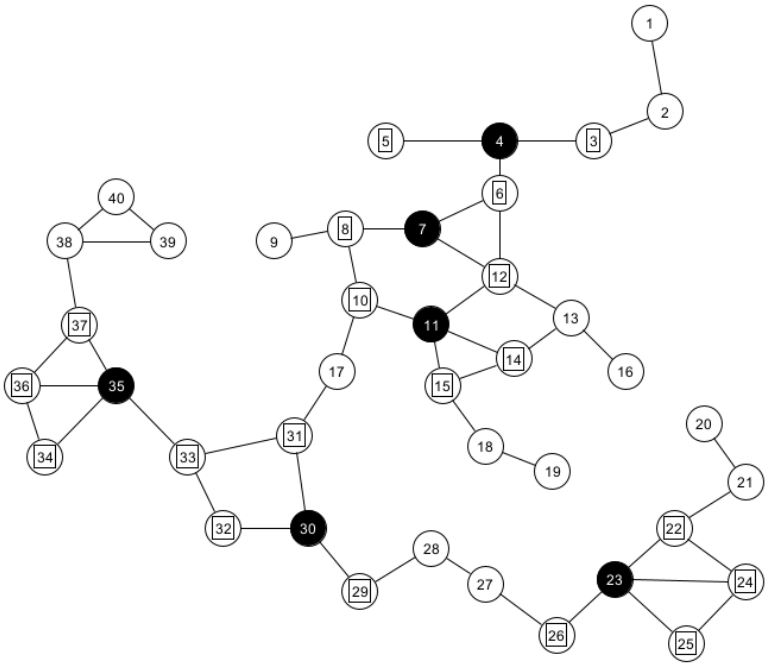

Figure 4.14: Step 12

• Step 13:We do the same work on the new graph, by choosing one node that have the maximum degree (13) , which the number of its neighbors equals 3 ,and put it in the list of the dominating set (marking) .

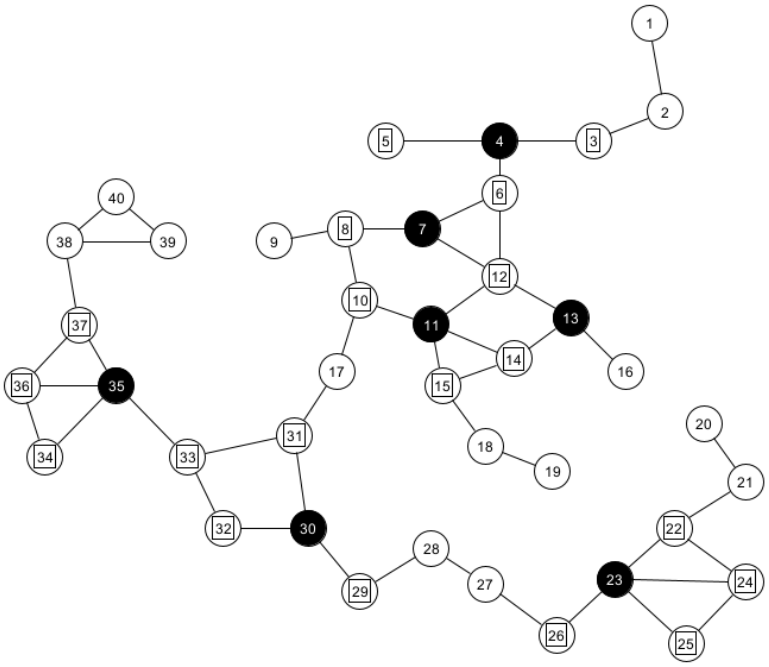

Figure 4.15: Step 13

• Step 14:Remove its neighbors from the list of original nodes of the graph.

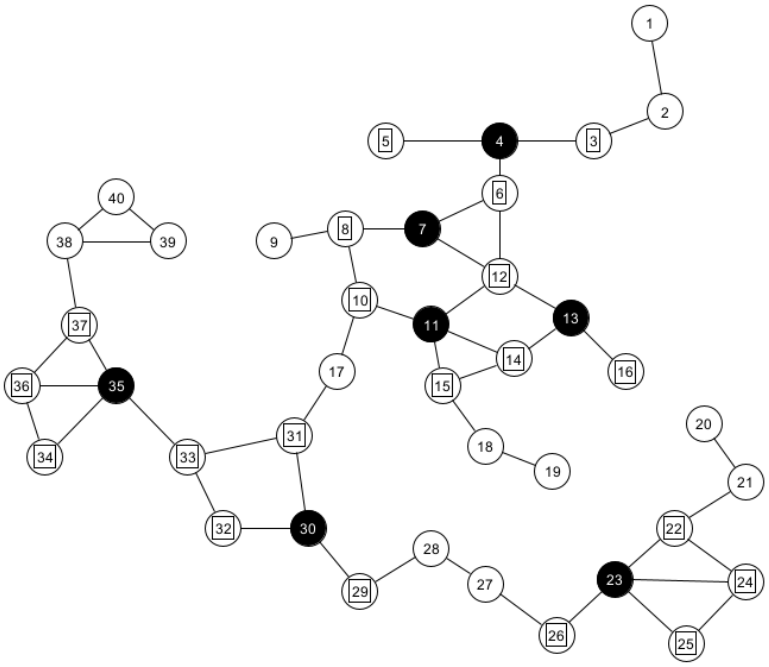

Figure 4.16: Step 14

• Step 15: We do the same work on the new graph, by choosing one node that have the maximum degree (38) , which the number of its neighbors equals 3 ,and put it in the list of the dominating set (marking) .

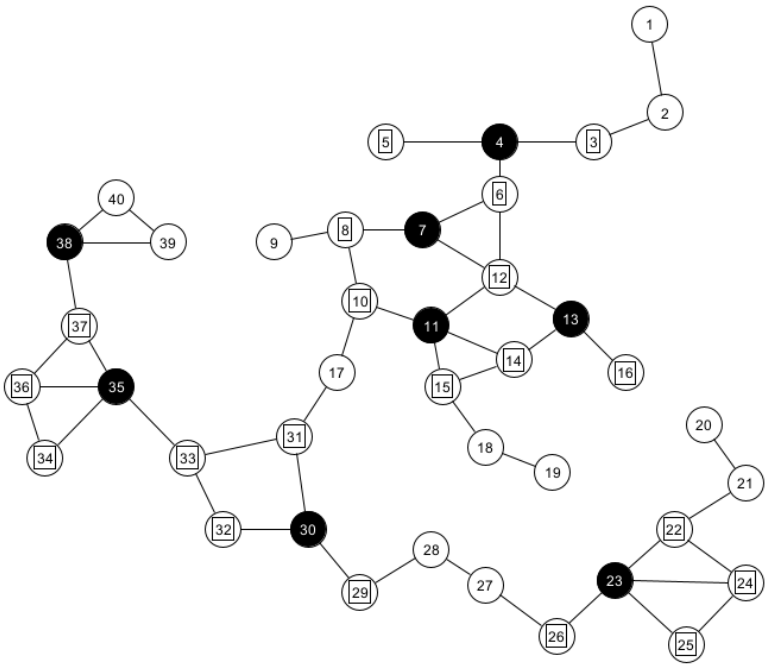

Figure 4.17: Step 15

• Step 16:Remove its neighbors from the list of original nodes of the graph.

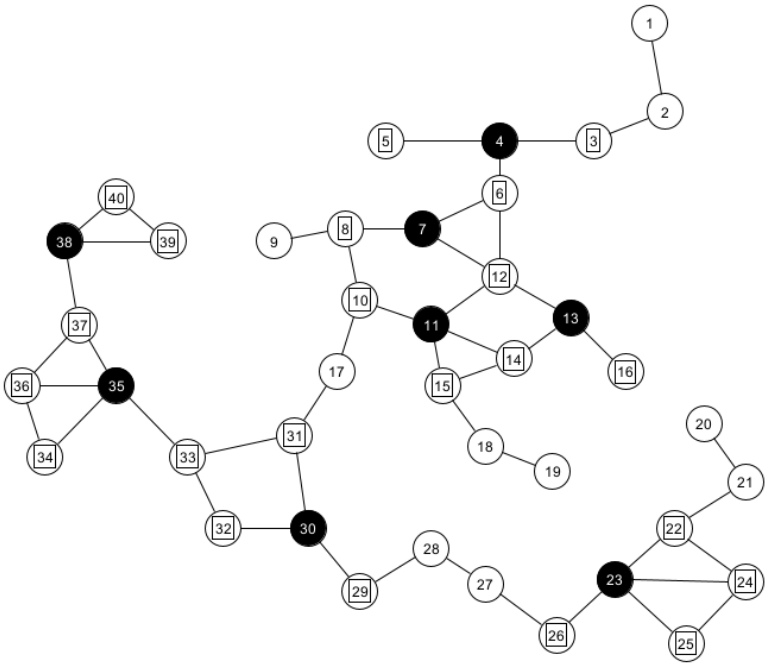

Figure 4.18: Step 16

• Step 17: We do the same work on the new graph, by choosing one node that have the maximum degree (2) , which the number of its neighbors equals 2 ,and put it in the list of the dominating set (marking) .

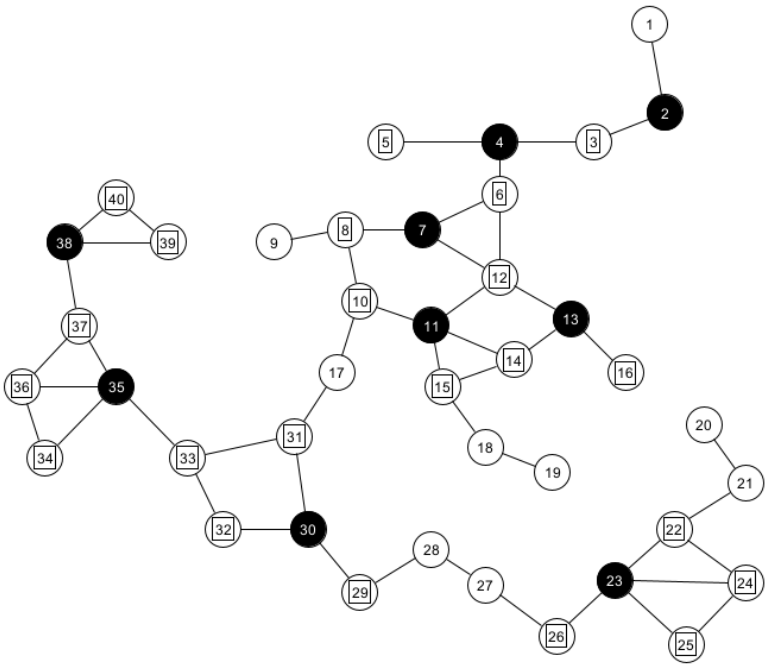

Figure 4.19: Step 17

• Step 18:Remove its neighbors from the list of original nodes of the graph.

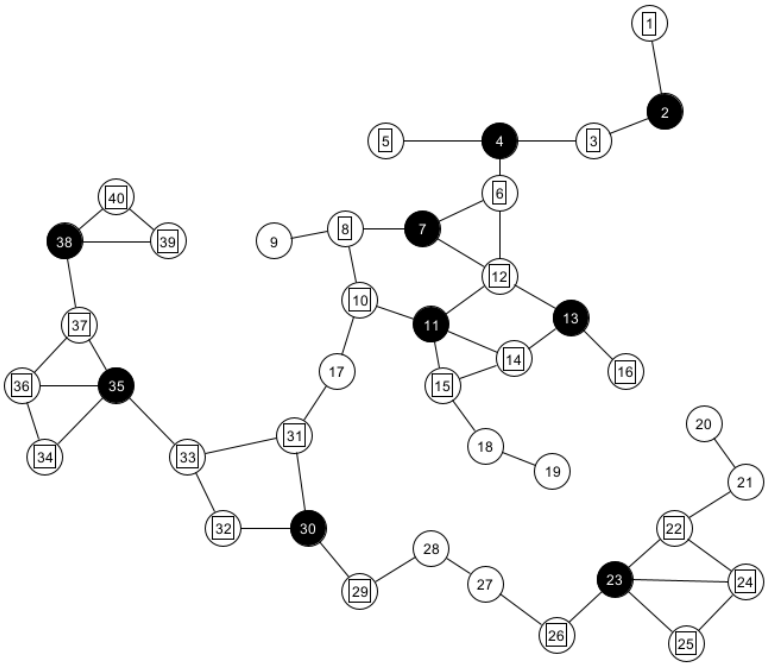

Figure 4.20: Step 18

• Step 19: We do the same work on the new graph, by choosing one node that have the maximum degree (17) , which the number of its neighbors equals 2 ,and put it in the list of the dominating set (marking) .

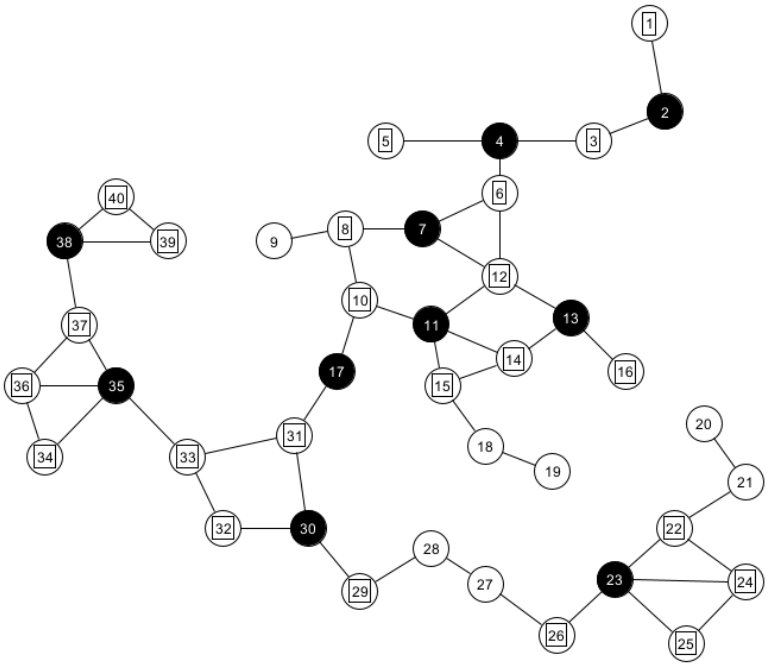

Figure 4.21: Step 19

• Step 20:We do the same work on the new graph, by choosing one node that have the maximum degree (18) , which the number of its neighbors equals 2 ,and put it in the list of the dominating set (marking) .

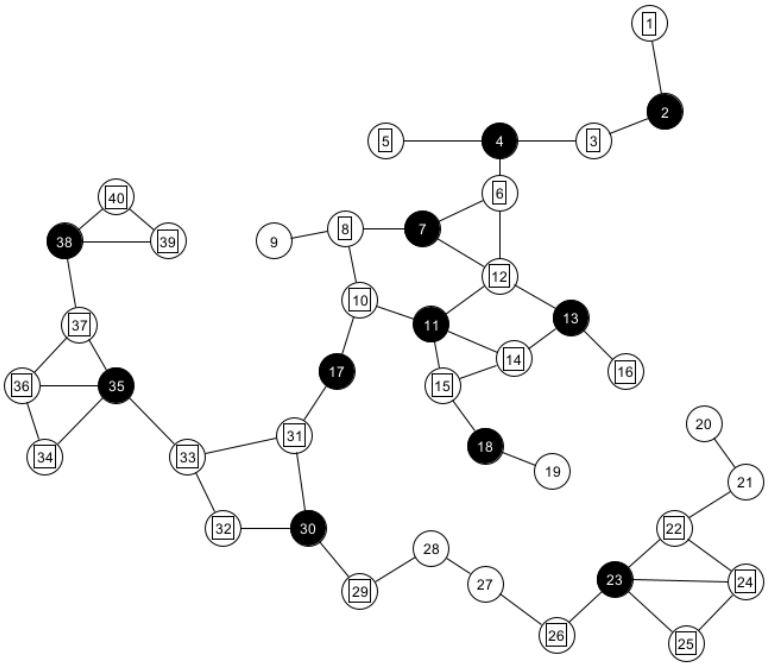

Figure 4.22: Step 20

• Step 21:Remove its neighbors from the list of original nodes of the graph.

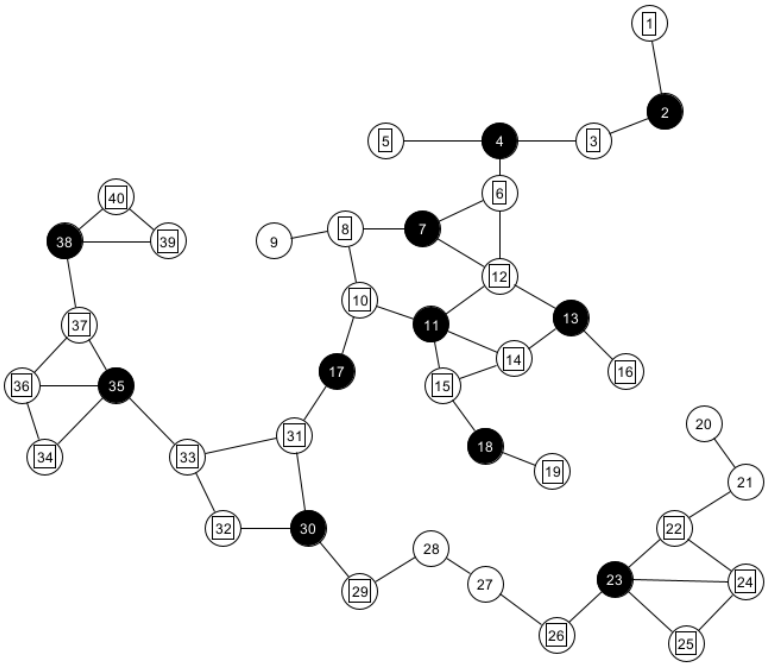

Figure 4.23: Step 21

• Step 22:We do the same work on the new graph, by choosing one node that have the maximum degree (21) , which the number of its neighbors equals 2 ,and put it in the list of the dominating set (marking) .

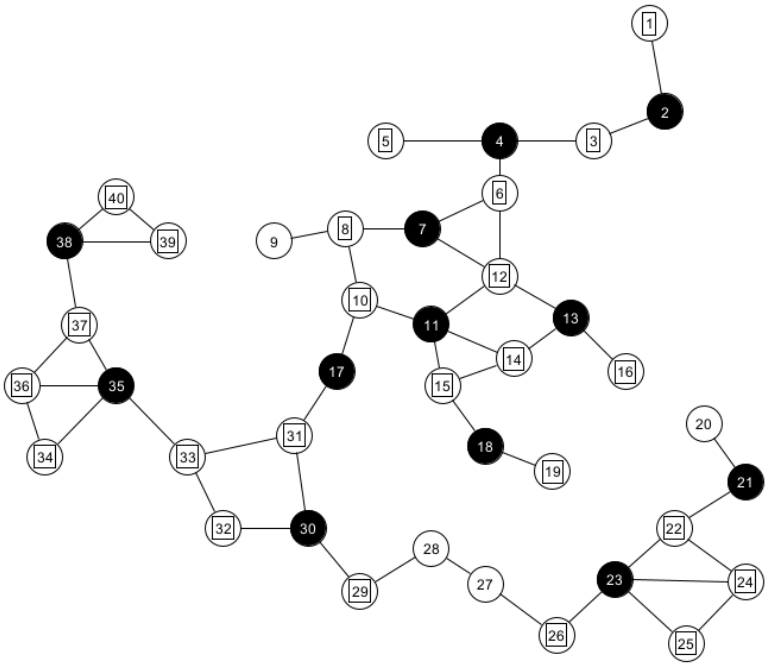

Figure 4.24: Step 22

• Step 23:Remove its neighbors from the list of original nodes of the graph.

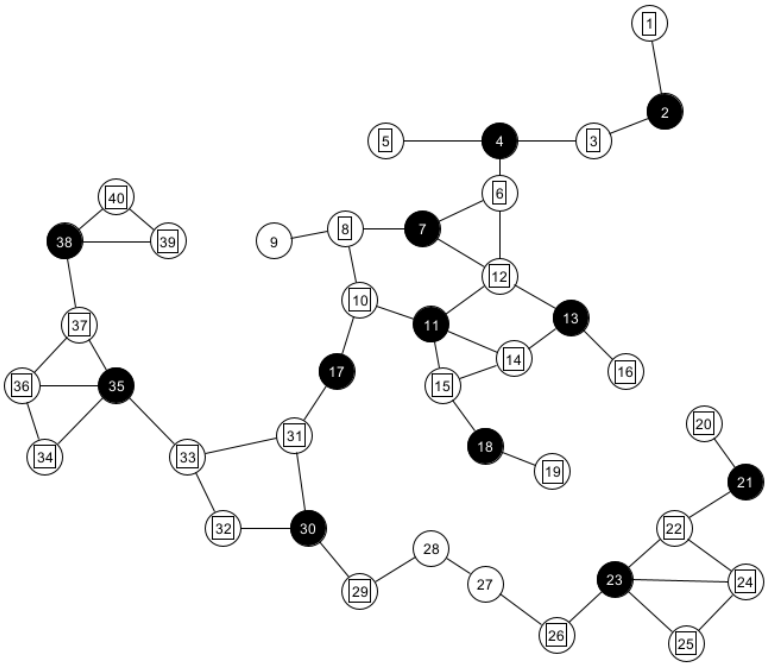

Figure 4.25: Step 23

• Step 24:We do the same work on the new graph, by choosing one node that have the maximum degree (28) , which the number of its neighbors equals 2 ,and put it in the list of the dominating set (marking) .

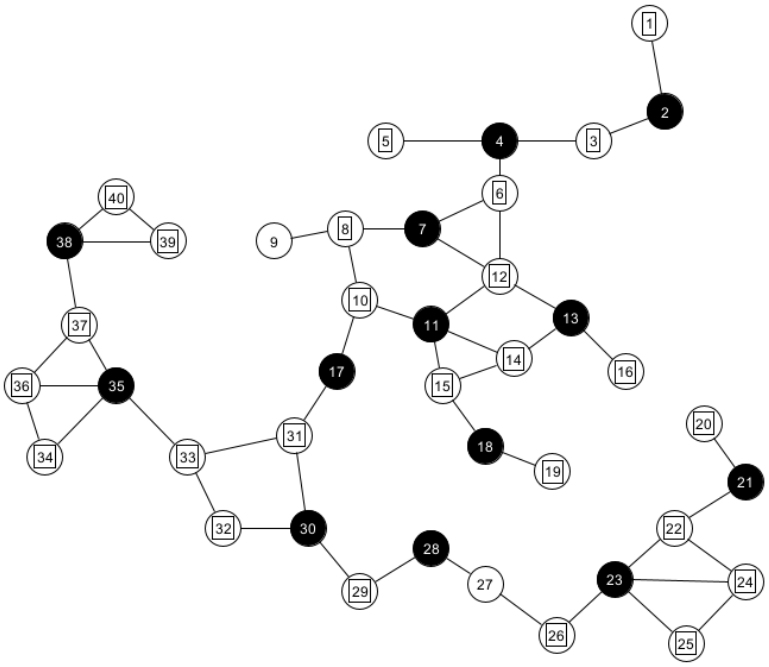

Figure 4.26: Step 24

• Step 25:Remove its neighbors from the list of original nodes of the graph.

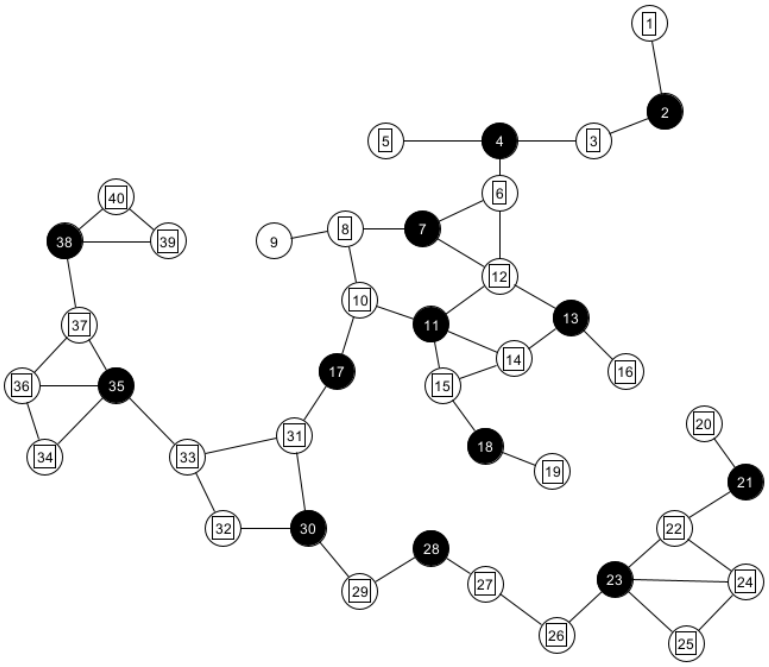

Figure 4.27: Step 25

• Step 26:We do the same work on the new graph, by choosing one node that have the maximum degree (9) , which the number of its neighbors equals 1 ,and put it in the list of the dominating set (marking) .

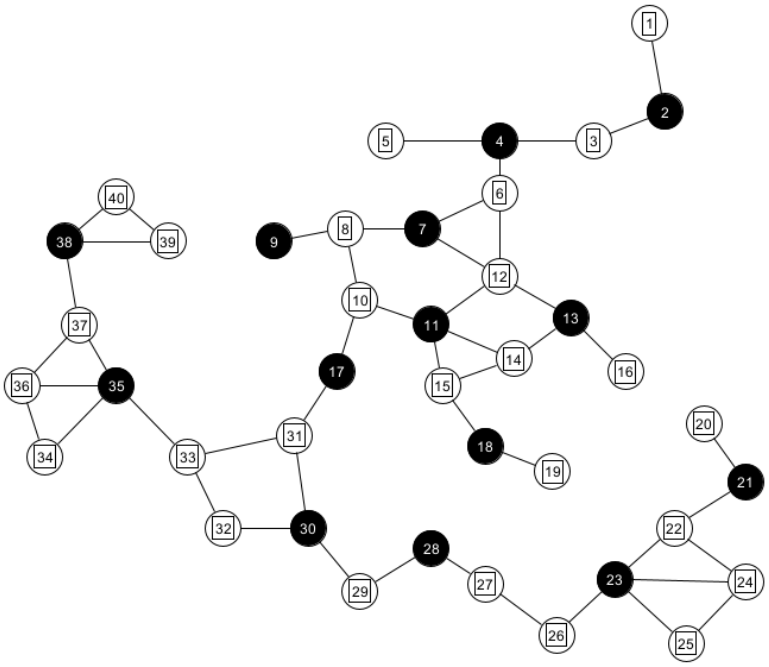

Figure 4.28: Step 26

1.2 Distributed construction

• Step 1:We choose the nodes those have the maximum degree in the graph (11,12,23 and 25) in which the number of their neighbors equals 4.After that , we put them in the list of the dominating set (marking).

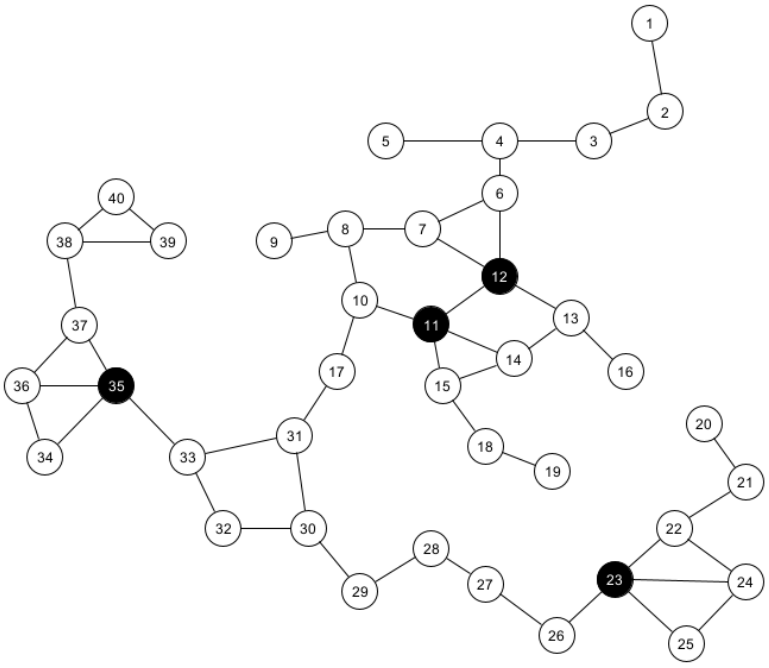

Figure 4.29: Step 1

• Step 2:The marking nodes send a message to their neighbors in order to remove them from the nodes original list of the graph at the same time (parallel).

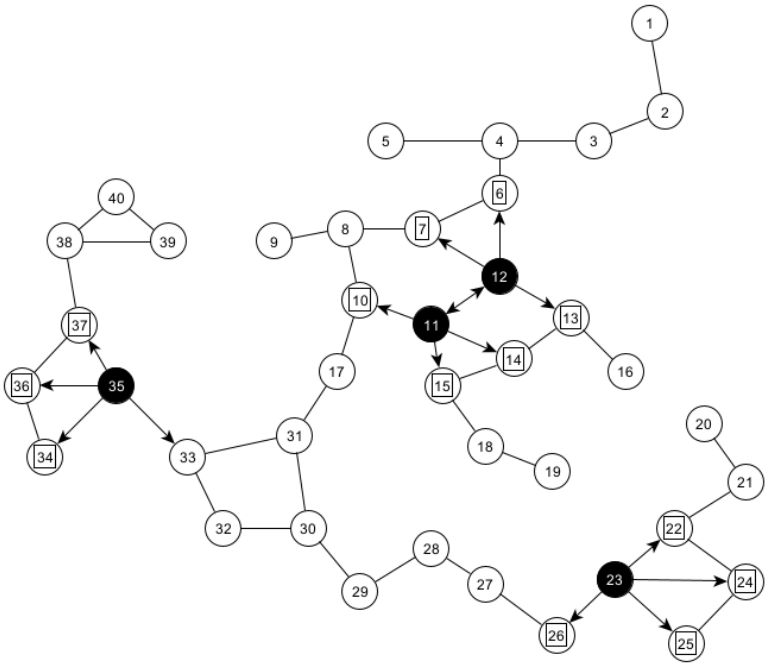

Figure 4.30: Step 2

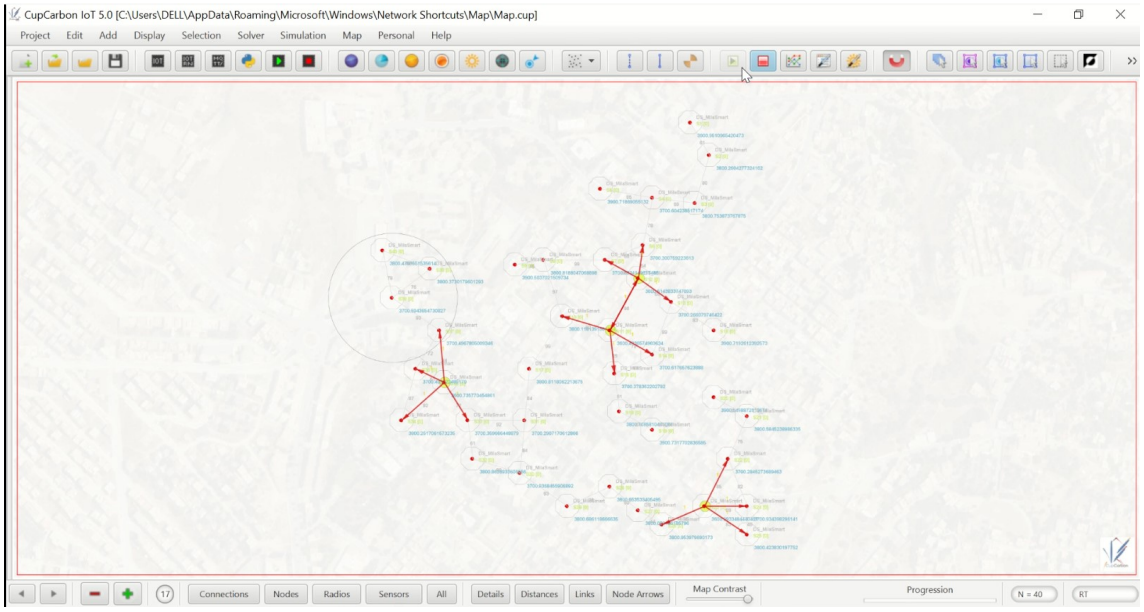

Figure 4.31: Step 2 in cupcarbon

• Step 3:By using the same way in the step 1 on a new graph , by choosing the nodes those have the maximum degree in the graph (38,4,31 and 30) in which the number of their neighbors equals 3.After that , we put them in the list of the dominating set (marking).

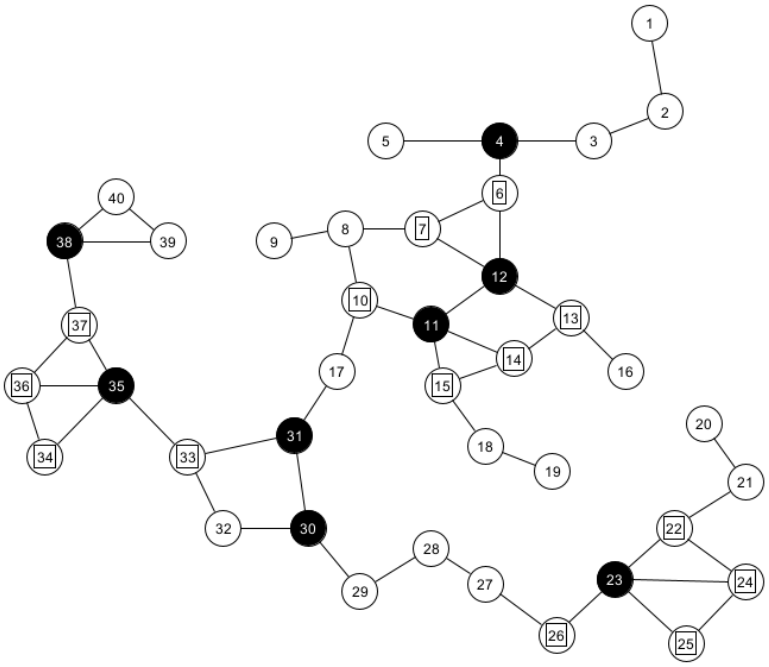

Figure 4.32: Step 3

• Step 4:The marking nodes send a message to their neighbors in order to remove them from the nodes original list of the graph at the same time (parallel).

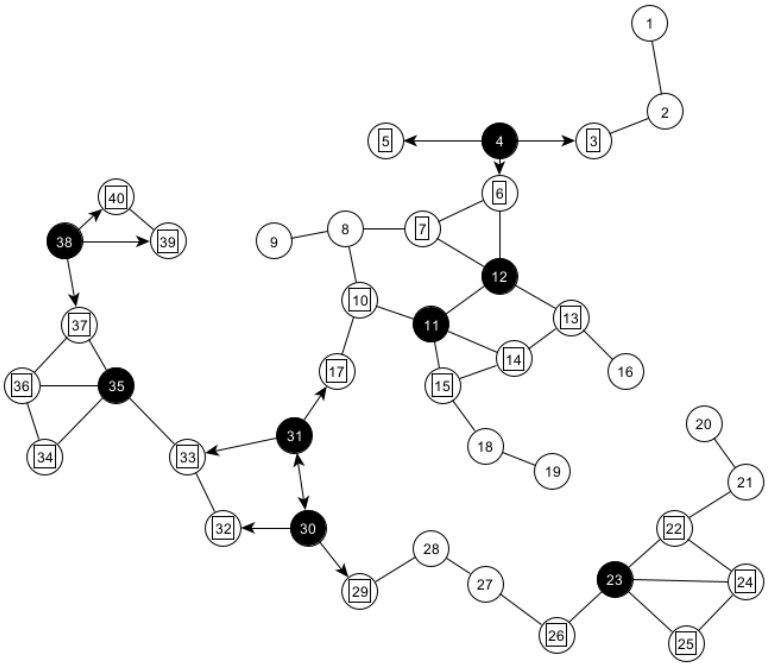

Figure 4.33: Step 4

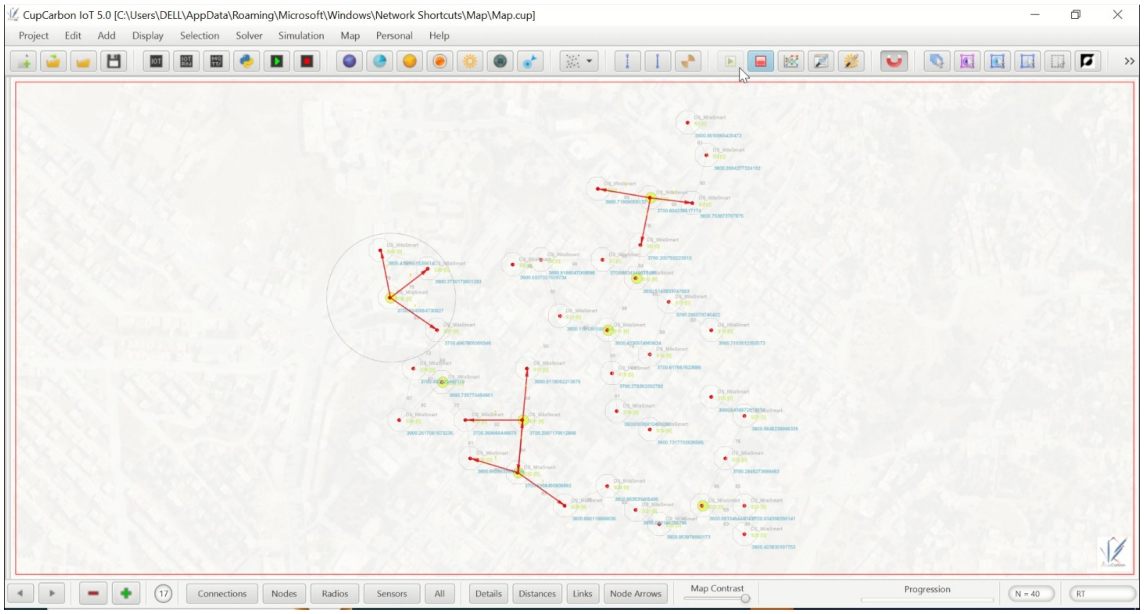

Figure 4.34: Step 4 in cupcarbon

• Step 5:By adopting the same method in the step 1 and 3 on a new graph , by choosing the nodes those have the maximum degree in the graph (2,8,18,28,21,and 27) in which the number of their neighbors equals 2.Then , we put them in the list of the dominating set (marking).

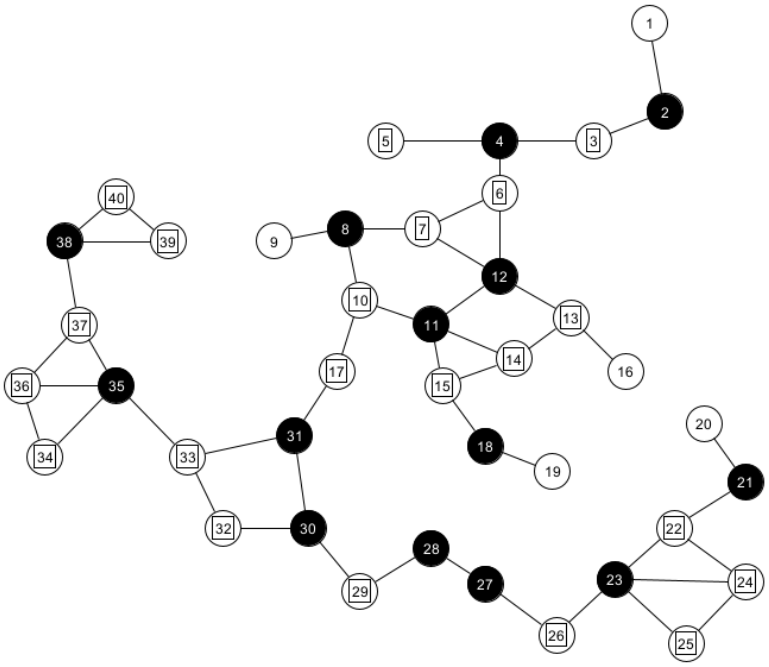

Figure 4.35: Step 5

• Step 6:The marking nodes send a message to their neighbors in order to remove them from the nodes original list of the graph at the same time (parallel).

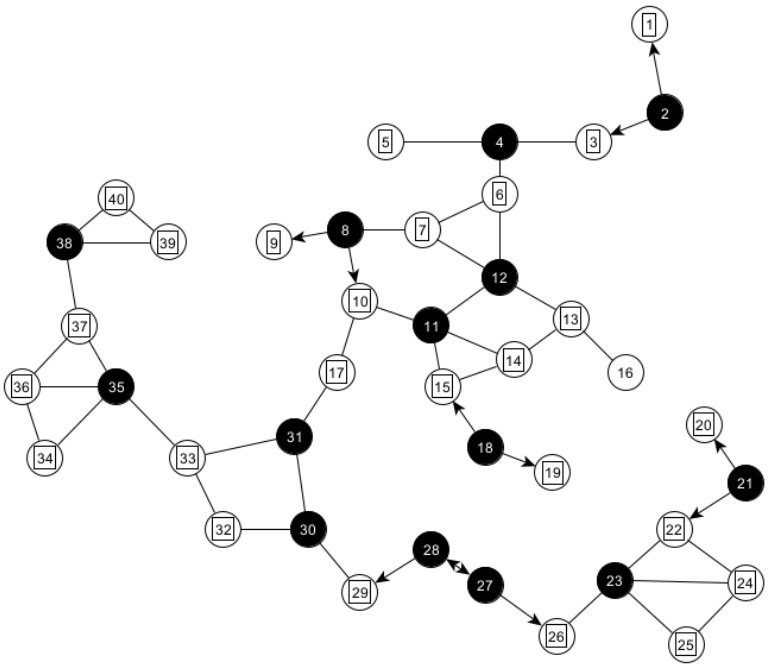

Figure 4.36: Step 6

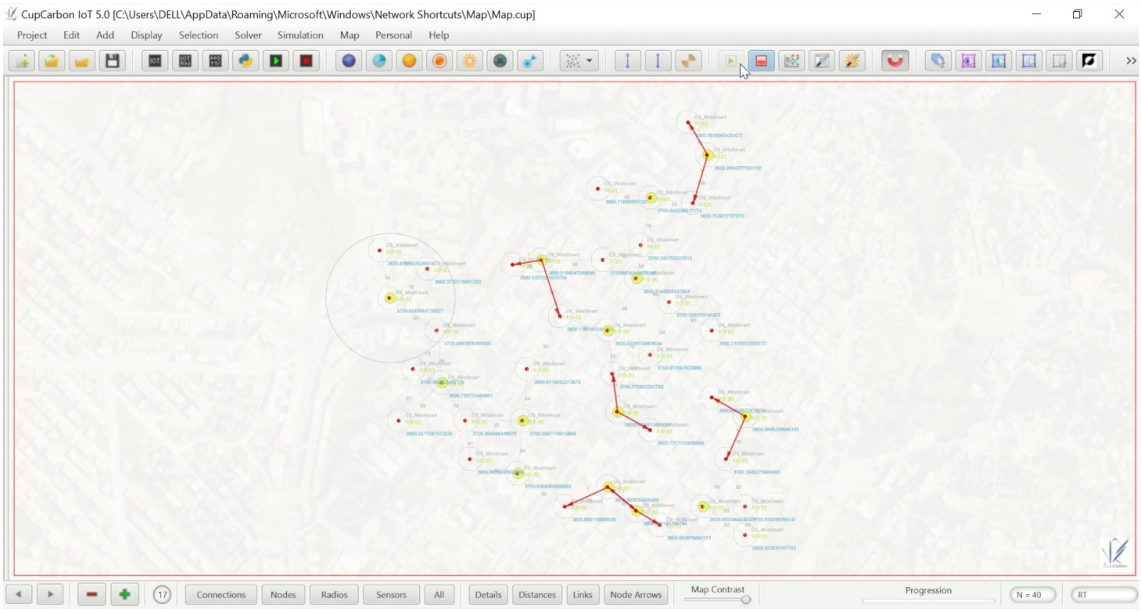

Figure 4.37: Step 6 in cupcarbon

• Step 7:By adopting the same method in the step 1,3 and 5 on a new graph , by choosing the nodes those have the maximum degree in the graph (16) in which the number of their neighbors equals 1.Then , we put them in the list of the dominating set (marking).

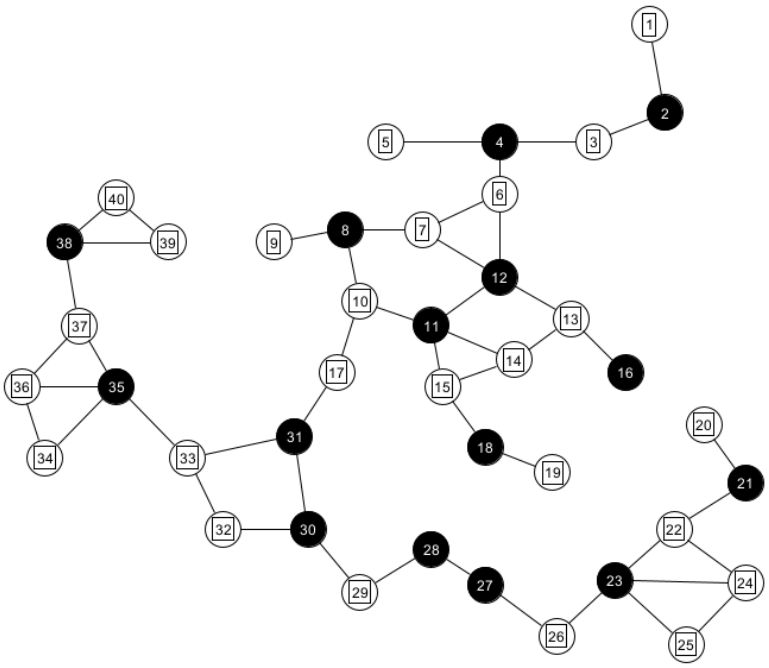

Figure 4.38: Step 7

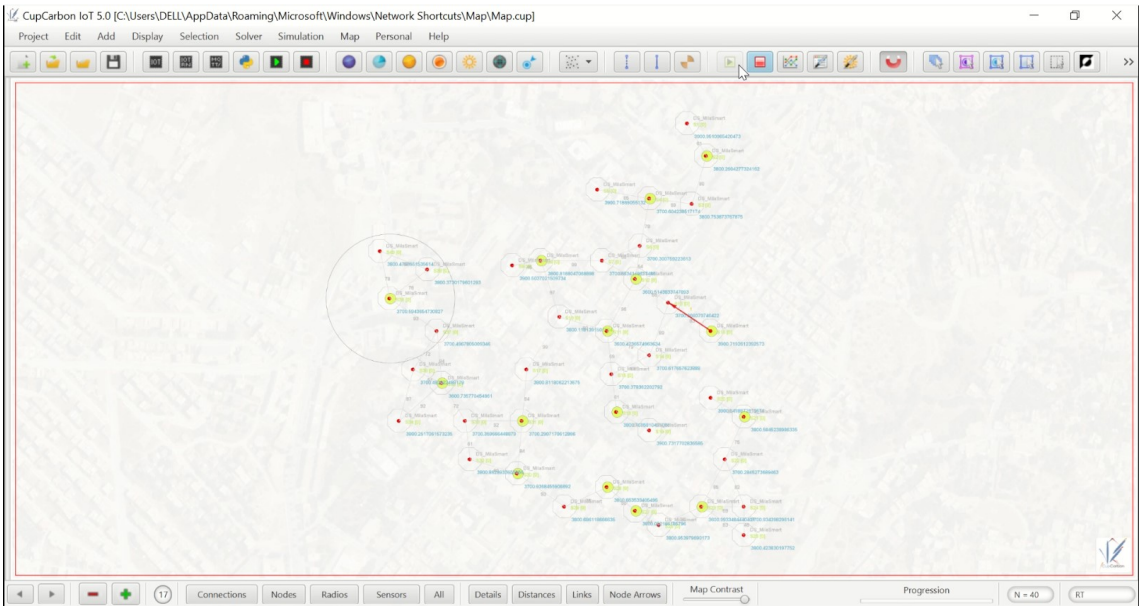

Figure 4.39: Step 7 in cupcarbon

4.2.2 Before simulation

The figure below represent the locations of the nodes in mila map

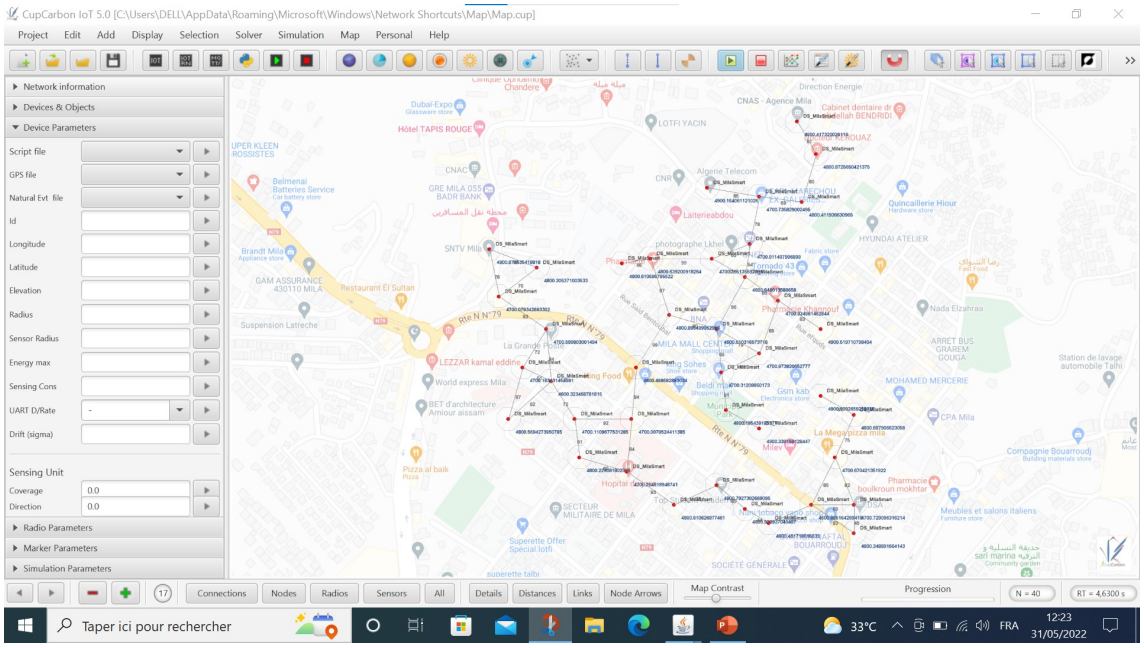

Figure 4.40: Mila Map before simulation

4.2.3 After simulation

The figure below represent the marking of the nodes in which the firewall should be placed.

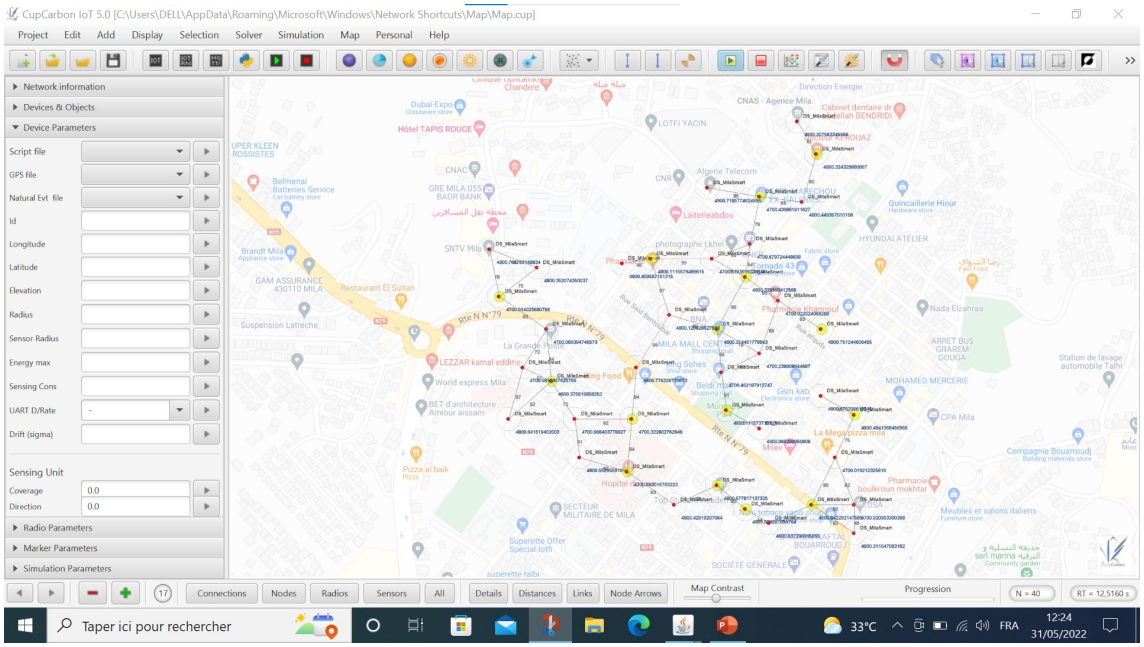

Figure 4.41: Mila Map after simulation

4.2.4 Comparison

Through our study and our experimentation on the centralized and distributed algorithm, we found the following points :

	Central Algorithm	Distrbuted Algorithm
Execution method	Executed step by step and each step we choose one node.	Executed in a parallel way and in several nodes.
Execution time	The execution time is long, it can take several hours in a large graph.	The execution time is short, no matter how large the graph is.
Communication	Nodes don't communicate with each other.	Nodes communicate with each other by synchronous messages.
Processor Status	The processor is in a stressed statebecause it is executed in one core ofthe processor.	The processor is in state ofrest because it is executed inseveral cores in the processor.
Complexity class	O (n): T(n)= n+4 Linear increase of the execution time when the parameter increases (if the parameter doubles, the time doubles).	O (1): T(n)= 41 The execution time does not depend on the data processed

Table 4.1: Comparison between the centralized and distributed algorithm

Conclusion

In this final chapter, we concluded with a simulation and evaluation of the performance of our parallel algorithm. We compared the obtained results with a central algorithm. The obtained results show that our parallel algorithm has a high performance in terms of execution time and speed and the processor cores used.

General conclusion

In this project, we tackled the problem of network optimization in an IoT environment. Objects are connected with information and communication technology, we presented the different IoT technologies and the method used to protect these connected objects we focused on firewalls, the theory of graphs necessary for the analysis of complex networks, so we gave a summary on the optimization problems and their methods of resolution.

In our project we have proposed a solution to the problem posed which is to develop a system that optimizes the complexity of graphs against attacks. We have used an algorithm that minimizes the number of firewalls in an IoT environment and ensures security performance against attacks.

The application of the algorithm makes it possible to simulate graphs in a complex network and find the optimal solution with the minimum number of firewalls to block attacks from the outside. The choice of node to be protected is ensured by the application of the domination algorithm.

At the end, our solution remains scalable as a future project using other variants of dominating parameter like secure dominating set .

Annex

IOT internet of things

Lifecycle attacks : It is a concept assuming that IoT devices, since they are quite advanced pieces of tech, lead their smart electronic lives that can be divided into three main stages with rather unimaginative but telling names: Beginning of Life (BoL), Middle of Life (MoL) and End of Life (EoL).

- During the BoL stage, the device may be preconfigured, but usually it has only some generic schemas (like default passwords).
- The focus within the MoL stage, apart from maintaining the device's basic functional purpose, should be kept on improving its reliability and maintainability.
- Finally, the EoL comes down to ensuring that the device may be easily and securely removed or replaced in case it is not worth anymore to spend resources on maintaining it, its current status is hard to determine or in case it's simply broken.

Physical attacks : occur when IoT devices can be physically accessed by anyone. With the majority of cybersecurity attacks occurring from the inside of a company, it's essential that your IoT devices are in a protected area, which is often not an option. Many physical cybersecurity attacks begin with the assailant inserting a USB drive to spread malicious code, which is why it's more important than ever to add AI-based security measures to ensure your devices and data are protected.

Bibliography

[1] Boulberhane Anis, Guermat Ishak *Studying and Implementing a Solution to the University Course Timetabling Problem*. Master thesis in computer science, Abd Elhafid Boussouf Mila University. 2020.

[2] Fairouz Beggas.*Decomposition and Domination of Some Graphs. Data Structures and Algorithms [cs.DS]*. Université Claude Bernard Lyon 1, 2017. English. tel-02168197.

[3] Mathieu Chapelle,. *Décompositions de graphes : quelques limites et obstructions. Complexité [cs.CC]*.Université d'Orléans, 2011. Français. tel-00659666v2.

[4] *https://en.wikipedia.org/wiki/Optimization problem*

[5] Marler, R. T. AND Arora, *J. S. Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization*, 2004, 369-395p.

[6] M. Merdjaoui B., *Optimisation multi-objectif par algorithmes génétiques et approche Pareto des paramètres d'usinage sous contraintes des limitations de production* . Mechanical engineering. university M'hamed Bougara Boumerdes, 2006. 149p.

[7] Basudeb Mondal and Kajal De., *An overview applications of graph theory in real field. International Journal of Scientific Research in Computer Science* , Dept of ECE and Goergen Institute for Data Science, 2(5):751–759, 2017.

[8] Gonzalo Mateos., *Graph Theory Review* ,Engineering and Information Technology, University of Rochester. 2020.

[9] mourad guettiche., *paramètre de graphes et protection des réseaux.* thèse de doctorat en science.,université de bejaia., 2019.

[10] Bechir A., *Résolution de problèmes d'optimisation par les systèmes multi-agents et les approches évolutionnaires.* Magister thesis: Data Mining et MultiMedia. Biskra: Université Mohamed Khider Biskra,2016,104p.

[11] Labed, Said. *Méthodes bio-inpirées hybrides pour la résolution de problèmes complexes.*Université Constantine 2. 2013.

[12] M. V. Bouquet, F. Delbot, Ch. Picouleau : *Partition of graphs with maximum degree ratio* ,(2020).

[13] Assia Brighen, Hachem Slimani, Abdelmounaam Rezgui, Hamamache Kheddouci., *Listing all maximal cliques in large graphs on vertex-centric Model.* , The Journal of Supercomputing., 2019.

[14] Loudini Meriem, Djemaioune Massika., *Les graphes sous Hdoop et Giraph : Algorithmes et implémentations.* , Université de Jijel. 2016.

[15] Boulmis Bilal, Lakhal Dounia, et Lalouci Ali., *A solution for managing firewalls in the Internet Of Things.* , Centre Universitaire Abdelhafid Boussouf Mila. 2021

[16] *https://www.guru99.com/iot-tutorial.html*

[17] *t.ly/vmhi*

[18] *https://www.arm.com/glossary/iot-security*

[19] *Security on Internet of Things (IOT) with Challenges and Countermeasures*, Volume 5, Issue 1, 2017 IJEDR, R.Vignesh and A.Samydurai, Valliammai Engineering College SRM Nagar, Kattankulathur-603203, TamilNadu, India.

[20] *t.ly/iF7C*

[21] *https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/inspired/smart-cities*

[22] Henry, Michel. *Pratiques expérimentales et modélisation: quelques questions didactiques posées par la simulation informatique.* 2005.

[23] Belattar, Brahim. *Modélisation et Simulation sur Ordinateur.* 2003/2004.

[24] Korichi, Ahmed. *Investigation sur la possibilité de l'évaluation de performance multicritères d'une entreprise par l'exploitation de la simulation sur ordinateur.*2004.

[25] aissani, Amar. *Modélisation et simulation.* 2007.

[26] *http://www.cupcarbon.com/*

[27] Mehdi, K., Lounis, M., Bounceur, A., and Kechadi, *Multi-Agent and Discrete Event Wireless Sensor Network Design and Simulation Tool.* , In IEEE 7th International Conference on Simulation Tools and Techniques (SIMUTools'14), Lisbon, Portugal, 2014.

[28] Ahcene Bounceur and all,*CupCarbon: A New Platform for the Design,Simulation and 2D/3D Visualization of Radio Propagation and Interferences in IoT Networks*,Universite de Bretagne Occidentale, Brest, France .

[29] E. J. Cockayne, P. J. P. Grobler, W. R. Gründlingh, J. Munganga, J. H. van Vuuren,*Protection of a graph. Util. Math.* 67 (2005), 19–32.

[30] *https://www.igi-global.com/dictionary/cognitive-radio-sensor-networks/26486*

[31] *https://www.microsoft.com/en-us/industry/government/resources/smart-cities*

[32] Couturier, Jean-François.*Algorithmes exacts et exponentiels sur les graphes :énumération, comptage et optimisation*. Université de Lorraine, 2012.

[33] Ahcene Bounceur and all,*A 3D Environment for IoT Simulation Under the CupCarbon Platform*,Laboratory Lab-STICC, University of Brest.Laboratory LIMED, University of Bejaia .Virtualys Company , Brest .

[34] *https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html*

[35] *https://us.norton.com/internetsecurity-emerging-threats-what-is-firewall.html*

[36] Ahcène Bounceur ,*Algorithmes Distribués*