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

N° Réf :……

Centre Universitaire
Abd elhafid Boussouf Mila

Institut des sciences et de la 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)

Etude et réalisation d'une application réseau à base de nœuds critiques

Préparé par

- ZOUARI Amel

Soutenu devant les jurys

Encadré par : LALOU Mohammed …..…..M.A.A
Président : GUETTICHE Mourad……M.A.A
Examinateur : MERABET Adil…….....M.A.A

Année universitaire : 2015/2016

Remerciements

En premier lieu, nous tenons à remercier notre DIEU "allah", notre créateur de nous avoir donné la force pour accomplir ce travail.

Nous adressons nos vifs remerciements à notre encadreur consultant Mr.Lalou Mouhamed pour nous avoir guidés tout au long de ce travail, pour sa compréhension, sa patience, sa compétence, et ses remarques qui nous ont été précieuses.

Nous présentons nos chaleureux remerciements aux enseignants du jury Mr.Mrabet Adil, Mr.Guettiche Mourad qui m'ont fait l'honneur de présider et d'examiner ce modeste travail.

Amel Zouari...

Résumé

L'étude de la diffusion dans les réseaux complexes est devenue une priorité dans ces dernières années. La diffusion peut être une propagation d'un virus dans un réseau informatique, d'une maladie dans une société, d'une rumeur dans un réseau social, etc. Son étude est abordée pour différents objectives, à savoir : la localisation de la source de la diffusion, la détection rapide de la contamination, le blocage de la propagation, etc. Une des plus importantes questions considérée est le blocage de la diffusion des entités négatives dont la propagation porte des risques sur le fonctionnement du réseau. Pour assurer une réponse optimale à cette question, différentes approches ont été proposées, l'une des plus efficaces est celle qui exploite les propriétés structurelles du réseau vue qu'il est le support dans lequel la diffusion peut avoir lieu. Dans ce travail, nous proposons d'utiliser les nœuds critiques du réseau comme pare-feu pour stopper la propagation d'une diffusion dangereuse. Un nœud critique est un nœud dont la suppression dégrade la connectivité du réseau. Ainsi, la chance de bloquer la diffusion augmente car une diffusion ne peut se propager que dans un environnement connecté. Cette approche assure une solution optimale parce que les variantes de *CNDP* (*Critical NodesDetection Problem)* cherchent souvent le minimum de nœuds critiques ou un nombre de nœuds adapté aux ressources existantes.

Mots-clés : Réseaux complexes. Diffusion. Les nœuds critiques. *CNDP* (*Critical NodesDetection Problem)*

Sommaire

Chapitre 01 : Réseaux & Graphes

Chapitre 02 : Réseaux complexes

Chapitre 03: probléme de détection de nœuds critiques (CNDP)

Chapitre 04 : Résalisation

Listes des Figures

Liste des Tableau :

Introduction générale

Introduction générale

De nombreux systèmes, aussi bien naturels qu'artificiels, peuvent être représentés par des réseaux, c'est-à-dire un ensemble de ressources ou de sites reliés par des liens. Les exemples de réseaux vont de l'Internet jusqu'aux les interconnexions de protéines, de personnes, d'équipements électriques, etc. Un système du monde réel est généralement compliqué, et son évolution ne suit pas un plan prédéfini. Ce genre de systèmes est représenté par ce qu'on appelle les « réseaux complexes », qui constituent un outil privilégié pour l'étude des interactions entre les entités de ces systèmes.

Récemment les réseaux complexes ont tiré beaucoup d'attention dans la recherche en raison de leur capacité à modéliser différents systèmes du monde réel. Les propriétés de ces réseaux permettent de décrire précisément les systèmes qu'ils représentent. Ces propriétés ont permis aussi d'étudier et de comprendre les différents phénomènes liés à ces systèmes, comme par exemple : la nature des interactions, le clustering (formant des communautés), le phénomène d'influence (un lien social peut déterminer un comportement), le phénomène d'achat (l'impact des liens sociaux sur la démographie), etc. Un des phénomènes les plus importants est le phénomène de la *diffusion*. La diffusion est la propagation d'une entité, dans le réseau, d'un individu à ses voisins jusqu'à la contamination de tout le réseau. Comme exemple de diffusion dans les réseaux complexes, on peut citer : la pandémie dans une population, la propagation d'un virus dans un réseau informatique, la diffusion d'une rumeur dans un réseau social, etc.

La diffusion dans les réseaux complexes a été étudiée pour différentes objectives, à savoir : la localisation de la source de diffusion, la détection rapide de la diffusion, la reconstruction des chemins de la propagation, le blocage de la diffusion, etc. Dans notre travail, nous nous intéressons à cette dernière, où la question à laquelle il faut trouver une solution optimale est comment bloquer la diffusion d'une entité nocive dont la propagation risque de compromettre le réseau et son fonctionnement ?

Différentes approches ont été proposées pour bloquer la diffusion dans un réseau complexe. L'une qui a montré son efficacité est celle qui se base sur l'exploitation des particularités structurelles du réseau. Elle consiste à déterminer dans une première étape les nœuds importants du réseau, et puis de bloquer la diffusion une fois arrive à ces nœuds. Pour déterminer les nœuds importants, différentes métriques ont été utilisées, comme exemple on peut citer : la centralité, le plus court chemin, la séparation en des composantes, etc. Récemment, le problème de détection de nœuds critique *CNDP* (en anglais *Critical Nodes Detection Problem)* a attiré beaucoup d'attention comme une alternative pour identifier les nœuds importants au sein d'un réseau en termes de connexité. *CNDP* est le problème qui cherche les nœuds dont la suppression minimise la connectivité dans le réseau. Dans ce

travail, nous nous intéressons au blocage de la diffusion dans un réseau complexe en utilisant le CNDP. Pour ce faire, ce mémoire est organisé en quatre chapitres comme suit :

Chapitre 1 : Réseaux & Graphes

Dans ce chapitre, nous présentons les différentes définitions et concepts de la théorie des graphes jugés nécessaires dans la modélisation des réseaux complexes, ainsi que les différents problèmes d'optimisation et leurs méthodes de résolution.

Chapitre 2 : Réseaux complexes

Ce chapitre est consacré à un état de l'art général sur les réseaux complexes, et les différents modèles de diffusion dans ces réseaux.

Chapitre 3 : Problème de détection de nœuds critiques.

Dans ce chapitre, nous définissons le problème de détection de nœuds critiques, et nous décrivons de façon détaillée ses différentes variantes.

Chapitre 4 : Réalisation

Ce chapitre contient trois parties, la première partie contient un état de l'art général sur la simulation. La deuxième partie contient une présentation des outils de développement que nous avons utilisés pour mettre en œuvre notre application. Dans la troisième partie, nous présentons notre application pour le blocage de la diffusion en utilisant les nœuds critiques du réseau.

Nous terminons ce mémoire par une conclusion générale qualificative des objectifs déjà tracés au début. Nous citerons aussi les perspectives de notre travail ainsi que les différentes améliorations à apporter à notre application pour qu'elle devienne plus performante.

Chapitre 1

Réseaux & graphes

Introduction

Beaucoup de modèles et systèmes autour de nous peuvent être représentés par des *réseaux*, où les éléments sont en interaction les uns avec les autres. Nous trouvons l'utilisation du terme *réseau* dans beaucoup de disciplines et sciences, y compris: la physique (transport, espace géographique), la biologie (les réseaux de neurones), les sciences sociales (réseaux d'individus), ect.

La modélisation et l'étude théorique des réseaux se fait en utilisant la théorie des graphes. Cette théorie constitue aujourd'hui un corpus de connaissances très important. En fait, les graphes permettent de manipuler plus facilement des objets et leurs relations avec une représentation graphique naturelle. L'ensemble des techniques et des outils mathématiques mis au point en théorie des graphes permet de démontrer facilement des propriétés, d'en déduire des méthodes de résolution des problèmes de nature difficiles.

Dans ce chapitre, nous présentons les différentes définitions et concepts de la théorie des graphes dont nous avons besoin dans la modélisation de réseaux.

1.1 Réseaux & théorie des graphes

L'approche la plus ancienne et la plus formelle pour étudier les réseaux n'est autre que la théorie des graphes. Cette théorie mathématique a des bases très anciennes mais des résultats datent du 20eme siècle. Sa principale caractéristique est l'utilisation d'un modèle simple et très puissant appelé *graphe*. Simple du fait qu'il est un schéma constitué tout simplement d'un ensemble de points reliés par des lignes, permettant de décrire un ensemble d'objets et leurs relations (les liens entre ces objets). Puissant du fait que ce schéma est capable d'illustrer différentes situations dans plusieurs domaines : mathématiques, physiques, sociologiques, etc, d'une part. D'autre part, sa manipulation permet de démontrer facilement des propriétés, d'en déduire d'autres, de développer des méthodes de résolution pour les problèmes confrontés, etc.

En utilisant la théorie des graphes, un réseau est représenté par un graphe constitué d'un ensemble d'éléments interagissant entre eux dont le type dépond du type du réseau (villes, individus, ordinateurs, etc.). Pour l'étudier, des outils très performants ont été développés tout en prenant en considération la nature (statique ou dynamique), la dimension (grande ou petite taille) et le type d'interactions (directionnel ou non) du réseau [1].

1.2 Notions de base

Définition 1.1 (Graphe)

Un graphe G est un couple (V, E), où V est un ensemble fini d'objets, et E un sous-ensemble de $V \times V$ constituant l'ensemble de liens reliant des couples d'objets. Les éléments de V sont appelés les sommets (ou les nœuds) du graphe, et ceux de E sont ses arêtes. Une arête $a \in E$ du graphe est une paire $a = (x, y)$, où x et y sont des sommets extrémités de a [2].

Définition 1.2 (Ordre et taille de graphe)

Etant donné un graphe $G = (V, E)$, on appelle ordre de G le nombre des nœuds ($|V|$), et taille de G, le nombre d'arêtes ($|E|$) [1].

Définition 1.3 (Sommet adjacent)

Soit un graphe $G = (V, E)$, et soient $x_i,\ x_j \in V$ deux sommets de G. x_j est dit adjacent à x_i s'il existe une arête entre x_i et x_j. L'ensemble des sommets adjacents d'un sommet x_i est défini par : $adj(x_i) = \{x_j \mid (x_i, x_j) \in E\}$ [3].

Définition 1.4 (Degré d'un sommet)

Dans un graphe G, le degré $d(x)$ d'un sommet x est le nombre d'arêtes incidentes à ce sommet, c'est-à-dit le nombre d'arêtes dont x est l'une des extrémités: $d(x) = |adj(x)|$. Un sommet qui n'apparaît dans aucune arête est dit isolé et son degré est nul [3].

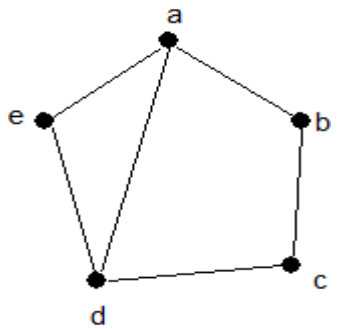


Figure 1. Graphe d'ordre 5 et de taille 6.

Exemple : la figure 1 représente un graphe $G = (V, E)$, avec L'ensemble de nœuds $V = \{a, b, c, d, e\}$ et celui d'arêtes $E = \{(a, b), (a, d), (a, e), (b, c), (c, d), (d, e)\}$. Les sommets adjacents au nœud a sont $adj(a) = \{b, d, e\}$ et sont degré d(a)=3.

Définition 1.5 (Graphe orienté)

Un graphe orienté est un graphe dans les arêtes sont orientées, *i.e.* un sommet x_i peut être en relation avec un autre sommet x_j sans que x_j soit nécessairement en relation avec x_i. Les arêtes d'un graphe orienté sont appelées *arcs*. Dans un arc (x_i, x_j), x_i est le sommet initial, et x_j le sommet terminal, et il est représenté graphiquement par $x_i \rightarrow x_j$ [3].

Définition 1.6 (successeurs / prédécesseurs)

Dans un graphe orienté. Pour chaque sommet x_i on distingue les sommets successeurs *succ* (x_i), *et* les sommets prédécesseurs *pred* (x_i) comme suit [3] :

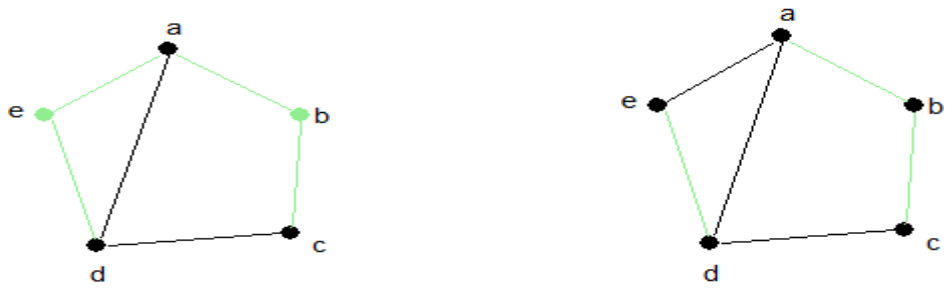
$$succ(x_i) = \{x_j \mid (x_i, x_j) \in E\}.$$

$$préd(x_i) = \{x_j \mid (x_j, x_i) \in E\}.$$

Définition 1.7 (Sous graphe/ Graphe partiel)

Etant donné un graphe $G = (V, E)$, un sous-graphe de G est un graphe $H = (V(H), E(H))$ tel que $V(H)$ est un sous-ensemble de sommets de V noté $V(H) \subseteq V$, et $E(H)$ sont les arêtes induites par $V(H)$ (les arêtes de G ayant leurs extrémités dans $V(H)$) : $E(H) = \{(x, y) \in E \mid x, y \in V(H)\}$, On obtient H en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.

Un graphe partiel de G est un graphe obtenu en enlevant une ou plusieurs arêtes au graphe G [2] [3].



(a) ***(b)***

Figure 2. (a) Sous-graphe H de G / (b) Graphe partiel F de G.

Exemple : La figure 2 (a) montre un sous-graphe $H = (V(H), E(H))$ du graphe G, où V(H)= $\{a, d, c\}$, et $E(H) = \{(a, d), (d, c)\}$. La figure 2 (b) montre un Graphe partiel $F = (V(F), E(F))$ du graphe G, où E(F)= $\{(a, e), (a, d), (d, c)\}$.

Définition 1.8 (L'ensemble stable)

Un ensemble stable d'un graphe $G = (V, E)$ est un sous-graphe induit par un sous ensemble des sommets, $S \subseteq V$. Tel qu'il n'existe pas d'arêtes entre ses sommets : $\forall\, x_i, x_j \in S, (x_j, x_i) \notin E$ [3].

Définition 1.9(Graphe complet)

Un graphe $G = (V, E)$ est dit complet si chaque sommet $x_i \in V$est adjacent à chaque autre sommet $x_j \in V$. Autrement dit chaque sommet est relié à tous les autres. Le graphe complet d'ordre n est noté K_n. Dans ce graphe chaque sommet est de degré $n - 1$ [4].

Définition 1.10 (Clique)

Une clique *C(V(C),E(C))* d'un graphe $G = (V, E)$ est un sous-graphe complet tel que : $V(C) \subset V \mid \forall\, x_i, x_j \in V(C), (x_j, x_i) \in E$ [3].

Définition 1.11 (Chemin)

Un chemin d'un sommet u vers un sommet v est une séquence d'arêtes, disions $\{(x_0, x_1), (x_1, x_2), \ldots . (x_i, x_{i+1}) \ldots . (x_{k-1}, x_k)\}$ tels que $u = x_0$ et $v = x_k$, et $(x_i, x_{i+1}) \in E$ pour $i \in [0, k-1]$. S'il existe un chemin de u à v on dit que v est accessible à partir de u. La longueur d'un chemin est le nombre d'arêtes. Un chemin forme un *cycle* si $u = x_0 = v$. Un graphe sans cycle est dit acyclique [3].

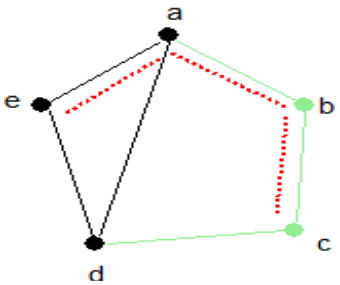


Figure 3. Clique de G / Chemin de e a c.

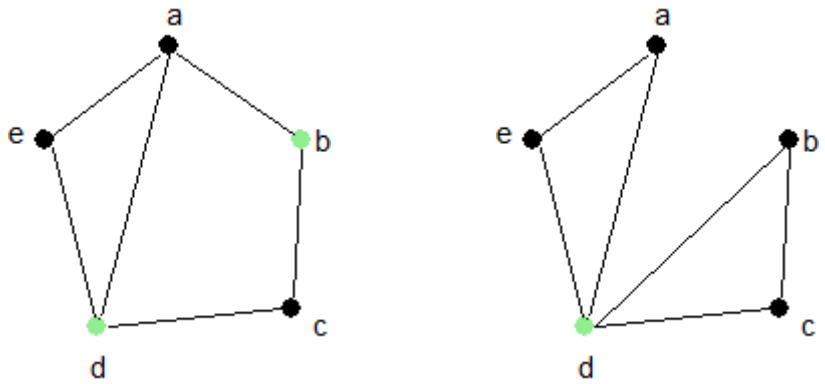
Exemple : la figure 3 représente un graphe $G = (V, E)$, où L'ensemble C= $\{a, e, d\}$ est une clique de G et l'ensemble $S = \{e, b\}$ est un ensemble stable de G. Le chemin entre e et c est $ch = \{(e, a), (a, b), (b, c)\}$.

Définition 1.12 (Graphe connexe)

Un graphe $G = (V, E)$ est dit connexe si on peut atteindre n'importe quel sommet à partir d'un sommet quelconque, *i.e.* pour toute paire de sommets x_i et x_j, il existe un chemin $\{x_i, x_k, \ldots . x_j\}$ entre x_i et x_j. Dans le cas contraire, le graphe est dit *non connexe*. On appelle *composante connexe* un sous-ensemble de sommets connexe [4]. Le graphe nous servant d'illustration depuis le début est un graphe connexe.

Définition 1.13 (séparateur /Point d'articulation)

Un séparateur d'un graphe est un sous-ensemble de sommets dont la suppression rend le graphe non-connexe, ainsi retirer un séparateur du graphe augmente le nombre de composantes connexes. Si aucune de ses parties n'est un séparateur le séparateur est dit *minimal*. Un point d'articulation est un séparateur de taille une [1].



(a) ***(b)***

Figure 4. (a) Séparateur minimal de G / (b) Point d'articulation de G.

Exemple : Soit un graphe $G = (V, E)$dans la figure 4 (a) L'ensemble $\{b, d\}$ est un séparateur minimal de G. Soit un graphe $G' = (V, E)$dans la figure 4 (b) le sommet "d"est un point d'articulation de G'.

1.3 Quelques classes de graphes

Une classe (ou famille) de graphes est un ensemble de graphes défini par une ou plusieurs propriétés structurelles. Touts graphe G possédant ces propriétés appartiennent à la classe de graphes correspondante. On note aussi qu'il y a des relations d'inclusion entre les classes de graphes [4] .

Une propriété de graphe *P* peut être :

- *Non triviale* si elle est vraie pour un ensemble infini de graphes et fausse pour un ensemble infini de graphes.
- *Héréditaire* si elle est close par suppression de sommets, *i.e.* pour tout graphe G satisfaisant *P*, tout sous-graphe induit de G satisfait aussi *P*.
- *Monotone* si elle est close par suppression de sommets ou d'arêtes, *i.e.* pour tout graphe G satisfaisant *P*, tout sous-graphe de G (non nécessairement induit) satisfait *P* [5].

Dans ce qui suit, nous présentons quelques classes de graphes :

1.3.1 Arbre & Arborescence

Un arbre $T = (V, E)$ est un graphe non orienté, connexe et sans circuit, avec un minimum d'arêtes, si on enlève une on déconnecte le graphe. Dans un arbre, deux sommets distincts sont reliés par un et un seul chemin. Cependant, une arborescence est un graphe orienté sans circuit disposant d'un sommet (unique) nommé racine r, tel que pour tout sommet x_i il existe un et seul chemin orienté de la racine r à x_i .

On appel arbre couvrant Td'un graphe G le sous-graphe acyclique de G contenant tous les nœuds de G de telle façon que l'ajout d'une arête à T crée un cycle. Un graphe n'admet un arbre couvrant que s'il est connexe [3].

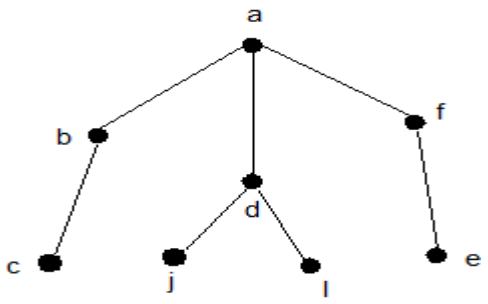


Figure 5. Un arbre.

1.3.2 Graphes parfaits

La classe de graphes parfaits est définie à partir de la propriété de coloriage [6]. Elle regroupe les graphes pour lesquels le problème de la coloration de graphe pouvait être résolu efficacement. En théorie des graphes, colorer un graphe signifie attribuer une couleur à chacun de ses sommets de manière à ce que deux sommets reliés par une arête soient de couleurs différentes, tout en utilisant un nombre minimal de couleurs, dit nombre chromatique [2].

1.3.3 Graphes chordaux

Un graphe G est un graphe chordal (ou triangulé) si tout cycle de G de longueur supérieure ou égale à quatre contient au moins une corde [5]. Un graphe triangulé est aussi un graphe parfait. Les graphes triangulés possèdent beaucoup de caractéristiques et de propriétés structurelles, parmi les plus importantes, on cite l'existence d'un *ordre d'élimination simplicial*. Un ordre d'élimination simplicial est un ordre sur les sommets du graphe tel que, pour tout sommet x, les voisins de x apparaissant après celui-ci dans l'ordre d'élimination forment une clique. Cette caractérisation est très utilisée pour l'étude de divers problèmes sur les graphes triangulés. Elle permet, par exemple, la reconnaissance d'un graphe triangulé en temps *linéaire* par un parcours *BFS* (Breadth First Search) [5].

1.3.4 Graphes d'intervalles

Un graphe d'intervalles est le graphe d'intersection d'un ensemble d'intervalles de la droite réelle. Chaque sommet représente un intervalle, et une arête relie deux sommets lorsque les intervalles correspondants ont une intersection non vide.

Les graphes d'intervalle sont bien adoptes pour représenter les problèmes d'ordonnancement. En effet, un problème d'ordonnancement se représente souvent par un ensemble de tâches, avec un temps de début et une durée pour chacune, et des chevauchements temporels entre les tâches,, où on cherche un ordonnancement de tâches, qui optimise une fonction objective telle que la minimisation des ressources utilisés pour achever ces tâches [5].

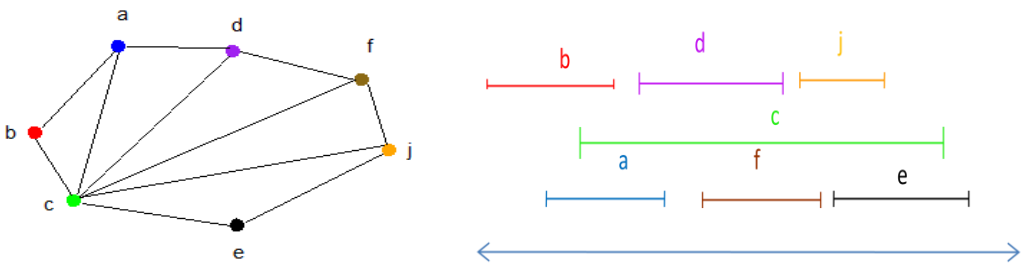


Figure 6. Un graphe chordal et son graphe d'intervalle.

1.3.5 Graphes aléatoires

Un graphe aléatoire est un graphe mathématiquement généré par un processus aléatoire. Un graphe aléatoire noté $G(N, p)$ est obtenu à partir d'un ensemble de N nœuds, en ajoutant aléatoirement des arêtes, crées indépendamment avec une probabilité p [7].

Les graphes aléatoires jouent un rôle important dans l'évaluation des performances et de la complexité des algorithmes [8].

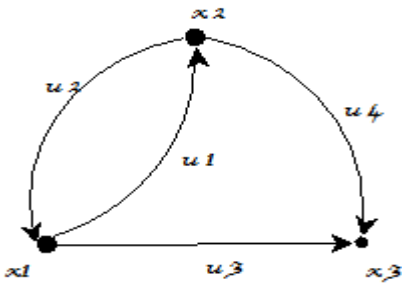
1.4 Représentation des graphes

Il existe différentes façons pour représenter un graphe. Du point de vue de l'efficacité des algorithmes, ces représentations ne sont pas équivalentes. On distingue principalement la représentation par matrice d'adjacence, par matrice d'incidence sommets-arcs (ou sommets-arêtes dans le cas non orienté) et par listes d'adjacence.

1.4.1 Matrice d'adjacence

On considère un graphe G orienté. La matrice d'adjacence M fait correspondre les sommets origines des arcs qui sont placés en lignes dans la matrice M, aux sommets destinations qui sont placés en colonnes. L'existence d'un arc $a = (x_i\ , x_j\ )$ dans le graphe G se traduit par la présence d'un 1 à l'intersection de la ligne i et la colonne j, et l'absence d'arc se traduit par la présence d'un 0 [3] [9] :

$$\mathrm{M[i,j]} = \begin{cases} 1 \text{ si } a = (x_i\ , x_j\ ) \in E \\ 0 \text{ sinon} \end{cases}$$



origine \ destination	x_1	x_2	x_3
x_1	0	1	1
x_2	1	0	1
x_3	0	0	0

Figure 7. Représentation de graphe par Matrice d'adjacence.

Dans le cas de graphes non orientés, la matrice est symétrique par rapport à sa diagonale descendante.

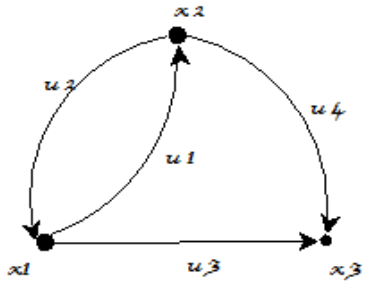
La matrice d'adjacence d'un graphe ayant n sommets nécessite n^2 emplacements mémoire. Donc dans le cas où le nombre d'arcs $|E| < n^2$ cette représentation est loin d'être optimale [3] [9].

1.4.2 Matrice d'incidence

Etant donné un graphe G orienté, la matrice d'incidence M fait correspondre les sommets qui sont placés en ligne dans la matrice M aux arcs qui sont placés en colonne dans la matrice M. L'existence d'un arc sortant $a = (x_i\ , x_j\ )$ dans le graphe G se traduit par la présence d'un 1 à l'intersection de la ligne i et la colonne j, et l'existence d'un arc entrant $a = (x_i, x_j\ )$ se traduit par la présence d'un -1 à l'intersection de la ligne i et de la colonne j, et l'absence d'arc se traduit par la

présence d'un 0 [9]:

$$M[i,j]\begin{cases} 1 \text{ si } a = (x_i\ , x_j\ ) \in E. \\ -1 \text{ si } a = (x_j\ , x_i\ ) \in E. \\ 0 \text{ sinon .} \end{cases}$$



	u_1	u_2	u_3	u_4
x_1	1	-1	1	0
x_2	- 1	1	0	1
x_3	0	0	-1	-1

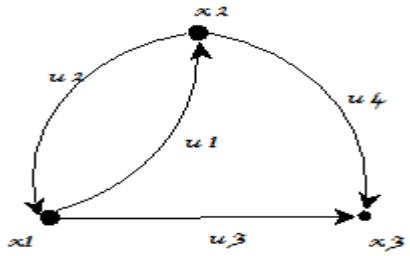
Figure 8. Représentation de graphe par Matrice d'incidence.

Dans le cas de graphes non orientés, L'existence d'une arête $a = (x_i\ , x_j\ )$ dans le graphe G se traduit par la présence d'un 1 à l'intersection de la ligne *i* et la colonne *j*, et l'absence d'arête se traduit par la présence d'un 0.

Cette représentation utilise un espace mémoire de $n \times m$ cases, tel que $n = |V|$ et $m = |E|$ [9].

1.4.3 Listes d'adjacence

Soit le graphe $G = (V, E)$. On suppose que les sommets de V sont numérotés de 1 à n, avec $n = |V|$. La représentation de G par listes d'adjacence consiste en un tableau T de n listes, une pour chaque sommet de V. Pour chaque sommet $x_i \in V$, la liste d'adjacence $T[x_i]$ est une liste chainée de tous les sommets x_j tels qu'il existe un arc $(x_i\ , x_j\ ) \in E$. Autrement dit, $T[x_i]$ contient la liste de tous les sommets successeurs de x_i . Les sommets de chaque liste d'adjacence sont généralement chainés selon un ordre arbitraire. Si le graphe est valué, on peut stocker dans les listes d'adjacence, en plus du numéro de sommet, l'évaluation de l'arête [3] [9].



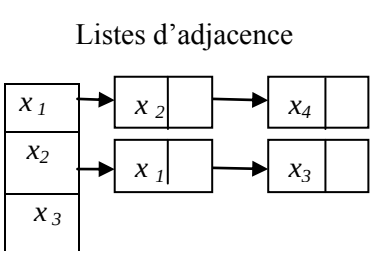


Figure 9. Représentation de graphe par Listes d'adjacence.

Dans le cas de graphe non orienté, pour chaque arête (x_i, x_j), on aura x_j qui appartiendra à la liste chainée de $T[x_i]$, et aussi x_i qui appartiendra à la liste chainée de $T[x_j]$ [3] [9].

Cette représentation utilise un espace mémoire de $n + m$ cases, tel que $n = |V|$ et $m = |E|$ .La liste d'adjacence d'un graphe ayant n sommets nécessite n emplacements mémoire. L'avantage de la représentation par listes d'adjacence par rapport a celle par matrice d'adjacence, est le gain obtenu en espace mémoire; ce type de représentation est donc mieux adapté pour une implémentation efficace [9].

1.5 La théorie de graphes et l'optimisation combinatoire

De manière générale, un graphe permet de représenter la structure et les connexions d'un ensemble complexe en exprimant les relations entre ses éléments. Les graphes constituent donc une méthode puissante qui permet de modéliser des situations différentes de problèmes. Dans le cas général résoudre ces problèmes nous ramène à l'étude de sommets et d'arête, c'est-à-dire l'exploration de graphe.

L'exploration d'un graphe nous permet d'obtenir un ensemble important de solutions potentielles pour un problème solvable. S'il s'agit de trouver la meilleure solution possible, le nombre important de ces choix rend impossible leurs parcours exhaustif en vue de recenser le choix le plus adéquat. L'existence de ce nombre de choix provoque une explosion combinatoire. On qualifie généralement de combinatoire, les problèmes d'ont la résolution se heurte à une explosion du nombre de combinaisons à explorer. L'exploration de toutes ces combinaisons en vue de trouver la meilleure solution est un problème d'optimisation combinatoire [10].

La notion de problème combinatoire est formellement caractérisée par la théorie de la complexité qui propose une classification des problèmes en fonction de la complexité de leur résolution. Nous introduisons ici la notion de complexité, et les classes de problèmes. Par la suite, nous parlons des différentes méthodes de résolution [10].

1.5.1 Problème d'optimisation combinatoire

Un problème d'optimisation consiste à chercher une instanciation d'un ensemble de variables soumises à des contraintes de façon à maximiser ou minimiser un critère. À chaque problème d'optimisation, on peut associer un *problème de décision* dont le but est de déterminer s'il existe une solution pour laquelle la fonction objectif soit supérieure (resp. inférieure) ou égale à une valeur donnée [11]. Lorsqu'un seul objectif (critère) est donné, le problème d'optimisation est mono-objectif. Dans ce cas, la solution optimale est clairement définie, c'est celle qui a le coût optimal (minimal,

maximal). Dans le cas ou plusieurs objectifs (critères) sont donnés, le problème d'optimisation est multi-objective. L'optimisation multi-objective est une branche de l'optimisation combinatoire dont la particularité est de chercher à optimiser simultanément plusieurs objectifs d'un même problème (contre un seul objectif pour l'optimisation combinatoire classique). Elle se distingue de l'optimisation multidisciplinaire par le fait que les objectifs à optimiser portent ici sur un seul problème[8].

Généralement ces critères sont employés pour classifier les différents problèmes d'optimisation. Ainsi qu'on résume dans le tableau ci- dessous :

Paramètre	**Valeurs**	**Type de problème**
Domaine des variables	Continu	Problème continu
	Discret	Problème combinatoire
Fonction objectif	Linéaire	Problème linéaire
	Non linéaire	Problème non linéaire
Nombre de fonctions objectifs	Scalaire	Problème mono-objectif
	Vectorielle	problème multi-objectif
Contraintes	Existence	Problème avec contrainte
	Non Existence	Problème sans contrainte
l'environnement	La fonction objective change dans le temps	problème dynamique

Tableau 1. Classes des problème d'optimisation.

On s'intéresse ici à l'optimisation combinatoire qui regroupe une large classe de problèmes d'optimisation ayant des applications dans de nombreux domaines aussi variés que la gestion, l'ingénierie, la production, les télécommunications, les transports, l'énergie et les sciences sociales. Généralement, le temps de résolution ainsi que l'espace mémoire requis pour résoudre un problème d'optimisation combinatoire est excessivement large. La théorie de la complexité propose une classification à ces problèmes en fonction de la complexité de leur résolution [12].

1.5.2 Complexité théorique d'un problème

La complexité est une notion cruciale en algorithmique et plus généralement en informatique. Elle nous permet d'étudier la difficulté d'un problème, afin de déterminer une bonne méthode à utiliser pour le résoudre. En effet, pour chaque problème posé, il est toujours possible de trouver une multiplicité d'algorithmes permettant de le résoudre. Cependant, pour résoudre un problème, on est appelé généralement à trouver l'algorithme le plus efficace. Donc, la question qui se pose est: quel algorithme est le plus performant (efficace) parmi ceux possibles de résoudre le problème?

La notion d'efficacité induite toutes les ressources de calcul nécessaires pour exécuter un algorithme. Or, le temps d'exécution est généralement le facteur dominant pour déterminer si un algorithme est assez efficace pour être utilisé dans la pratique, pour cela on se concentre principalement sur cette ressource, mais la taille des données joue aussi un rôle dans une telle détermination. La détermination de l'efficacité en fonction du temps d'exécution et d'espace mémoire dépend des caractéristiques de la machine et d'autres facteurs liés au système d'exploitation, et pour définir plus rigoureusement ce qu'est l'efficacité d'un algorithme, on doit considérer une approche complètement indépendante de toutes ces considérations. L'approche est alors de calculer la complexité de l'algorithme, qui fournit un ordre de grandeur du nombre d'opérations élémentaires nécessaires pour que l'algorithme retourne la solution du problème [12].

Définition 1.14 (complexité algorithmique)

On définit la complexité algorithmique d'un problème comme une estimation du nombre d'instructions à exécuter pour résoudre les instances de ce problème. La fonction de complexité d'un algorithme fait correspondre pour une taille donnée du problème le nombre maximum d'instructions nécessaires pour résoudre une instance quelconque de cette taille [12].

De façon générale on distingue deux grands types de problèmes :

- *Les problèmes de décision* : ils posent une question dont la réponse est oui ou non. Ainsi, ce qui est important est l'existence d'une solution, et ce n'est pas nécessaire de la trouver.
- *Les problèmes d'existence* : dans ce cas, la résolution du problème ne s'arrête plus au point de savoir si le problème admet ou non une solution. L'algorithme doit être en mesure de fournir la solution si celle-ci existe [10].

1.5.3 Les différentes mesures de complexité

Il existe trois mesures différentes de la complexité : la complexité dans le meilleur des cas, la complexité en moyenne cas et la complexité dans le pire des cas. L'analyse dans le meilleur des cas informe sur le temps minimum de calcul qu'effectue un algorithme. L'analyse en moyenne cas permet de connaître le temps moyen de calcul d'un algorithme. Elle renseigne sur le comportement général d'un algorithme. L'analyse dans le pire des cas informe sur le temps maximum de calcul qu'effectue un algorithme. Ce dernier est le plus courant. L'intérêt de l'analyse dans le pire des cas est d'évaluer les algorithmes dans leur fonctionnement extrême, et ainsi aucune mauvaise surprise n'est à prévoir, car le pire des cas est connu à l'avance [8].

1.5.4 Classes de complexité de problème

La notion de classe de complexité permet de classer les problèmes en fonction de leur complexité. La définition des classes de problèmes fait appel aux notions d'algorithmes déterministes et non-déterministes :

- *Algorithme déterministe* : est un algorithme classique qui fait des choix déterministes à chaque étape.
- *Algorithme non-déterministe* : set un algorithme théorique lorsqu'il se trouve devant plusieurs voies possibles il peut donner la mieulleur solution on utilisant des horistique .Autrement dit de choisir systématiquement la meilleure [12].

Dans ce qui suit, nous citons les principales classes de complexité selon le temps d'exécution et selon l'espace mémoire requis pour résoudre le problème :

1.5.4.1 Classes en temps

Dans cette classification, le facteur considéré est le temps d'exécution. Les différentes classes sont :

- ***La classe P (polynomial)****:* un problème de décision est dans *P* si une décision (concernant l'existence ou non d'une solution) peut être prise par un algorithme déterministe en un temps polynomial par rapport à la taille de l'instance , c'est-à-dire pouvant être résolu par un algorithme de complexité polynomiale d'ordre $O(n^k)$ pour un *k* constant donné et pour une instance de taille *n*. La classe *P* contient tous les problèmes relativement faciles, *i.e.* pour lesquels on connaît des algorithmes efficaces [12].

  Exemple : *le problème de la connexité dans un graphe*.

  Étant donné un graphe de *N* sommets. La question est de savoir si le graphe est connecté ou non. Autrement dit si toutes les paires de sommets sont reliées par un chemin. Pour le

résoudre, on dispose de l'algorithme de parcours en profondeur DFS, qui va construire un arbre couvrant du graphe à partir d'un sommet. Si cet arbre contient tous les sommets du graphe, alors le graphe est connexe. Dans le cas d'une implémentation par matrice d'adjacence DFS est d'une complexité $O(n^2)$, donc le problème est bien dans la classe P [3].

- ***La classe NP (Non-déterministe Polynomial)*** : il ne signifie pas non polynomial mais *polynomial non-déterministe*. La classe NP est donc une *extension* de la classe *P*, en autorisant des choix non déterministes pendant l'exécution de l'algorithme. La classe *NP* est la classe des problèmes de décision pour lesquels la réponse "oui" peut être décidée par un algorithme non-déterministe en un temps polynomial par rapport à la taille de l'instance. Intuitivement, les problèmes dans *NP* sont tous les problèmes qui peuvent être résolus en énumérant l'ensemble d'un grand nombre de solutions possibles (éventuellement exponentiel), et en les testant avec un algorithme polynomial. c'est-à-dire pouvant être résolus par un algorithme de complexité d'ordre $O(k^n)$ pour un k constant donné et pour une instance de taille n [12] [10].

  Par exemple, considérons le problème de la factorisation d'un entier en produit de facteurs premiers. On ne sait pas s'il existe un algorithme polynomial qui sache le résoudre. On ne sait donc pas si ce problème est dans *P*. Par contre, étant donné k nombres $p_1, p_2 \dots p_k$, il est trivial de vérifier si $n = p_1 \times p_2 \times \dots \times p_k$. Ce problème est donc dans *NP* [12].
- ***La classe NP-complets***: la classe *NP-complet* contient les problèmes les plus difficiles de la classe *NP*. Un problème *NP-complets* possède la propriété que tout problème dans *NP* peut être transformé en celui-ci en temps Polynomial. C'est-à-dire qu'un problème est *NP-complets* quand tous les problèmes appartenant à *NP* lui sont réductibles. Si on trouve un algorithme polynomial pour un problème *NP-complets*, on trouve alors automatiquement une résolution polynomiale pour tous les problèmes de la classe NP.

  L'exemple typique de cette classe est le problème du voyageur de commerce PVC (en anglais Travelling Salesman Problem : TSP) [12] [10].
- ***La classe NP-difficiles*****:** un problème est *NP-difficile* s'il est plus difficile qu'un problème *NP-complet*, c'est-à-dire s'il existe un problème *NP-complet* se réduisant à ce problème par la réduction de Turing [10].

  L'exemple typique de cette classe de problème est le problème du sac à dos multidimensionnel.

1.5.4.2 Classes en espace

Dans cette classification, le facteur considéré est l'espace mémoire requis pour trouver une solution au problème. Les différentes classes sont :

- ***La classe L (logarithmique)***: la classe des problèmes qui peuvent être résolus en un espace logarithmique sur une machine déterministe.
- ***La classe NL (Non-déterministe logarithmique):*** la classe des problèmes qui peuvent être résolus en un espace logarithmique sur une machine non-déterministe.
- ***La classe P- SPACE (Polynomial space):*** la classe des problèmes qui peuvent être résolus en un espace polynomial sur une machine déterministe.
- ***La classe EXP-SPACE (Exponentiel space)*** : la classe des problèmes qui peuvent être résolus en espace exponentiel [8].

1.5.5 Méthodes de résolution des problèmes d'optimisation

Généralement la résolution d'un problème d'optimisation passe par trois étapes:

- *L'analyse du problème* : regroupant la caractérisation du problème puis sa modélisation.
- *L'expression de l'objectif à optimiser* : qui nécessite une bonne connaissance du problème.
- *Le choix de la méthode de résolution* : qui permet à partir de la représentation du problème dans un espace de recherche d'obtenir une solution ou un ensemble de solutions optimales par rapport à la fonction d'évaluation [12].

Plusieurs méthodes de résolution existent dans la littérature pour les problèmes d'optimisation combinatoire. Ces méthodes font partie de deux groupes de nature différente : les algorithmes exactes et les algorithmes approchées (ou heuristiques).

1.5.5.1 Les algorithme exactes

Les algorithmes exacts sont des méthodes qui permettent de répondre de manière exacte à un problème. Pour les problèmes de décision, l'idée la plus simple pour résoudre exactement un problème est d'énumérer tous les sous ensembles susceptibles d'être des solutions et de les tester en cherchant une solution optimale.

Ainsi, les méthodes exactes garantissent la complétude de la résolution, et donnent la solution optimale à touts les coups. Mais de point de vue de complexité, et malgré les progrès réalisés pour cette approche, le temps de calcul nécessaire pour trouver une solution optimale risque d'augmenter exponentiellement avec la taille du problème. Les méthodes exactes rencontrent généralement des

difficultés face aux applications de taille importante, elles sont très exigeantes sur les ressources des ordinateurs, par exemple. Pour le problème de voyageur de commerce, la solution exacte d'un problème symétrique, avec 2392 villes a été déterminée sur une période de plus de 2392 ! heures sur ordinateur super puissant [12].

Dans cette famille d'algorithmes, nous pouvons citer la programmation linéaire (e.g simplexe), les méthodes exactes reposant sur le principe "diviser-et-régner" (Pour ce type de méthodes, on décompose le problème en plus petits sous-problèmes, plus faciles à résoudre, et on combine les résultats jusqu'à arriver à résoudre le problème initial), les méthodes par séparation et évaluation (branch and bound) et ses dérivées (branch and cut & branch and price) [12].

1.5.5.2 Les algorithmes approximatifs

Contrairement aux algorithmes exactes, les algorithmes approximatifs (heuristiques) permettant de trouver une solution proche de la solution optimale mais n'est pas optimale. Ils ont l'avantage de permettre de trouver une solution en un temps raisonnable. Ils constituent donc une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille. Les méthodes approchées sont fondées principalement sur diverses heuristiques, souvent spécifiques à un type de problème.

On peut citer des heuristiques très simples comme les algorithmes gloutons (pour ce type on fait une succession de choix optimaux localement, jusqu'à ce que l'on ne puisse plus améliorer la solution, et ce, sans retour en arrière possible), ou les approches par amélioration itérative.

Pour améliorer cette approche, des progrès importants ont été réalisés permettant l'apparition d'une nouvelle génération des algorithmes approchés puissants et généraux, souvent appelés métaheuristiques. Dans cette catégorie, on peut citer l'exemple de recuit simulé, la recherche tabou, les algorithmes évolutionnaires, les algorithmes de colonies de fourmis, etc. [12].

Conclusion

Dans ce chapitre, nous avons présenté les différentes notions de base sur la théorie des graphes utilisés dans ce rapport. Nous avons aussi parlé de la complexité des algorithmes, ainsi que de différents types de problèmes d'optimisation et leurs approches de résolution.

Chapitre 2

Réseaux complexes

Introduction :

Alors que la théorie des graphes s'intéresse à tout type de graphes, la théorie des réseaux s'intéresse aux graphes présents dans le monde réel. Son existence remonte à 1736 avec Euler qui l'utilisa pour résoudre le problème des sept ponts de Königsberg [1]. Récemment, les scientifiques se sont intéressés aux réseaux complexes en tant que modèles de systèmes réels. Comme exemple de systèmes complexes, nous pouvons citer : les chaînes alimentaires d'écosystèmes, les réseaux d'interaction protéines-protéines, les réseaux de contacts entre les individus (réseaux sociaux en ligne), le réseau internet, le réseau WWW, etc.

Les études empiriques faites sur les systèmes complexes ont mis en évidence plusieurs propriétés permettant des descriptions topologiques plus précises des réseaux qu'ils forment. Ces propriétés ont permis à leur tour d'expliquer divers phénomènes associés aux réseaux tels que le processus de diffusion. La propagation des rumeurs dans un réseau social, de virus dans un réseau informatique, ou d'agents pathogènes dans une population, sont des exemples de processus de diffusion dans un réseau.

Dans ce chapitre, nous allons présenter les systèmes complexes, leurs propriétés ainsi que les modèles qui les représentent, et vu que nous nous intéressons à l'étude du phénomène de diffusion dans ces réseaux, nous présenterons par la suite les différents modèles théoriques de diffusions.

2.1 Réseaux complexes

2.1.1 Réseau complexe ?

Un réseau complexe est un ensemble d'éléments en interactions mutuelles entre eux, ou le comportement global du système ne peut pas être déduit de la somme de ses parties et de leurs propriétés [9]. Une définition semblable est donnée dans [13]: un réseau complexe est un système auto-organisé dont l'évolution fait émerger des propriétés non prévues au départ.

2.1.2 Propriété des réseaux complexes

2.1.2 .1 Structure de communauté

Dans un réseau complexe, une communauté est un ensemble de nœuds fortement liée entre eux et faiblement liés avec les autres nœuds du graphe. Généralement, les nœuds d'une même communauté partagent les mêmes centres d'intérêt ou ayant le même profil. On dit qu'un réseau possède une structure de communauté s'il prend la forme d'un ensemble des communautés interconnectées [14] [15].

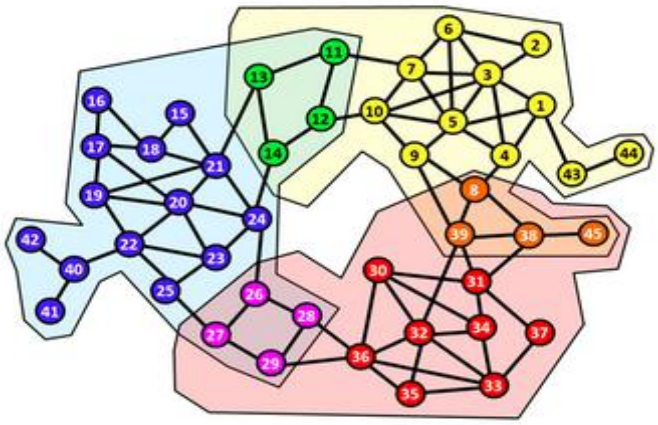


Figure 10 . Réseau avec une structure de communauté.

2.1.2.2 Adage

L'adage « l'ami de mon ami est mon ami », ou l'effet de regroupement (*clustering*) reflète le concept *d'agrégation* (ou de transitivité) dans le réseau complexe. C'est une mesure de probabilité que deux nœuds voisins d'un même nœud ont souvent plus de chances d'être également voisins l'un de l'autre [15]. Pour quantifier cette tendance dans un réseau on utilise le coefficient d'agrégation $C \in [0,1]$ comme suit:

$$C = \frac{\text{nombre de cycles de longueur 3 (triangles) dans le reseau}}{\text{nombre de chemins de longueur 2 dans le reseau}} .$$

Une valeur nulle de ce coefficient indique l'absence de triangles dans le réseau, tandis que C = 1 indique que chacune des paires de voisins de chacun des nœuds du réseau sont eux même voisins [16].

2.1.2.3 Distance moyenne

La distance $d_G\ (x, y)$ entre x et y est la longueur de plus court chemin entre x et y, elle est aussi appelée distance géodésique. La longueur géodésique moyenne de chemin entre chacune des paires de nœuds est une caractéristique globale du réseau qui représente le niveau de séparation typique entre deux nœuds [17]. Dans un réseau complexe elle est souvent de l'ordre du logarithme du nombre de nœuds ou inferieure [16] .

2.1.2.4 Distribution des degrés

Le degré d'un nœud est le nombre de liens qu'il a avec d'autres nœuds du réseau. On appelle premiers voisins deux nœuds qui sont directement reliés par un lien, deuxièmes voisins deux nœuds qui peuvent être minimalement rejoints par l'intermédiaire d'un autre nœud, et ainsi de suite. Comme

son nom l'indique, *la distribution des degrés P(k)* est la distribution du nombre de liens qu'ont les nœuds dans un réseau, ou de façon équivalente la probabilité qu'un nœud choisit le degré *k*. Cette distribution s'obtient en traçant un histogramme des degrés des nœuds du réseau, et elle peut être [15] :

- **Régulière** : dans ces réseaux, tous les nœuds ont le même degré d.
- **Aléatoire** : les réseaux construits à partir de N nœuds ou un lien existe entre chacune des $\binom{N}{2}$ paires de nœuds est possible avec une probabilité *p*. Leur distribution de degrés est donnée par une distribution binomiale :

$$p_k = \binom{N}{k} p^{\,k}\ (1-p)^{N-k}\ \backsimeq\ \frac{(N_p)^k\ \ e^{N_p}}{k!}.$$

  Elle peut être approximée par une distribution de Poisson dans la limite $N \to \infty$.
- **Exponentielle** : certains réseaux réels de nombre important de nœuds ont une distribution de degrés qui suit approximativement une exponentielle **:**

$$p_k = (1 - e^{-1/k}\ \ )e^{-k/k}\ \ .$$

  Où k est un paramètre libre du système.
- **Loi de Puissance :** dans ces réseaux la distribution de degrés est donnée par une loi de puissance pure :

$$p_k \propto\ \ k^{-\alpha}\,.$$

  Ou une loi de puissance avec coupure exponentielle :

$$p_k = \frac{k^{-\alpha} e^{-k/k}}{Li_{\,\alpha}\left(e^{-k/k}\right)}\ pour k \geq 1.$$

  Ou $Li_{\,n}\ x$ est le n^{iem} polylogarithme de *x*.

Dans un réseau réel, bien que la plupart des nœuds aient un faible degré, il existe une probabilité non-négligeable qu'il existe des nœuds possédant un degré très élève. La présence de ces grands connecteurs a d'importantes répercussions sur la résilience des réseaux au retrait aléatoire des nœuds, ou sur la propagation de l'information ou d une épidémie [15].

2.1.3 Modélisation d'un réseau complexe

Pour étudier les réseaux complexes, des modèles qui permettent de générer des réseaux fidèles à ceux du monde réel ont été proposés en se basant sur les propriétés illustrées ci-dessus [15]. Nous citons ici les principaux modèles qui existent dans la littérature:

2.1.3.1 Le modèle de graphe aléatoire (Random)

Un réseau aléatoire est un réseau sans aucune organisation et dont le degré de connectivité est décrit par une variable aléatoire. Ce type de graphes a été introduit pour la première fois en 1959 par Erdös & Rényi. Pour générer un réseau aléatoire, on spécifie simplement le nombre total de nœuds N et la probabilité de connexion entre les nœuds p. La distribution de degré dans un réseau aléatoire est souvent donnée par une loi binomiale si le degré est constant moyen, ou par une loi de poisson si le degré est constant élevé. Le coefficient d'agrégation est $c=p$, et la distance moyenne est $l = \frac{\log N}{\log k}$ ou k est le degré moyen [14].

On note que, malgré le hasard qui semble régner dans de tels réseaux, il existe des résultats mathématiques intéressants qui y sont associés. En effet les graphes aléatoires jouent un rôle important dans l'évaluation des performances et de la complexité des algorithmes [8].

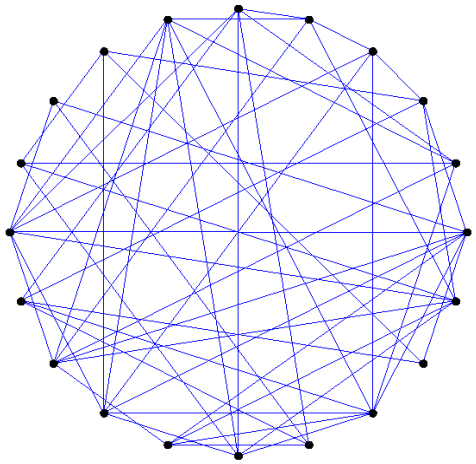


Figure 11. Graphe aléatoire.

2.1.3.2 Le modèle petit-monde (small-world)

Un réseau petit monde est un réseau caractérisé par :

- Une distance moyenne faible entre toutes paire de sommets.
- Un niveau de *clustering* local élevé, c'est-à-dire que les sommets sont généralement très connectés à leurs voisins immédiats.

Ce type de réseaux a été défini par Watts & Strogatz en 1998, en se basant sur deux mesures topologiques, à savoir : la distance géodésique moyenne et le coefficient de transitivité. Ils ont calculé ces mesures pour différents réseaux réels, et les ont comparées à celles des deux principaux modèles du moment, à savoir les graphes réguliers (complètement ordonnés) et les graphes aléatoires (complètement désordonnés). Les premiers possèdent une transitivité et une distance moyenne élevées, alors que pour les seconds, ces deux mesures sont petites. L'analyse a montré que les réseaux

réels se situent entre ces deux extrêmes. Leur distance moyenne est petite, du même ordre de grandeur que celle obtenue pour des réseaux aléatoires, et loin de celle des réseaux réguliers. Leur transitivité est élevée, du même ordre de grandeur que celle des réseaux réguliers, et loin de celle des graphes aléatoires [17].

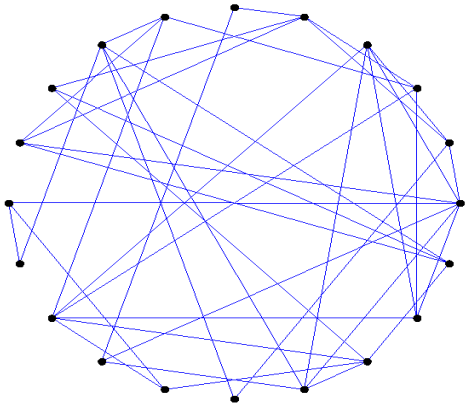


Figure 12. Réseau petit-monde.

2.1.3.3 Le modèle de réseau sans échelles (scale-free)

Un réseau sans-échelle est un réseau qui a la propriété sans échelle. Cette propriété a été observée par Barabási & Albert où le degré de nœuds mesuré sur plusieurs réseaux réels est distribué selon une loi de puissance. Les modèles classiques du moment n'expliquent pas cette distribution ; dans les graphes réguliers, le degré est par définition le même pour tous les nœuds c'est-à-dire la distribution régulière, alors que pour les graphes aléatoires, il est distribué selon une loi de poisson. Dans les deux cas, il existe donc une valeur caractéristique pour le degré, ce qui n'est pas le cas avec la loi de puissance.

La propriété sans-échelles a été observée sur des réseaux modélisants des systèmes réels très différents : grille électrique, Web, réseau d'interactions sociales, Internet, etc. Pour l'expliquer, Barabási & Albert utilisent deux mécanismes : la croissance du système et l'attachement préférentiel. Le premier implique que le réseau soit construit itérativement par addition de nouveaux nœuds. Le second stipule que ces nouveaux nœuds ont tendance à se connecter à des nœuds déjà existants avec un degré élevé. De ce fait, la propriété sans-échelles a un effet direct sur la structure du réseau : une grande majorité de nœuds est très faiblement connectée, alors qu'un nombre réduit possède un degré très élevé [17].

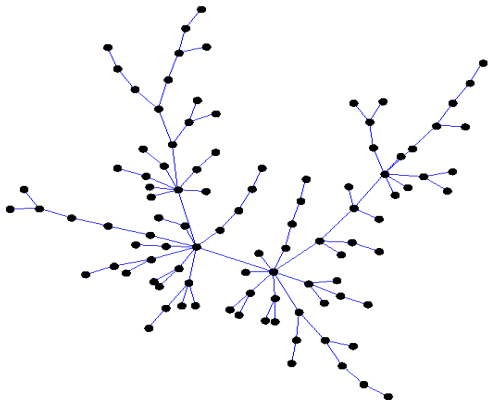


Figure 13. Réseau sans échelles.

2.2 Modèles de diffusion dans un réseau

2.2.1 Propagation dans un réseau

Le processus de propagation (diffusion) dans un réseau est la transmission d'une entité, qu'elle soit désirable ou non (exemple : maladie infectieuse, rumeur, virus informatique, publicité virale, idée, etc.) entre les nœuds voisins du réseau. Cette transmission provoque un changement d'état des nœuds [16]. À fin d'étudier le phénomène de diffusion et de mieux le comprendre, des modèles ont été créé pour reproduire le comportement d'une diffusion et décrire de façon formelle comment elle se déroule une propagation.

2.2.2 Modèle de diffusion

Un modèle de diffusion est généralement défini comme une représentation mathématique et/ou logique de la transmission dans un réseau et des processus y afférent. Ces modèles quantitatifs fournissent une représentation de la dynamique de transmission spatio-temporelle des entités entre les nœuds d'un réseau. La modélisation de la diffusion facilite l'évaluation de l'efficacité potentielle des mesures d'intervention prises, et fournit une estimation de l'étendue, de la durée et de l'extension géographique prévisible. Au titre d'exemple, dans le cas d'une épidémiologie, l'intérêt des modèles épidémiologiques réside dans leur capacité à étudier des scénarios hypothétiques et à fournir aux décideurs des éléments permettant d'anticiper les conséquences de l'incursion d'une infection et l'impact des stratégies d'intervention [18].

On emprunte généralement le vocabulaire de la modélisation épidémiologique pour décrire un processus de propagation. Ainsi, on appelle *infection* le facteur pouvant se propager dans le réseau. On dit qu'un nœud est *Infectieux* s'il porte l'infection et peut la transmettre; un nœud ne portant pas l'infection mais pouvant l'acquérir est dit *Susceptible* ; et un nœud ne portant pas l'infection ni ne pouvant l'acquérir est dit *Retiré*. Ces caractéristiques intrinsèques aux nœuds sont appelées états

épidémiologiques [16]. Nous présentons dans ce qui suit les différents modèles de diffusion existent dans la littérature :

2.2.2.1 Modèle SI (Susceptible/Infected)

Dans le modèle *SI*, les nœuds d'une population sont dans l'un des états suivants: Susceptible (état S) ou infecté (état I). Chaque nœud *x*, a pour chaque instant, une probabilité *p* d'être contaminé. Une fois qu'un nœud est contaminé, il reste dans cet état, et ne peut pas guérir.

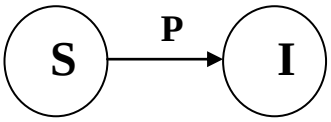


Figure 14. Modèle SI (Susceptible/Infected).

On considère que la probabilité qu'un individu susceptible devienne infecté est proportionnelle au nombre d'individus actuellement infectés et que le coefficient de proportionnalité soit $i \geq 0$. Si un grand nombre d'individus est en cause, on peut s'attendre à ce que la probabilité qu'un individu devient infecté est $i \times$ (le nombre des individués déjà infectés). Sous forme d'′equations différentielles, ceci devient [19]:

$$\frac{dS}{dt} = -i \times (\text{le nombre des individués déjà infectés}).$$

$$\frac{dI}{dt} = i \times (\text{le nombre des individués déjà infectés}) .$$

Ce modèle a rapidement été étendu. En effet, pour représenter un épiderme, il n'est pas réaliste de considérer que les nœuds infectés le restent indéfiniment. Il se peut qu'ils soient détruits ou qu'ils guérissent. Dans le cas où ils guérissent, il y a deux possibilités : les nœuds guéris peuvent devenir immunisées à l'épiderme, ou bien avoir à nouveau le même risque que les autres d'être contaminées. De ce fait, des nouvelles variations de ce modèle ont été proposées, les plus utilise sont le modèle SIS et SIR [20].

2.2.2.2 Modèle SIS (Susceptible/Infected/Susceptible)

Dans le modèle SIS, les nœuds d'une population sont dans l'un des états suivants : Susceptible (état S) ou Infecté (état I). Chaque nœud *x*, a pour chaque instant, une probabilité *p* d'être contaminé. Une fois qu'un nœud est contaminé, il a une probabilité *p'* de guérir et devenu à nouveau susceptible.

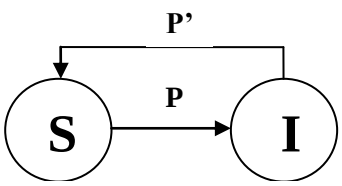


Figure 15. Modèle SIS (Susceptible/Infected/Susceptible) .

Si on considère que pour une instance *t*, il y a gI individus guérissant, le système d'´equations devient donc [19]:

$$\frac{dS}{dt} = -i \times (\text{le nombre des individués déjà infectés}) + \text{gI} .$$

$$\frac{dI}{dt} = i \times (\text{le nombre des individués déjà infectés}) - \text{gI} .$$

2.2.2.3 Modèle SIR (Susceptible/Infected/Recovered)

Le modèle SIR (Susceptible/Infected/Recovered) divise la population en trois catégories: les nœuds susceptibles de se faire infecter (état S), les nœuds infectés et contagieux (état I), et les nœuds ne pouvant plus transmettre la maladie guérissant avec immunité (état R).

La contamination se fait sur le même principe que le modèle SI, à la différence qu'un nœud immunisé ne peut pas être infecté. De plus, les nœuds infectés peuvent être guéris et passent dans un état immunisé. Chaque nœud *x*, a pour chaque instant, une probabilité *p* d'être contaminé, une fois contaminé, il y a une probabilité *p'* d'être guéri.

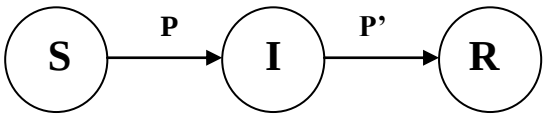


Figure 16. Modèle SIR (Susceptible/Infected/Recovered).

Ce modèle est généralement formulé ainsi [16]:

$$\frac{dS(t)}{dt} = -\lambda S(t)\frac{I(t)}{I(t)+S(t)+I(t)} .$$

$$\frac{dI(t)}{dt} = \lambda S(t)\frac{I(t)}{I(t)+S(t)+I(t)} - \alpha I(t) .$$

$$\frac{dR(t)}{dt} = \alpha I(t) .$$

Le taux de retrait α fixe la probabilité αdt qu'un élément infectieux devienne retiré pendant un intervalle de temps$[t, t+dt]$. Par conséquent, chaque élément devenant infectieux a une probabilité de ($\alpha^{-\alpha\tau}\ d\tau$) reste infectieux pendant une période de temps comprise entre $[\tau, \tau+d\tau]$. Lorsqu'une distribution plus concentrée autour de sa moyenne est désirée, une solution fréquente consiste à discerner plusieurs stades infectieux: on parle alors d'un modèle SI^nR permettant les états {S, I_1, I_2, …. , I_n,R},et la probabilité de passage d'un état infectieux au suivant est de $n\alpha dt$ pour l'intervalle de temps $[t, t+dt]$.

Le taux d'infection λ détermine à la fois la virulence de l'infection et la densité de la population étudiée. Afin d'isoler ces effets, on décompose λ en deux paramètres tels que $\lambda = \beta k$. Ainsi, k donne le nombre moyen d'interactions par élément alors que β, également appelé taux d'infection, donne le nombre moyen βdt de nouvelles infections par interaction entre un élément susceptible et un élément infectieux pendant un intervalle de temps $[t, t + dt]$. Le terme $\frac{I(t)}{I(t)+S(t)+I(t)}$ détermine la fraction des kSt interactions des éléments susceptibles qui sont faites avec des éléments infectieux.

En termes de réseaux, la structure du système est régie par un modèle d'Erdôs et Rényi canonique [16]: $N = S(t) + I(t) + R(t)$ (N constant), où chaque lien a une probabilité $p = \frac{k}{N-1}$ d'exister. Dans ce contexte, un nœud susceptible possédant k_I liens vers des nœuds infectieux a une probabilité $k_I \beta dt$ de devenir infectieux pendant un intervalle de temps $[t, t + dt]$. Aucune adaptation particulière n'est nécessaire pour le taux de retrait : un nœud infectieux a une probabilité αdt de devenir retiré pendant un intervalle de temps $[t, t + dt]$.

Conclusion

L'étude des phénomènes du monde réel devient de plus en plus une nécessité. Tout réseau réel est un réseau complexe, et l'un des phénomènes important dans ce type de réseaux est la diffusion. Dans ce chapitre, nous avons présenté, dans une première partie, les différentes caractéristiques des réseaux complexes. Ensuite, nous nous sommes focalises sur le problème de diffusion dans ces réseaux, tout en présentant les différents modèles qui existent.

Chapitre 3

Le problème de détection de nœuds critiques (CNDP)

Introduction

Plusieurs mécanismes et fonctions dans les réseaux, tel que la propagation et la synchronisation sont influencées par un sous ensemble de nœuds qui sont, généralement, qualifiés de nœuds importants, tout dépend de leurs rôles dans le réseau. Identifier les nœuds importants d'un réseau permet de bien comprendre leurs propriétés structurelles et fonctionnelles. D'un point de vu connexité, les nœuds importants sont ceux qui maintiennent la connectivité du réseau, et ainsi leur suppression déconnecte le réseau. Ces nœuds sont connus dans la littérature par les « nœuds critiques », et le problème qui les étudie est le « Problème de détection de nœuds critiques dans les réseaux », en anglais ***Critical Nodes Detection Problem (CNDP).***

Dans ce chapitre nous allons introduire les différentes variantes du problème *CNDP* tout en discutant leur complexité et leurs différentes approches de résolution, ainsi nous allons présenter leurs différents domaines d'application.

3.1 Problème de détection de nœuds critiques (CNDP)

Le problème de détection de nœuds critiques *CNDP* est un problème d'optimisation qui, étant donné un réseau, consiste à trouver un ensemble de nœuds de taille minimale dont la suppression provoque une fragmentation maximale du réseau. Pour évaluer la déconnexion dans le réseau après la suppression des nœuds, différentes métriques ont été considérées dans la littérature, y compris : minimiser le nombre de composantes connexes, minimiser la connectivité de paires de nœuds dans le réseau, maximiser le nombre de composantes connexes, minimiser la taille de composantes, etc. [21] [22].

3.2 Le nœud critique ?

Dans un réseau, un nœud critique est celui qui maintient la cohérence du réseau et dont la suppression dégradera sa connectivité [22]. Cette dégradation est évaluée par une métrique à satisfaire par le réseau induit, et la solution au problème doit optimiser la métrique considérée. Le tableau 2 présente quelques métriques ainsi que la solution correspondante considérons le graphe de la figure 19. Sous la condition qu'un seul nœud est à supprimer.

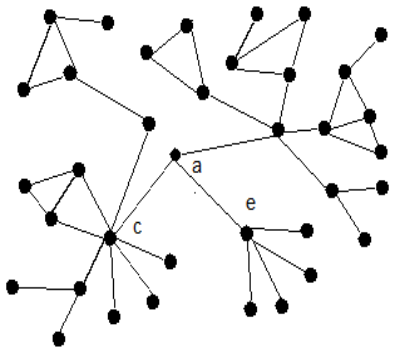


Figure 17. Un graphe pour illustre la solution optimale pour quelques métriques de connectivité.

Maximiser le nombre des composantes	Le nœud à supprimer {c}	génère sept composantes
minimiser la taille des composantes	Le nœud à supprimer {a}	génère des composantes de taille maximale de (de seize nœuds)
Maximiser le nombre de composantes de taille <2	Le nœud à supprimer {e}	Produit quatre composants de taille <2

Tableau 2. Des solutions optimales pour des métriques considérées dans le graphe de la figure 19.

On remarque que la solution optimale du problème dépond de la métrique considérée. Ainsi tout déponds de cette métrique, plusieurs variantes de CNDP ont été proposées dans la littérature.

3.3 Différentes variantes de CNDP

Dans le cas général, *CNDP* cherche un ensemble de nœuds dans un graphe dont la suppression minimise ou maximise une métrique prédéfinie. Donc, on a en entrée un graphe $G = (V; E)$ et une métrique de connectivité nommée δ, et on sortie un ensemble de nœuds S ⊆ V tel que *G[V\S]* optimise la métrique δ. Généralement, on a deux cas [22]:

i) Optimiser la métrique δ de telle sorte que le nombre des nœuds à supprimer ne dépasse pas k nœuds ($|S| \leq k$).
ii) Minimiser le nombre de nœuds à supprimer, de telle sorte que la métrique δ soit bornée par une borne donnée β.

La classe des variantes du premier cas est applicable quand nous avons des informations sur le nombre de nœuds à identifier mais nous ne savons pas le moyen d'atteindre notre objectif de façon optimale. Ceci est le cas de, par exemple, la destruction d'une organisation terroriste, nous essayons de neutraliser la communication entre ses membres en éliminant certains d'entre eux, et en raison des

ressources limitées, nous ne pouvons neutraliser que K membre tout en maximisant la fragmentation du réseau de communication en utilisant les ressources possibles afin d'atteindre notre objectif. Cependant, si nous cherchons à neutraliser totalement le réseau terroriste, et nous savons qu'au moins ***t*** personnes doivent communiquer pour prendre une décision, donc pour atteindre notre objectif, nous essayons d'éliminer un sous ensemble de nœuds tels que le réseau devient composé de sous ensembles d'au maximum ***t-1*** membres. Dans un tel cas, nous ne savons pas le nombre de nœuds à cibler, mais nous avons des informations sur la structure du réseau que nous devions obtenir, ainsi c'est dans une telle situation que les variantes du deuxième cas sont applicable.

On note que la métrique $\boldsymbol{\delta}$ sert à décrire comment la structure du réseau doit être déconnectée après la suppression de nœuds critiques [22].

Dans ce qui suit, nous allons détailler chacune des variantes présentées dans la littérature :

3.3.1 CNP: Critical Node Problem

Dans cette variante, la métrique considérée est la minimisation de la connectivité par paires dans le réseau. Donc, on cherche à trouver un ensemble $S \subseteq V$ d'au maximum *k* nœud, dont la suppression minimise la connectivité par paires dans le graphe induit $G[V/S]$. L'utilisation de cette métrique permet de, simultanément, maximiser le nombre de composantes connexes et de minimiser la variance de cardinalité entre les différentes composantes dans $G[V/S]$. Ceci peut être formulé en utilisant la fonction suivante:

$$f(V \setminus S) = \sum_{i \in C} \frac{\delta i(\delta i-1)}{2}.$$

Ou *C= {C1, C2, C3, ... Cm}* est l'ensemble de toutes les composantes connexes dans $G[V \setminus S]$ après la suppression des k nœuds, et δi représente la taille de la composante Ci. Ainsi, la version décisionnelle de CNP est [23] [22] :

Entrée: un graphe non orienté $G = (V; E)$ et un entier k.

Sortie: S=arg min$\sum_{i \in c} \frac{\delta i(\delta i-1)}{2}$ ou $|S| \leq k$.

3.3.2 MaxNum - Maximize the Number of connected components

Dans cette variante la métrique considérée est la maximisation de nombre de composantes connexes. Donc, on cherche à supprimer un nombre déterminé de nœuds d'un graphe afin de déconnecter les nœuds du graphe autant que possible tout en maximisant le nombre de composantes. La version décisionnelle de *MaxNum* peut être prescrite comme suit [24] [22]:

Entrée: un graphe $G = (V ; E)$ non orienté et un entier$k$.

Sortie : $S = \arg\max \rho, |S| \leq k$ ou ρ est le nombre de composants connecté dans $G[V/S]$.

3.3.3 MinMaxC - Minimize the largest connected component size

Dans cette variante, la métrique considérée est la minimisation de la taille de composantes connexes. *MiniMax* cherche l'ensemble de k nœuds dont la suppression minimise la taille de la plus grande composante connexe dans le graphe induit. La formulation de cette variante peut être prescrite comme suit [25] [22] :

Entrée: un graphe $G = (V ; E)$ non orienté et un entier k.

Sortie:$S = \arg\min \delta_h$,$|S| \leq k$ ou h est la plus grande composante connexe dans $G[V /S]$.

3.3.4 CC-CNP: Cardinality Constrained Critical Node Problem

CC-CNP cherche à trouver un ensemble de nœuds S ⊆ V avec une cardinalité minimum, de telle sorte que les indices de connectivité des nœuds dans le graphe induit G[V\S] sont inférieurs à un seuil donné β. L'indice de connectivité d'un nœud est défini comme étant le nombre de nœuds accessibles à partir de ce nœud. La formulation générale du problème peut être prescrite comme suit [26] [22]:

Entrée: un graphe $G = (V ; E)$ non orienté et un entier β.

Sortie : $S = \arg\min, |S|$, ou $\sum_{i,j\in V\setminus S} u_{ij} \leq \beta$

$$u_{ij} = \begin{cases} 1 \text{ si i et j sont dans le même composant de } G = (V ; E) \\ 0 \text{ sinon} \end{cases}.$$

3.3.5 β-vertex Disruptor Problem

La variante β -*vertex disrupteur* cherche à réduire au minimum le nombre de nœuds à supprimer telle que la connectivité de l'ensemble de paires de nœuds connectés ne dépasse pas un seuil donné. Ce seuil est défini par une fraction donnée β de la connectivité par paires de tout le réseau. On note que l'utilisation de β comme un paramètre d'entrée permet de traiter différents niveaux de connectivité dans le graphe induit. Le problème peut donc être formulé comme suit [27] [22]:

Entrée: un graphe $G = (V ; E)$ non orienté avec $|V| = n$ et un entier$0 \leq \beta \leq 1$.

Sortie: Un ensemble minimal de nœuds $S \subseteq V$ dont la suppression contraintes la connectivité de la paire dans $G[V /S]$ pour $\beta\binom{n}{2}$.

3.3.6 3C-CNP: Component-Cardinality-Constrained CNP

3C-CNP cherche à minimiser l'ensemble de nœuds à supprimer tout en limitant la taille de chaque composante connexe dans le graphe induit à une borne donnée β. Sa formulation peut être énoncée comme suit [22]:

Entrée: un graphe $G = (V ; E)$ non orienté avec un entierβ.

Sortie: Un ensemble minimal de nœuds $S \subseteq V$ ou $\delta_h \leq \beta$ pour toute composante $h \in G[V/S]$.

3.4 Complexité du CNDP

Nous avons défini précédemment la complexité algorithmique d'un problème donné comme une estimation du nombre d'instructions à exécuter pour résoudre les instances de ce problème. La fonction de complexité d'un algorithme fait correspondre pour une taille donnée le nombre maximum d'instructions qui lui est nécessaires pour résoudre une instance quelconque de cette taille. Nous avons aussi vu que les problèmes dont la complexité est exponentielle sont des problèmes *NP-complets*. Concernant *CNDP*, toutes les variantes sont *NP-complets* dans le cas d'un graphe général, et certaines d'eux restent *NP-complet* même dans le cas de classes de graphes particuliers. En dépit de sa difficulté, *CNDP* a bien été étudié sur différentes classes de graphe. De nombreuses approches ont été explorées afin de le résoudre en fournissant des solutions exactes à l'aide de la programmation dynamique [24], et la programmation linéaire en nombre entier [25], ou des solutions approchées en utilisant des algorithmes heuristiques, des algorithmes d'approximation polynomial et des algorithmes de recherche stochastiques [28].

3.5 Approches de résolution de CNDP

Pour résoudre *CNDP*, nous visons à distinguer les nœuds critiques de ceux non-critiques. A première vu, il semble facile d'identifier ces nœuds en utilisant un algorithme gluten et une des mesures utilisées pour identifier les nœuds importants d'un réseau, tels que la centralité. Ainsi, à chaque itération nous sélectionnons le nœud central comme nœud critique. Cependant, une telle approche échoue souvent à identifier avec précision les nœuds critiques pour deux raisons principales:

- Si la mesure choisie n'est pas conçue pour distinguer les nœuds importants en termes de connectivité. Par conséquent, cette mesure n'est pas appropriée pour identifier les nœuds critiques [23]. Ceci est le cas, par exemple, de la centralité de degré. Afin d'illustrer ce point, nous considérons le graphe de la figure 18. Nous rappelons que la centralité de degré choisit le nœud de degré maximal. Nous supposons que notre objectif est de maximiser le nombre de composantes connexes dans le graphe en supprimant un nœud. Donc, supprimer le nœud ***a,*** qui est le nœud central, n'a aucun impact sur la connectivité du graphe, alors que supprimer ***b***, qui a un degré de centralité inférieure, génère un graphe en trois composantes.

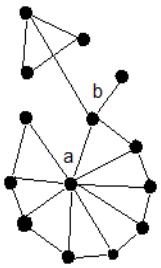


Figure 18 . Contre exemple de l'approche de centralité.

- Dans *CNDP,* on cherche généralement un ensemble de nœuds dont qui, ensemble, influent sur la connectivité du réseau. Donc, un ensemble de nœuds qui influent sur la connectivité quand ils sont considérés individuellement n'influent pas nécessairement par la même envergure quand ils sont ensemble, et dans notre algorithme gluten les nœuds critiques sont choisis individuellement chaque itération. Pour illustrer ce point, nous considérons le graphe de la figure 19**.** Nous supposons que notre objective est de maximiser le nombre de composantes connexes par la suppression au maximum deux nœuds. La suppression des nœuds d'articulation semble un bon choix puisque l'objective est de maximiser le nombre de composants connectés et les nœuds d'articulation sont les nœuds d'interconnexion des composantes du graphe. Cependant, nous pouvons remarquer que l'impact global lors de la suppression des nœuds ***{d, e}***, qui ne sont pas tous deux des nœuds d'articulation, est meilleur que la suppression des nœuds ***{c, d}***, qui sont tous des nœuds d'articulation [22].

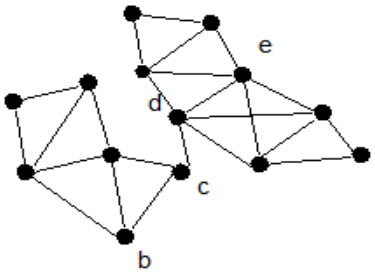


Figure 19. Contre exemple de l'approche des points d'articulation.

3.6 Applications de CNDP

L'identification des nœuds critiques d'un réseau permet d'analyser et de comprendre les propriétés structurelles de ce réseau, ce qui simplifie son contrôle que ce soit l'objective est de le protéger ou bien le détruire. Les deux points majeurs qui font que *CNDP* soit un paramètre important dans le domaine des réseaux sont [22] :

- Les applications réseau sont généralement conçues pour s'exécuter dans un environnement connecté, et *CNDP* est le problème qui s'occupe de la détermination des nœuds dont la préservation fournit un tel environnement, et dont la suppression le détruit. Ainsi, l'identification de ces nœuds est très utile pour étudier les applications réseau avant et après la conception [22].
- *CNDP* est un paramètre à double tranchant. En effet, il peut être utilisé pour des perspectives offensives ou défensives, selon l'objectif de l'application à réaliser. Par exemple, pour un administrateur réseau informatique, les nœuds critiques sont ceux qu'il faut protéger contre les attaques virus afin d'assurer la connectivité dans le réseau (défensive), ou ce sont ceux qu'il faut cibler pour détruire un réseau adverse (offensive). Ainsi, les nœuds critiques peuvent être utilisés dans des applications de protection et de surveillance défensive pour des fins positive, ou dans des attaques et des tentatives offensives pour des fins négatives [22].

Donc l'identification de nœuds critiques, en tant que moyen efficace pour l'analyse des réseaux, a été appliquée dans de nombreux domaines pour différents fins, notamment pour: la vaccination de populations, l'évaluation de la vulnérabilité d'un réseau, l'analyse des réseaux sociaux, sécurité des réseaux de télécommunication, l'étude des organismes biologiques, etc. Dans ce qui suit, nous présentons les différentes applications de *CNDP* classifiées par domaine:

3.6.1 Etude biologique

En biologie, la structure des protéines et la nature de leurs interactions sont largement étudiées. Des organismes biologiques, tels que les bactéries et les virus, sont constitués d'un ensemble de protéines en interaction entre eux formant un réseau, qu'on peut représenter par un graphe où les nœuds sont les protéines et les arêtes sont les interactions entre eux. Identifier les nœuds critiques qui maintiennent la connexion dans ces réseaux peut fournir des informations utiles pour de nombreuses applications biologiques. Par exemple, d'un point de vu conception de médicaments, les protéines critiques sont ceux qu'on doit détruire pour neutraliser l'organisme nuisible, c'est ce qui a été fait pour détruire les cellules cancéreuses [29].

3.6.2 Évaluation de la vulnérabilité des réseaux

Un réseau, quel que soit son type, peut être vulnérable à plusieurs événements perturbateurs tels que les catastrophes, et l'évaluation de sa vulnérabilité est un factor important pour sa bonne gestion. En général, les points faibles d'un réseau sont ses nœuds critiques, car leur destruction provoque une dégradation maximale de ses performances. L'évaluation de la vulnérabilité d'un réseau en utilisant les nœuds critiques permet d'explorer ses vraies faiblesses, et par conséquent, le protéger contre tout événement perturbateur [22] [30].

3.6.3 Réseaux de communication.

Dans les réseaux de télécommunication, les nœuds critiques sont ceux qui doivent être détruits pour désactiver la possibilité de communication dans le réseau adversaire, et sont ceux qu'il faut préserver pour d'éviter toute tentative de neutralisation de son propre réseau de communication. Par exemple, dans le domaine militaire, une des techniques de destruction du réseau de communication de l'ennemi est de faire le brouillage pour les appareils électroniques qui assurent la communication qui sont des points critiques du réseau [31].

3.6.4 Analyse des réseaux sociaux.

Dans les réseaux sociaux, les individus sont en interactions entre eux. Pour un réseau social, les chercheurs sont intéressés à étudier à la fois la structure du réseau ainsi que le comportement de phénomènes sociaux. Généralement, l'étude de la structure des réseaux permet de mieux comprendre les phénomènes émergents. L'un des phénomènes qui a attiré plus d'attention récemment est la propagation d'une entité au sein du réseau. Il a été étudié pour différents objectifs, à savoir : la localisation de la source, le blocage de la propagation, etc. Pour étudier ce phénomène, plusieurs paramètres ont été utilisés, à savoir : la centralité avec ses différentes variantes, les keys players, les communautés, etc. Un des paramètres qui a récemment montré son efficacité dans l'étude de ce phénomène est le CNDP.

En effet, si l'objective est de bloquer la diffusion, on peut bien utiliser 3C-CNP, qui garantit que toute interaction est détectée après au plus L saut dans le réseau, où L est la contrainte sur la cardinalité de la composante connexe dans la variante 3C-CNP. En outre, si l'objectif est de minimiser temps de détection de la source de diffusion, cette variante permet de limiter l'espace de recherche et d'isoler le nœud source dans une composante qui ne dépasse pas L nœuds [22].

3.6.5 Vaccination et contrôle des maladies contagieuses.

Dans les réseaux sociaux, l'étude de la propagation de la contamination a attiré l'attention des scientifiques ces dernières années. Généralement, on cherche à identifier seulement quelques

personnes dont la vaccination empêche la propagation de la contamination. Donc, on peut identifier les nœuds critiques d'un réseau social comme personne à vacciner, et ainsi l'utilisation de CNDP constitue une stratégie efficace pour stopper la propagation des épidémies tout en utilisant un nombre limité de vaccins [22].

3.6.6 Problèmes de sécurité.

Protéger les applications contre les comportements malveillants est l'une des principales préoccupations dans la conception d'une application réseau. Ainsi, la conception doit tenir on compte des faiblesses du réseau qu'un attaquant peut exploiter. Généralement, la conception des applications réseaux est basée sur la sélection d'un ensemble de nœuds appelé « noyau », tel que l'ensemble stable ou l'ensemble dominant. Ainsi, leur sécurité dépond de la sécurité de ce noyau. Partant du principe que plus il y a des nœuds critiques dans le noyau, plus l'application est vulnérable, et inversement, moins il y a de nœuds critiques, plus l'application est robuste ; le *CNDP* est un bon paramètre à prendre en considération lors de la conception des applications réseaux sécurisées [22].

Conclusion

Dans ce chapitre, nous avons détaillé le problème de détection de nœuds critiques dans les réseaux, tout en illustrant sa difficulté via des exemples. Nous avons aussi présenté les différentes variantes qui existent dans la littérature, et pour chacune des variantes, nous avons donné une formulation. À la fin du chapitre, nous avons parlé des différentes applications de ce paramètre dans différents domaines de la science.

Chapitre 4

Réalisation

Introduction

Le phénomène de diffusion est le résultat d'interaction d'un ensemble d'entités. Pour comprendre le phénomène de diffusion, il faut faire face à tous ses variables. Cela implique de proposer des hypothèses sur le fonctionnement de ses entités et d'observer leurs comportements en interaction. Cependant, il est difficile de reproduire dans les conditions d'expérimentation classique ce phénomène pour tester nos hypothèses. La seule démarche qui permettre de le faire n'est d'autre que la simulation.

Ces dernières décennies, les simulations numériques sont devenues un outil privilégié d'investigation dans différents domaines de la science, et le besoin de simuler par ordinateur des réalisations de plus en plus complexe devient de plus en plus croissant.

Dans ce chapitre, nous allons présenter la partie réalisation et mise en œuvre de notre travail, qui consiste au développement d'un outil permettant l'analyse des réseaux et l'étude du phénomène de diffusion. L'outil implémente, d'une part, les algorithmes de différents paramètres nécessaires pour extraire les propriétés du réseau à étudier, ainsi que les différents algorithmes de détection de nœuds critiques. D'autre part, il implémente les différents modèles de diffusion, et permet d'illustrer le blocage de la propagation dans le réseau utilisant les nœuds critiques.

Nous présentons dans une première partie les principes généraux de la simulation. Ensuite, nous décrivons notre environnement de travail et les différents outils de développement utilisés. Enfin, nous détaillons les différents algorithmes implémentés par l'application, et nous incluons quelques interfaces pour expliquer son fonctionnement.

4.1 Principe généraux de la simulation

Etymologiquement, le terme ‘‘simulation’’ est dérivé du mot latin « *SIMULARE* » qui veut dire: copier, feindre, faire paraître comme réelle une chose qui ne l'est point. Ainsi, la simulation peut être définie comme suit :

4.1.1 C'est quoi la simulation ?

La simulation est l'expérimentation sur un modèle. C'est une procédure de recherche scientifique qui consiste à réaliser une reproduction artificielle (modèle) du phénomène que l'on désire étudier, à observer le comportement de cette reproduction lorsque l'on fait varier expérimentalement les actions que l'on peut exercer sur celle-ci, et à en induire ce qui se passerait dans la réalité sous l'influence d'actions analogues [32].

On peut donc dire que simuler le fonctionnement d'un système c'est imiter son fonctionnement au cours du temps en manipulant un modèle. Ceci est équivalent à la génération d'un historique des changements d'état du système, et à l'observation de cet historique pour faire des déductions sur les caractéristiques de fonctionnement du système. C'est donc une méthodologie essentiellement pratique qui permet de modéliser aussi bien des systèmes conceptuels (qu'on veut concevoir) que des systèmes existants déjà. Elle peut être utilisée pour [33]:

- Décrire et analyser la dynamique d'un système.
- Répondre aux questions de type «What If ? » sur le système réel.
- Aider à la conception d'un système réel.

4.1.2 Quand simuler ?

Cette question est liée aux caractéristiques du problème lui-même et au choix d'une méthode pour le résoudre. En général, il y a deux voies pour résoudre un problème: analytiquement et expérimentalement. Les méthodes analytiques donnent souvent une solution optimale, alors que les méthodes expérimentales mènent souvent à une bonne solution, mais pas nécessairement à la solution optimale. Ainsi, la simulation peut être vue comme la conduite d'une expérimentation indirecte (sur le modèle et non sur le système) [34]. Elle est souvent caractérisée comme la méthode de dernier ressort, ainsi si on a la possibilité de résoudre le problème posé avec des méthodes analytiques, il serait préférable de les utiliser car elles conduisent à des solutions optimales.

La simulation d'un système réel devient nécessaire dès que les modèles analytiques deviennent, soit trop complexes en termes de calcul et de temps de résolution, soit trop simplifiés vis-à-vis de la réalité rendant, de ce fait, les résultats obtenus ne sont pas représentatifs du comportement du système dans un environnement réel. Généralement, il est nécessaire de simuler dans les cas suivants [34]:

- Les expériences sur le système réel sont trop coûteuses en termes de ressources matérielles et humaines.
- Les expériences sur le système réel ne sont pas reproductibles ni représentatives de tous les environnements possibles. Dans ce cas, la simulation permet de caractériser le comportement global du système dans différents environnements.

4.1.3 Pourquoi simuler ?

Généralement, la simulation assure :

- Une évaluation des performances du system.
- Un contrôle ou une optimisation du système : estimer l'influence des facteurs les plus significatifs pour garantir les performances exigées du système.

- Une comparaison des scenarios : comparer différentes conceptions, stratégies, politiques et règles de fonctionnement.
- L'apprentissage : la simulation sert très souvent à l'apprentissage dans un environnement virtuel [35].

4.1.4 Différent types de simulation.

Généralement en désigne deux type de simulation.

4.1.4.1 La simulation continue.

La simulation continue est utilisée dans les cas où le système se présente sous la forme d'un ensemble d'équations différentielles à résoudre. Elle permet de suppléer à la résolution analytique quand celle-ci est impossible [36].

4.1.4.2 La simulation discrète.

La simulation à événements discrets est une technique utilisée dans le cadre de l'étude de la dynamique des systèmes. Une simulation à événements discrets est une modélisation informatique où le changement de l'état d'un système, au cours du temps, est une suite d'événements discrets. Chaque événement arrive à un instant donné et modifie l'état du système. Elle se divise en trios grandes catégories :

- *Asynchrone (time-slicing)* : on simule à chaque fois le passage d'une unité de temps sur tout les système.
- *Synchrone (event-sequencing)* : on calcul l'arrivée du prochain événement, et on ne simule qu'événement par événement, ce qui permet souvent des simulations rapides, bien qu'un peu plus complexe à programmer.
- *La simulation par agents* : la simulation est segmentée en différentes entités qui interagissent entre elles. Elle est surtout utilisée dans les simulations économiques et sociales, où chaque agent représente un individu ou un groupe d'individus. Par nature, son fonctionnement est asynchrone [36].

4.1.5 Etapes de la simulation

Les étapes qui constituent tout projet de simulation peuvent être schématisées par l'organigramme ci-dessous [34] :

:

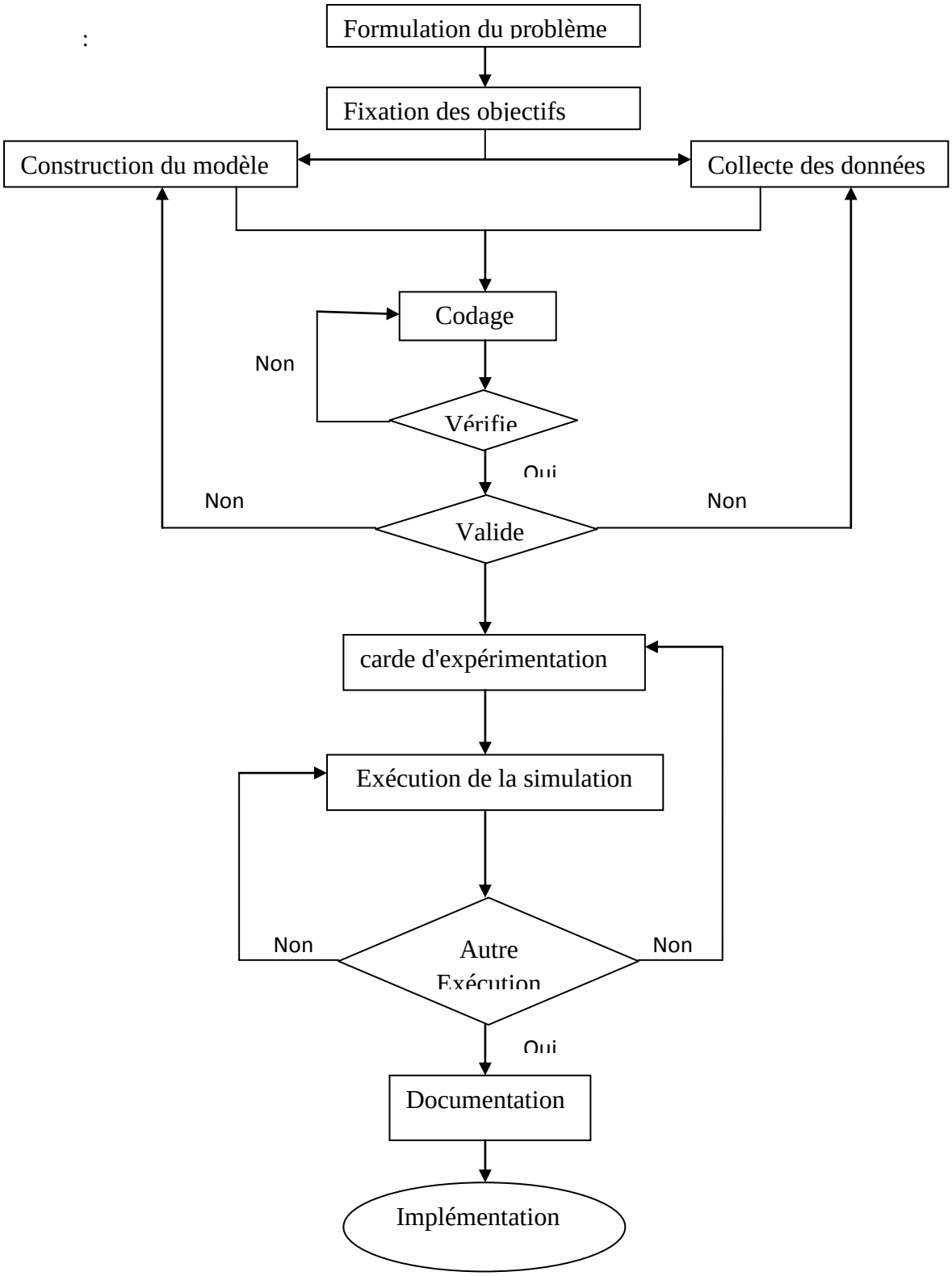


Figure 20. Etapes de la simulation

- ***Formulation du problème***

Tout projet de simulation commence d'abord par l'énoncé du problème à résoudre [34].

- ***Fixation des objectifs***

Une fois le problème formulé, il faudra définir les objectifs visés par le projet de simulation. Ceci comprend : les questions auxquelles l'étude par simulation qu'on veut mener devra apporter une réponse, le personnel, le matériel et les logiciels informatiques requis, les divers scénarios qu'on veut investiguer, les coûts de l'étude ainsi que les temps requis, etc. [34].

- ***Construction du modèle***

Il s'agit de construire un modèle conceptuel qui est une abstraction du système réel. Ce modèle peut être vu comme un ensemble de relations mathématiques et logiques concernant les composants et la structure du système [34].

- ***Collecte des données***

Une fois le problème formulé et les objectifs visés identifiés, il faudra établir un inventaire des besoins en données sur le système réel [34].

- ***Codage***

Il s'agit de traduire le modèle conceptuel obtenu à l'étape 3 dans une forme acceptable par l'ordinateur (i.e. programme, appelé aussi modèle opérationnel) [34].

- ***Vérification***

L'étape de vérification est extrêmement importante dans tout projet de simulation. Elle concerne le modèle opérationnel (le programme). Il s'agit de s'assurer que le modèle s'exécute sans erreurs. La vérification est primordiale même pour des modèles de taille réduite car ces derniers peuvent aussi comporter des erreurs [34].

- ***Validation***

La validation consiste à s'assurer que le modèle conceptuel est une représentation fidèle du système réel. Il s'agit en fait de savoir si le modèle peut être substitué au système réel pour le but de l'expérimentation. Dans le cas ou le système existe, la façon idéale de valider le modèle conceptuel est de comparer ses sorties avec celles du système. Malheureusement, on n'a pas toujours cette possibilité surtout dans les projets de conception de nouveaux systèmes [34].

- ***Conception d'un cadre d'expérimentation***

Il s'agit de définir pour chaque scénario devant être simulé ou expérimenté un certain nombre de paramètres tels que : la durée de la simulation, le nombre de simulations à faire (réplications), l'état initial du modèle et les règles de gestion des files d'attente. Le programme soir fournir la possibilité de séparer le cadre d'expérimentation qui contient toutes les données et les informations pour exécuter la simulation. Ainsi, il sera possible de faire différentes expérimentations sur le même modèle en changeant uniquement le cadre d'expérimentation [34].

- ***Exécution de la simulation et analyse des résultats***

Le modèle opérationnel (ou le programme) est le support principal pour réaliser une simulation sur ordinateur. Il délivre en sortie des résultats purement statistiques (moyenne, minimum, maximum,...). L'analyse de ces résultats aura pour objectifs d'estimer les mesures de performances des scénarios qu'on a expérimentés [34].

- ***Exécutions supplémentaires***

A ce niveau, on dispose d'un ensemble de résultats provenant des différentes simulations qu'on a réalisé ainsi que d'une analyse de ces résultats. Il s'agira de déterminer sur la base de cette analyse si d'autres simulations doivent être faites, si d'autres scénarios non prévus doivent être expérimentés afin de s'assurer que le modèle répond bien aux objectifs visés dans l'étape 2 [34].

- ***Documentation***

La documentation est nécessaire, et elle concerne aussi bien le modèle que les résultats de la simulation. Si le modèle aura un jour à être réutilisé par d'autres personnes, la documentation les aidera à comprendre le fonctionnement du modèle et leur facilite toutes modifications ou mises à jour du modèle [34].

- ***Implémentation***

L'objectif de toute simulation est de proposer pour un problème plusieurs solutions. Le choix de la meilleure solution devra être fait par l'analyste qui la justifiera dans la documentation et la propose au client. La décision de retenir cette solution pour une éventuelle implémentation reste donc une responsabilité du client [34].

4.1.6 Avantages et inconvénients de la simulation

La simulation est une technique de modélisation du monde réel. Comme toute approche scientifique la simulation présente des avantages et des inconvénients :

- ***Avantage.***

Elle permet de :

- représenter le fonctionnement d'un système composé de différent centre d'activité.
- mettre en évidence les caractéristiques les interactions entre les différentes composantes du système, et décrire la circulation de différents objets traités.
- observer le comportement du système dans son ensemble et dans son évolution dans le temps.
- résoudre les problèmes complexes, explorer de nouvelles politiques.
- vérifier des solutions analytiques (preuves de théorème) ou formelles (preuves de programmes).
- Permet de simuler rapidement un processus évoluant sur des mois ou des années [35].

- ***Inconvénients.***

L'inconvénient majeur de la simulation est que les résultats ne sont qu'approximatifs en raison des simplifications apportées au modèle.

4.2 Outils de travail

4.2.1 L'environnement de développement (NetBeans)

NetBeans est un environnement de développement intégré (EDI) Java. Il fut développé à l'origine par une équipe d'étudiants à Prague, racheté ensuite par Sun Microsystems. En 2002, Sun a décidé de rendre NetBeans open-source.

La conception de NetBeans est complètement modulaire, ce qui fait de lui une boite à outils facilement améliorable ou modifiable. Il comprend les fonctions suivantes [8] :

- La configuration et la gestion de l'interface graphique des utilisateurs,
- Le support de différents langages de programmation,
- Le traitement du code source (édition, navigation, formatage, inspection..),
  - L'import/export des objets depuis et vers d'autres IDE, tels qu'Eclipse ou JBuilder,
  - L'accès et la gestion de bases de données, serveurs Web, ressources partagées,
  - La gestion de tâches (à faire, suivi, etc.),
  - L'aide via la documentation intégrée.

4.2.2 Le langage de programmation(Java)

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton de Sun Microsystems en 1995. Il possède de nombreuses caractéristiques parmi lesquelles on peut citer [8]:

- Il fonctionne comme une machine virtuelle (Indépendant de toute plateforme).
- Il est simple (pas de pointeur).
- Il autorise le multitâche (multithreading).
- Il peut être utilisé sur INTERNET.
- Il est gratuit.
- C'est un langage compilé : avant d'être exécuté, il doit être traduit dans le langage de la machine sur laquelle il doit fonctionner.
- Il contient une très riche bibliothèque de classes(Packages) qui permettent de :
  - Créer des interfaces graphiques.
  - Utiliser les données multimédia.
  - Communiquer à travers les réseaux…etc.

Pour modéliser les graphes en JAVA, nous avons utilisé la bibliothèque GraphStream. C'est une bibliothèque conçue pour traiter les aspects dynamiques aspects des graphes. Son principal accent est mis sur la modélisation des réseaux d'interactions dynamiques de différentes tailles. Elle propose plusieurs classes qui permettent de modéliser des graphes orientés et non orientés, de stocker tout type d'attributs de données sur les éléments graphe. Elle des structures *«stream of graph events »* (flux d'événements de graphique), qu'est à l'origine de son nom. Les composants qui génèrent un flux d'événement sont appelés *sources*, et ceux qui le reçoivent sont appelés *sinks*. Ainsi, cette bibliothèque fournit des fonctionnalités pour afficher les graphes (à l'aide de Viewer qui permet une mise en page automatique de nœuds) [8].

4.3 Notre application

L'objective principale de notre application est de stopper la diffusion d'une propagation (une épidémie, un virus informatique, une rumeur, etc.) à l'aide des nœuds critiques. Pour ce faire, elle procède en quatre étapes :

- **Etape 1 :** Création du réseau à étudier.
- **Etape 2 :** Identification de nœuds critiques.
- **Etape 3 :** Le choix du modèle de diffusion.
- **Etape 4 :** Lancement de la simulation de la diffusion.

4.3.1 Création du réseau

Le réseau à étudier est représenté par un graphe à utiliser par l'application. Il y a deux possibilités pour créer ce graphe, soit nous le créons à l'aide de l'application, ou nous le chargeons directement d'un fichier benchmark.

4.3.1.1 L'utilisation des benchmarks

Dans ce cas, le graphe existe dans un fichier écrit selon un format bien défini. Pour l'utiliser, il suffit de le charger dans l'application à l'aide du parseur. Plusieurs formats existes pour représenter différents type de réseaux (orienté, non orienté, valué, etc.), les plus répandus sont :

Pajek	Commence d'abord par le mot clé 'vertices', et une liste des sommets avec l'Id, puis le mot clés 'Edge' et les arrêtes, représenté avec matrice ou une liste. Une troisième colonne de poids arrêtes est optionnelle: *Vertices 10 1 "entity" 2 "thing" . . . *arcs 1 2 .
Edgelist	C'est une liste de deux colonnes. La premier ligne contient le nombre de nœuds, puis le nombre des arrêtes. Les autres lignes contient deux nombres correspondant aux Id des deux extrémités d'arrête. Une troisième colonne de poids arrêtes est optionnelle: 4 3 1 2 2 3 3 4
GML	GML (Graphique Modelling Language), commence par le mot clé « *graph* », Et entre deux croche « [] » une liste des nœuds définie par la structure suivante: *node* *[* *id I* *]* et une liste des arrête définie par la structure suivante: *edge* *[* *source i* *target j* *]*

Tableau 3. Exemple des formats de benchmark.

Dans notre application, nous avons adopté le format *Edgelist* pour la représentation des réseaux.

4.3.1.2 Génération du graphe:

Dans ce cas, nous générons le graphe à étudier selon un modèle. Notre application permet la création du graphe selon les trois modèles les plus répandus, à savoir :

- ***Graphe aléatoire (Random graph):***

  Un graphe aléatoire de *N* nœuds est un graphe mathématiquement généré par un processus aléatoire. Dans notre application, nous avons utilisé le graphe aléatoire binomial, souvent noté $G(N,p)$, de paramètres $\frac{N(N-1)}{2}$ et p, où chacune des $\frac{N(N-1)}{2}$ arêtes est présente avec une probabilité ≥ *p,* cela indépendamment du statut des autres arêtes. [8] Ainsi :

1) Pour chaque arrête a_i des $\frac{N(N-1)}{2}$ arrêtes
   Tire une probabilité $p\ (a_i)$
2) Si $p\ (a_i) \geq p$ alors
   Ajouter a_i au graphe G

Figure 21. Pseudos code de graphe aléatoire.

- ***Graphe Small-world (le modèle Watts – Strogatz):***

  Ce modèle, proposé par *Duncan J. Watts* et *Steven Strogatz (*1998), est un générateur de graphes avec des propriétés petit monde. [8] Le pseudo code utilisé est :

Entrée : le nombre de nœuds N, le degré moyen K (supposé être un entier pair) et un paramètre β satisfaisant : $N \gg K \gg \ln(N) \gg 1$ et $0 \leq \beta \leq 1$.

Sortie : un graphe non orienté avec *N* nœuds et $\frac{NK}{2}$ arrêts.

1) Construire un graph anneau régulier de *N* nœuds $\{x_1, x_2, x_3, \dots, x_N\}$, chaque nœud x_i est connecté à $\frac{K}{2}$ adjacentes nœuds de chaque côté comme suit :

   L'arrêt (x_i, x_j) existe si $0 < |i-j| mod\ \left(N - 1 - \frac{K}{2}\right) \leq \frac{K}{2}$.

2) Pour chaque nœud x_i, annuler l'arrête (x_i, x_j) avec $(i < j)$ avec une probabilité β, et la remplacer par l'arrête (x_i, x_k) où *k* est choisie avec une probabilité uniforme à partir de toutes les valeurs possibles qui permettent

Figure 22. Pseudos code graph small-world.

- ***Graphe Scale-free (le modèle Barabási–Albert):***

Le modèle Barabási-Albert (BA) permet de générer des graphes avec des propriétés sans échelle, largement observés dans des systèmes naturels et artificiels (Internet, World Wilde Web, réseaux sociaux, etc.). Le pseudo code utilise est [8] :

1. Ajouter un nœud initial x_1, puis ajouter un nouveau nœud à chaque fois.
2. Chaque nouveau nœud x_i est connecté à L nœuds déjà existants, avec une probabilité proportionnelle au nombre de liens que les nœuds déjà ont. Ainsi, la probabilité p_i pour que le nouveau x_i soit connecté au nœud x_i est : $p_i = \frac{k_i}{\sum k_j}$.

   Où k_i est le degré de nœud x_i, et $x_j \in$ l'ensemble de nœuds existants.

Figure 23. Pseudos code graphe scale-free.

Dans ce modèle, les nœuds très liés *hubs* ont tendance à accumuler rapidement des liens encore plus, alors que les nœuds avec seulement quelques liens ont peu de chances d'être Choisi comme destination pour un nouveau lien, car les nouveaux nœuds ont une préférence pour se fixer sur les nœuds déjà fortement liés [8].

4.3.2 Identification de nœuds critique

4.3.2.1 K-Critical Node Problem CNP

Etant donné un graphe G= (V, E), nous rappelons que *CNP* cherche à trouver un ensemble $S \subseteq V$ de k nœuds, dont la suppression minimise la connectivité par paires dans le graphe induit $G[V/S]$. Nous avons implémenté l'heuristique ci-dessous [28].

L'algorithme commence, dans une première étape, par trouver l'ensemble stable du graphe G (Maximal Indépendant Set). Pour cette étape on a utilise la procédure $MaximalIndepSet()$. On note que cette procédure ne trouve pas nécessairement l'ensemble indépendant maximal, mais il trouve un ensemble indépendant maximum.

Procédure $MaximalIndepSet$ (G: graphe)
$S \leftarrow \emptyset$
While $V \neq \emptyset$ **do**
Chose v minimum degree in G.
$S \leftarrow S \cup \{v\}$.
remove v and its neighbors from G.
End while
Return S /∗ *Maximal Independent Set* ∗/.

End procédure $MaximalIndepSet$.

Procédure CriticalNode (*G:graphe, k:nombre de noeuds critiques*)

Étape 1 : $MIS \leftarrow MaximalIndepSet(G)$

Étape 2: **While** $(MIS \neq |V| - K)$ **do**

$$j = \text{argmin} \sum_{i \in c} \frac{\delta i(\delta i - 1)}{2} : i \in V/MIS$$

$$MIS \leftarrow MIS \cup \{j\}$$

end while

Return V/MIS /∗ set of *k* nodes to delete ∗/

End procédure CriticalNode

Figure 24. Pseudos code critical node problem.

Après l'obtention du MIS, la boucle de la procédure *CriticalNode ()* choisit (dans la boucle While) le nœud $j \in V/MIS$, qui si on l'ajoute au graphe *G*(MIS) minimise la fonction objectif. Ensuite, le MIS est augmenté en ajoutant le nœud j, et le processus se répète jusqu'à ce que $MIS = |V| - K$. A cette étape, la procédure se termine et l'ensemble de nœuds critiques est retourné. Cet algorithme à une complexité $O(|V|^2|E|)$ [28].

4.3.2.2 MaxNum - Maximize the Number of connected components

Dans *MaxNum,* on cherche seulement à maximiser le nombre de composantes connexes par la suppression d'un ensemble de *k* nœuds, contrairement au CNP qui vise à la fois, à maximiser le nombre de composants connexes tout en minimisant la variance dans les cardinalités des composantes générées. Pour *MaxNum,* nous avons proposer le pseudo-code suivant :

Procedure MaxNum (*G*,*k*)

Étape 1: $MIS \leftarrow MaximalIndepSet(G)$

Étape 2: **while**$(MIS \neq |V| - K)$ **do**

$j = \arg\max \rho$ (ρ *est le nombre de c*omposantes connexe).

$MIS \leftarrow MIS \cup \{j\}$.

end while

Return V / MIS /∗ set of *k* nodes to delete ∗/.

End procedure MaxNum.

Figure 25. Pseudos code MaxNum.

4.3.2.3 CC-CNP: Cardinality Constrained Critical Node Problem

CC-CNP cherche un ensemble de nœuds S ⊆ V avec une cardinalité minimum, de telle sorte que les indices de connectivité des nœuds dans le graphe induit G[V\S] sont inférieurs à un seuil donné β. Pour cette variante, nous avons proposer le pseudo code suivant :

Procédure CC-CNP (*G*, β)

Étape 1: $MIS \leftarrow MaximalIndepSet(G)$

Étape 2: Remain_Nodes ← V/MIS

Critical ← ∅

While (Remain_Nodes ≠ Ø) **do**

For ($i = 1$ *to* Remain_Nodes) **do**

Let *i* be the node of minimal degree.

If $(\frac{|s|(|s|-1)}{2} \leq \beta \; \forall \; s \; is \; a \; component \subseteq G(MIS \cup \{i\}: i \in V/MIS)$

then $MIS \leftarrow MIS \cup \{i\}$

Romove {i} from Remain_Nodes

Else

If (∃ a component $s \subseteq G(MIS \cup \{i\}: i \in V/MIS) : \frac{|s||s|-1)}{2} \geq \beta$)

Critical ← $\{i\}$

Romove {i} from Remain_Nodes

end if

end if

end for

end while

return Critical /* set of nodes to delete */

end procedure CC-CNP

Figure 26. Pseudos code CC-CNP.

4.3.3 Choix du modèle de diffusion

Pour la simulation du processus de propagation de la diffusion, nous avons implémenté les trois modèles les plus répandus, à savoir : le *Modèle SI (Sain/Infected), le Modèle SIS (Sain/Infected/Sain) et le Modèle SIR (Sain/Infected/ Recovered*). (voir *section 2.2.2*).

4.3.4 Lancement de processus de diffusion

Dans cette section, nous allons présenter un scénario général de l'utilisation de notre application de la création du réseau jusqu'au blocage de la diffusion.

- ***Création d'un réseau :***

Le choix d'un modèle de réseau à créer parmi les quatre modèles possibles, comme il est illustré dans la figure sous-dessous.

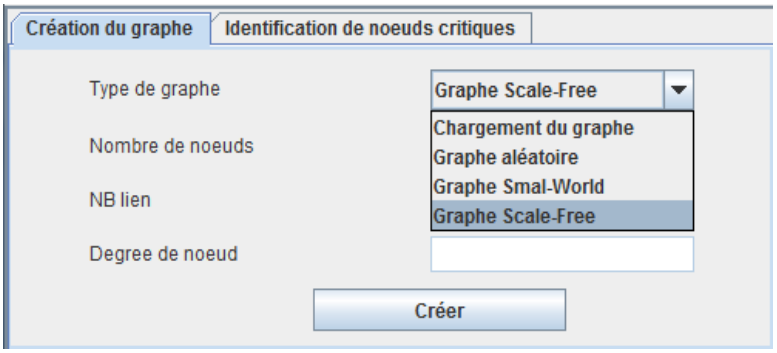


Figure 27. L'angle de chois de modèle de graph.

Chaque modèle a ces propres paramètres selon les codes illustrés précédemment dans la section 4.3.1.2. La figure 28 illustre la création d'un réseau Scale-free de 150 nœuds avec un nombre de lien initial égal à 1:

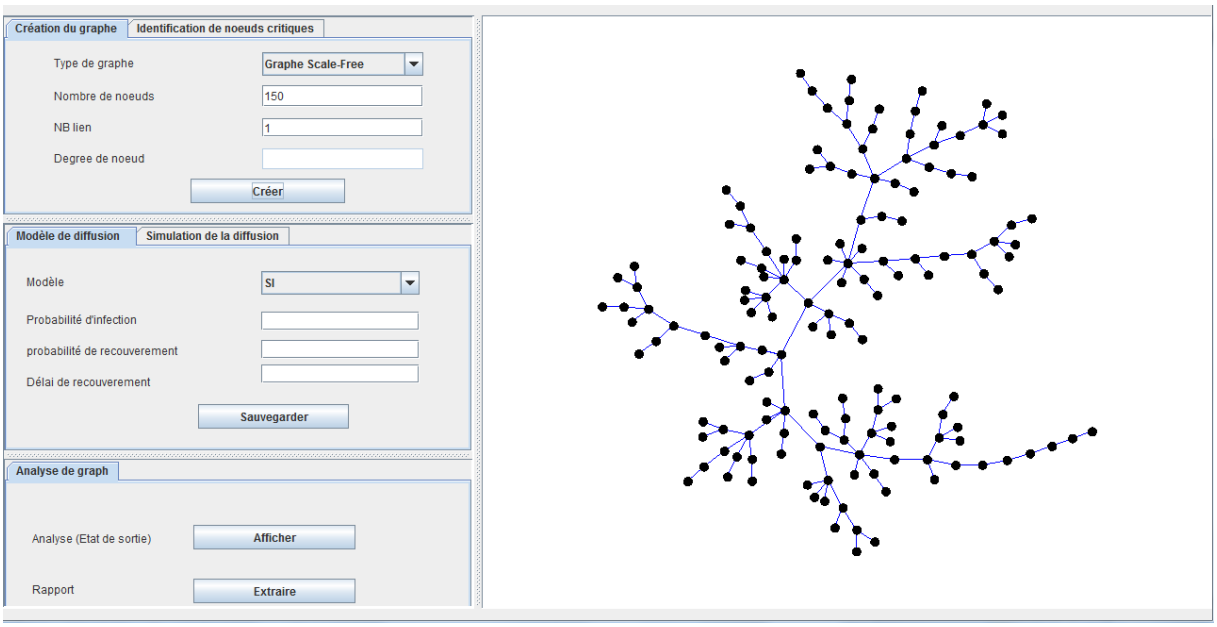


Figure 28. Générer un graphe Scale -free avec 150 nœuds et NB lien initial 1.

- ***Identification de nœuds critiques :***

Le choix de la variante CNDP pour la détection de nœuds critiques.

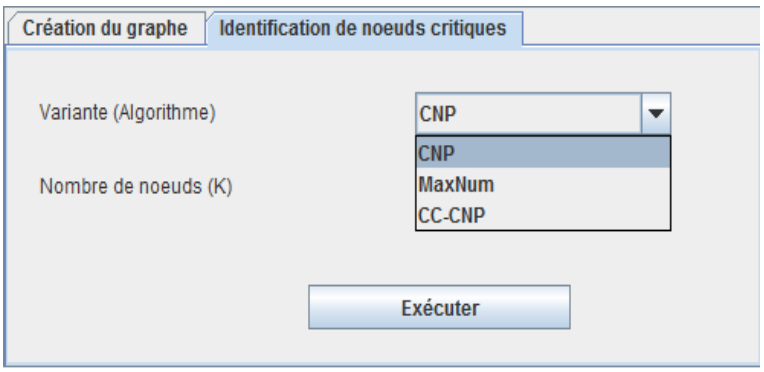


Figure 29. L'angèle de chois de variante CNDP.

Chaque modèle à ces propres paramètres selon les codes illustré précédemment dans la section 4.3.1.3. La figure 30 illustre l'identification de k=10 nœuds critiques dans le réseau déjà généré en utilisant la variante CNP :

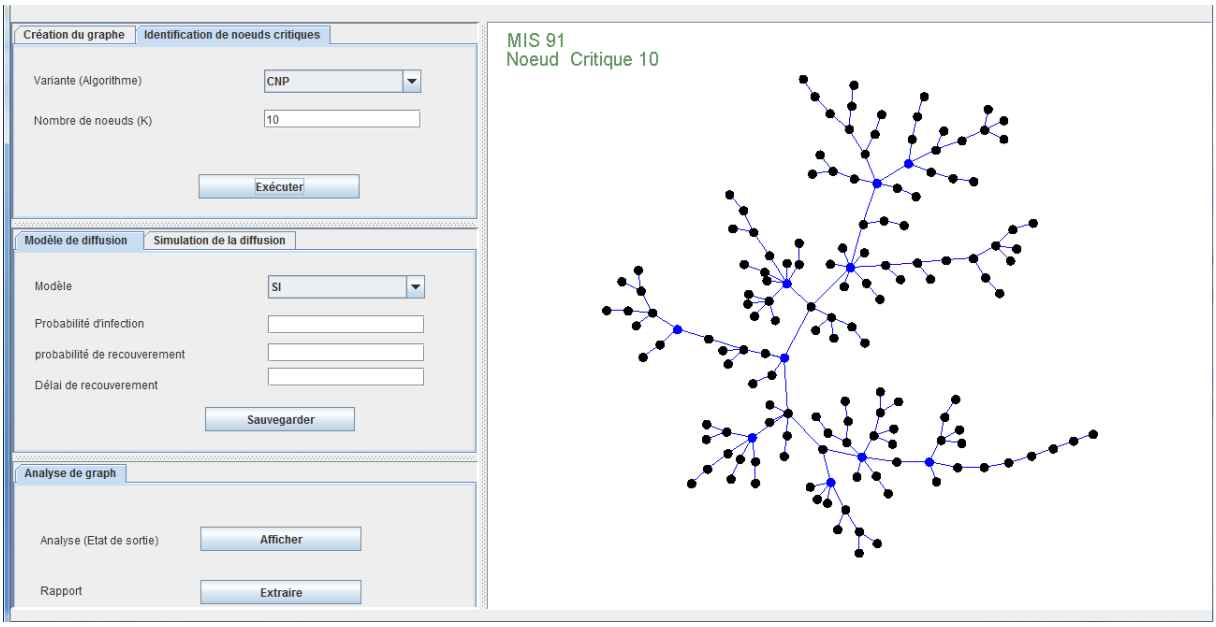


Figure 30.La solution optimale selon la variante CNP avec K= 10.

La figure 31 illustre l'identification d'un ensemble de nœuds critiques dans le réseau généré en utilisant la variante CC-CNP avec L égal à 15 :

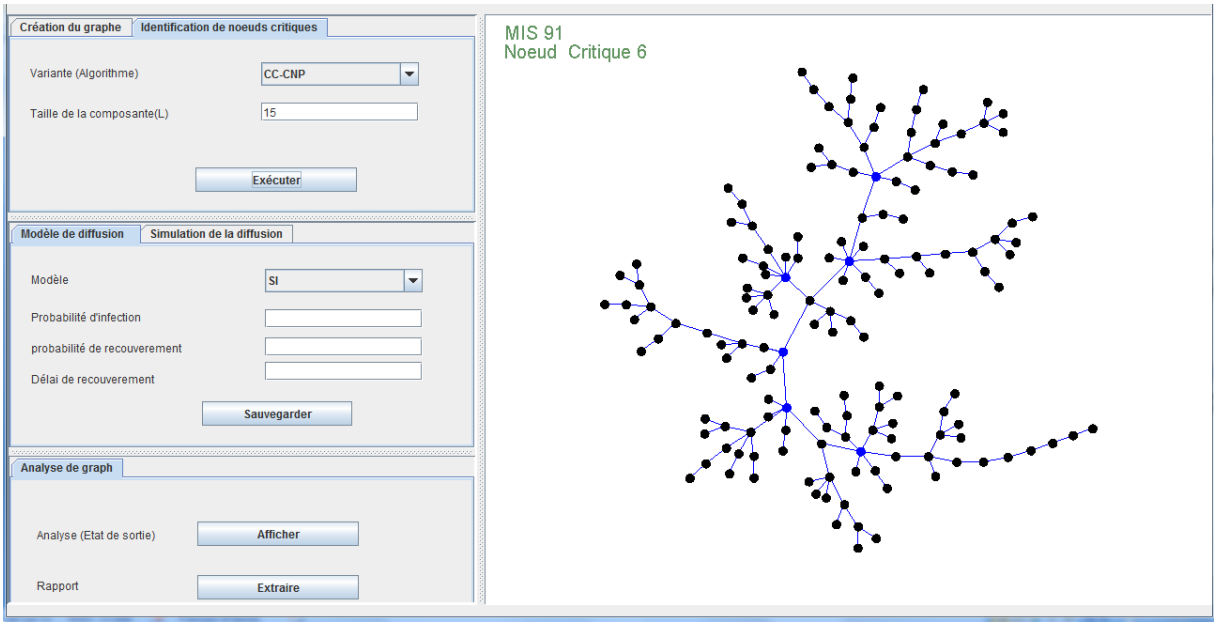


Figure 31.La solution optimale selon la variante CC-CNP avec L= 15.

- ***Choix du modèle de diffusion***

Nous choisissions un modèle de diffusion parmi les trois modèles existants *SI*, *SIR*, *SIRS*. Nous sauvegardons les paramètres de modèle choisi selon les modèles illustré dans la section 2.2.2 :

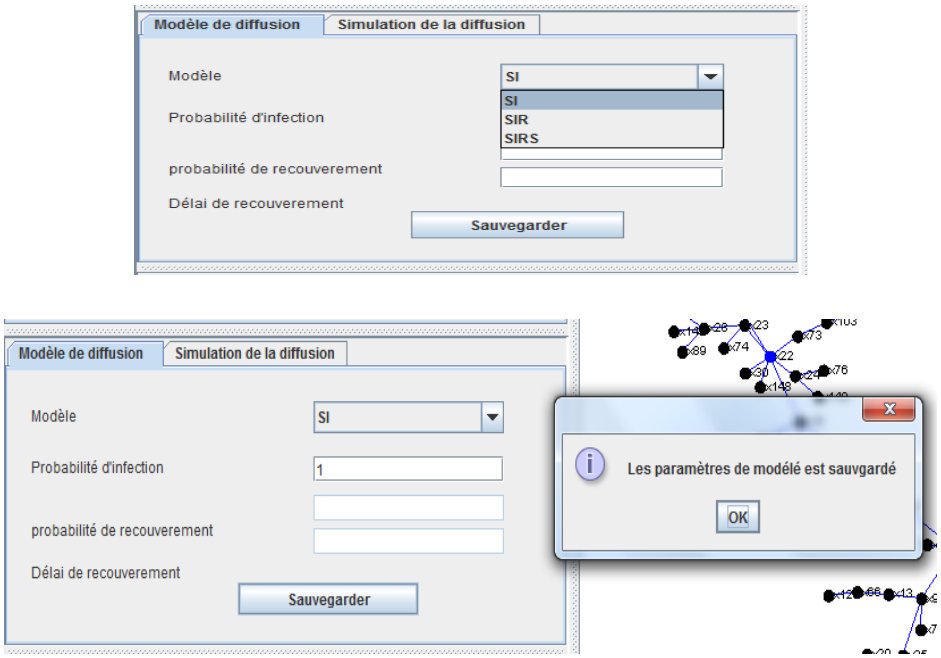


Figure 32. Le chois de modèle de diffusion, et la sauvegarde des paramètres.

- ***Lancement de la simulation***

Tout d'abord, nous choisissions la source de diffusion selon un des deux cas possibles : via le champ « Source de diffusion » dans l'onglet « simulation de la diffusion », ou en utilisant la souris à l'aide d'un clic sur le nœud choisi comme source. Ensuite, on détermine la vitesse de propagation.

La figure 33 illustre le choix de modèle SI avec une probabilité d'infection égale à 1. Les nœuds encadré appartienne a la même composante, Un nœud rouge représente un nœud infecté (I), un nœud noir représente un nœud sain (S). Les nœuds entoure sont de la même communauté.

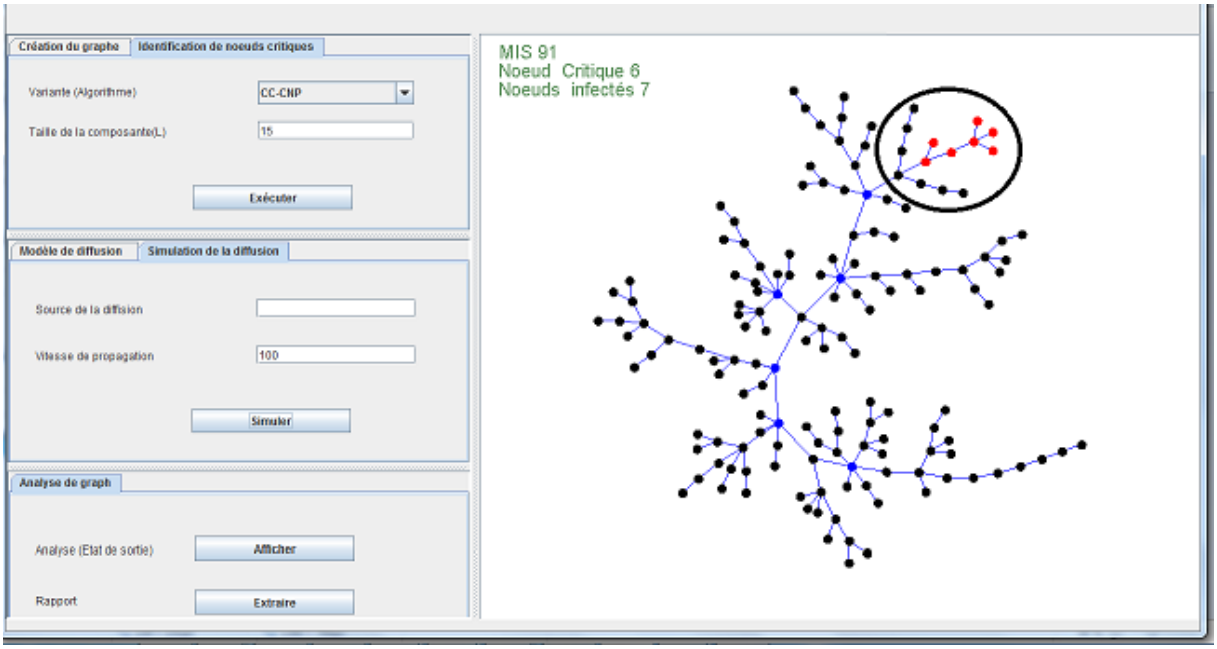


Figure 33. Scenario d'infection selon le modèle SI avec B=1 dans le début de diffusion.

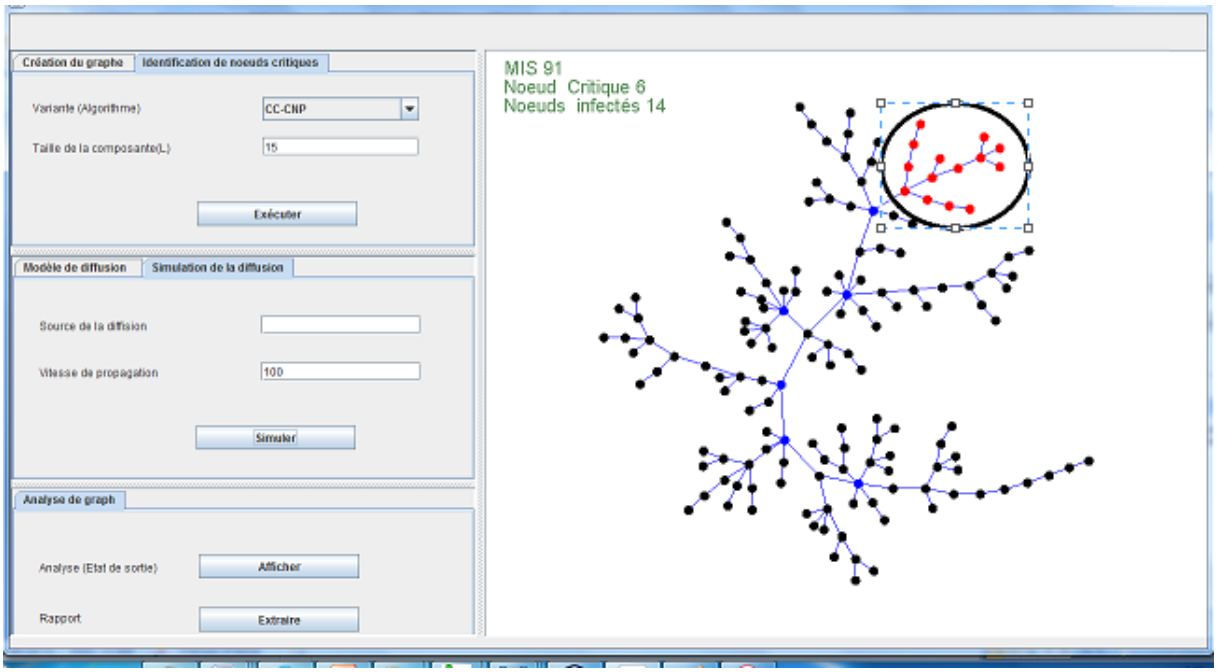


Figure 34. Scenario d'infection selon le modèle SI avec B=1 à la fin de diffusion.

La figure 35 illustre le choix de modèle SIR avec une probabilité d'infection égale à 1, une probabilité de recouvrement égale à 1, et une vitesse de recouvrement égale à 800. Un nœud bleu représente un nœud guéri (R). Les nœuds entoure sont de la même communauté.

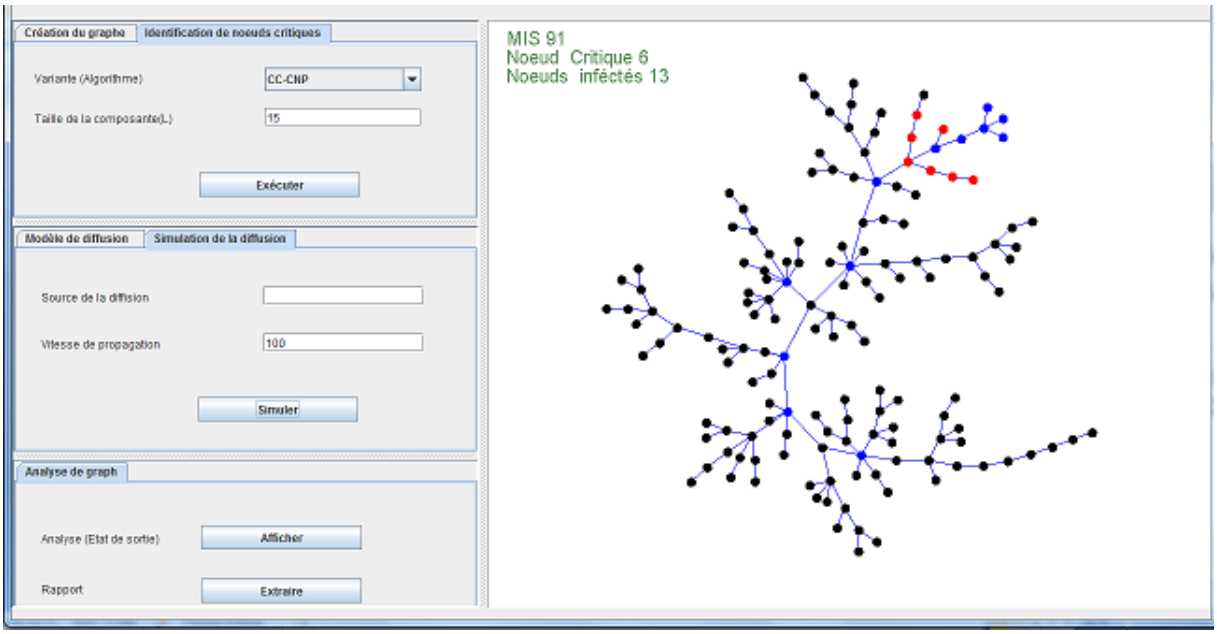


Figure 35. Scenario d'infection selon le SIR avec B=1, B'=1 au début de diffusion.

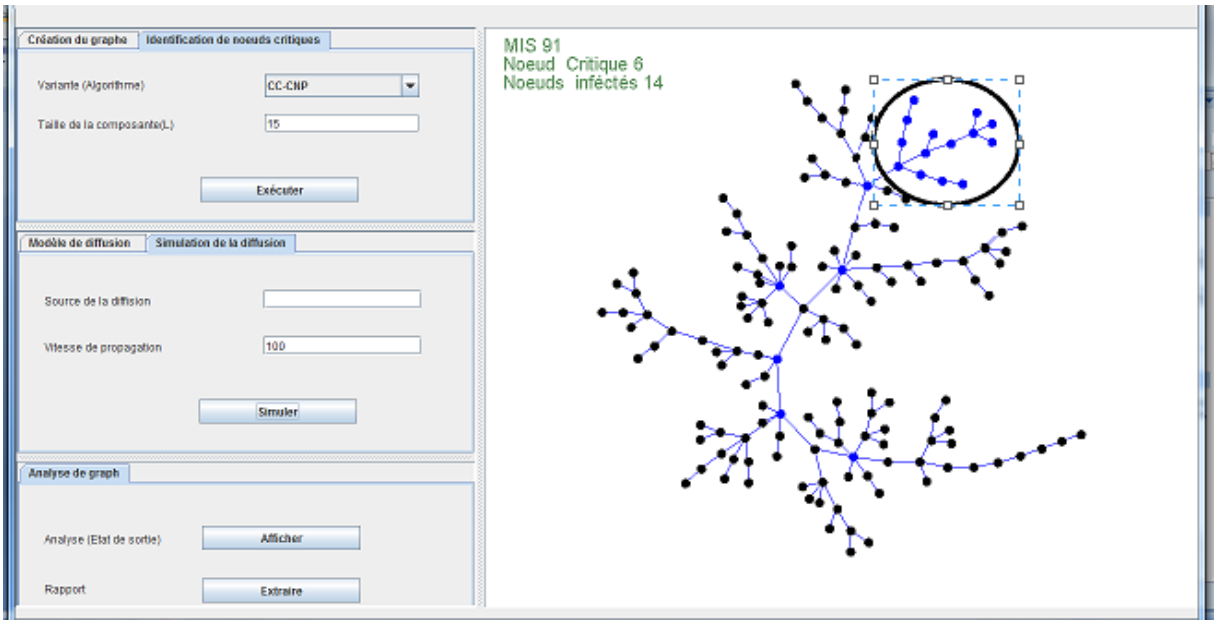


Figure 36. Scenario d'infection selon le modèle SI avec B=1 à la fin de diffusion.

4.3.5 Analyse de réseau

Notre application permet Ainsi d'effectuer une analyse sur le réseau à étudier. Les différents paramètres considérés par cette analyse sont :

a. *L'ordre de réseau :* représente le nombre de nœuds dans le réseau.
b. *La taille de réseau :* représente le nombre d'arrêtes dans le réseau.
c. *Le degré minimal (maximal) dans le réseau :* représente le nombre de liens minimal (maximal) que peut avoir un nœud avec les autres nœuds du réseau.
d. *La distribution de degrés du réseau* : représente la distribution des degrés de tous les nœuds du réseau.
e. *La densité du réseau*: elle mesure si le réseau a beaucoup ou peu d'arrêtes, calculé en utilisant la formule suivant : $\frac{|E|*2}{|V|(|V|-1)}$.
f. *Nombre de communautés :* Nous allons essayer de proposer une approche qui permet de détecter les différentes communautés dans un réseau. Nous allons utiliser la notion d'*edge Betweeness. Edge Betweeness* est un indice de centralité d une arête dans un réseau . Il est égal au nombre de chemins les plus courts de tous les sommets de tous les autres qui passent à travers cette arête. Une arête avec une grande betweenness centrale a une grande influence sur le transfert des objets à travers le réseau. La centralité de betweenness d'une arête *a* est donnée par l'expression:

$$edge\ Betweeness=\sum_{s\neq t}\frac{\sigma_{st}(a)}{\sigma_{s,t}}\ .$$

Où σ_{st} est le nombre total de plus courts chemins plus courts dans le réseau, et $\sigma_{st}(a)$ est le nombre de ces chemins qui traversent *a*.

g. *Taille maximale de communauté* : représente la taille de la plus grande composante.

h. *Le diamètre de réseau* : représente le plus long chemin parmi les plus courts chemins existe dans le réseau.

i. *Le plus court chemin :* le plus court chemin entre deux nœuds du réseau est la distance minimale (en nombre de nœuds à parcourir) entre ces deux nœuds. Pour le calculer, nous avons implémenté l'algorithme de Dijkstra [8] avec une petite modification :

```
Procédure Shortest_path (G : graphe, source, destination : node )
    visite ← adj [source].  // vector of neighbors of source node
    Distance [i] ← 1, for all node i ϵ visite  // vector of distance from source node
    source← marked.
    While (visite ≠Ø) do
        If (destination ∈ visite) then
            break .
        Else
            Let x be the node of minimal vlaue in Distance[].
            visite ← visite/x
            x ← marked.
            For (all node y ϵ adj[x])
                if (y ∉visite || y != marked)
                    visite ← visite ∪ y
                    Distance[y] ← Distance[y] + 1;
                end if
            end for
        end if
    end While
    return distance destination
End procedure Shortest_path
```

Figure 37. Pseudos code Shortest_path.

Pour afficher les différents paramètres et les extraire dans un fichier texte nous utilisions les boutons *afficher* et *extraire* dans le troisième panneau (a gauche aux bas).

Conclusion

Dans ce chapitre, nous avons présenté une vue sur la simulation. Nous avons aussi présenté les différents outils et langages de programmation utilisés pour le développement de notre application. Enfin, nous avons a ajouté quelques interfaces pour illustrer le fonctionnement de notre application Tout en expliquant le déroulement des scénarios de simulation.

Conclusion générale

Conclusion générale

L'objective principale de notre projet est l'étude d'une fonctionnalité réseau en utilisant les nœuds critiques. Pour cela, nous nous sommes intéressés au blocage de la diffusion dans les réseaux complexes en utilisant les nœuds critiques comme pare-feu pour stopper la propagation.

Nous avons présenté les différents concepts de la théorie de graphes nécessaires pour l'analyse des réseaux complexes, ainsi nous avons donné un aperçu sur les différentes caractéristiques des problèmes d'optimisation et leurs méthodes de résolution. Ensuite, nous avons introduit les réseaux complexes et les différents modèles de diffusion dans ce genre de réseaux.

Notre projet s'est réalisé en deux phases : la détection des nœuds critiques dans le réseau, ensuite la simulation et le blocage de la diffusion. Pour la première, nous avons présenté les différentes variantes de détection de nœuds critiques que nous avons implémentées. Pour la deuxième, nous avons implémenté les différents modèles de diffusions, et avons montré par la suite le blocage d'un propagation a l'aide des nouds critiques. Pour ce faire, nous avons utilisé le langage Java. Ce langage contient des packages qui facilite la mise en œuvre de grandes applications.

En utilisant notre application, on peut simulation toute diffusion possible dans un réseau complexe, et trouver une solution optimale tout en exploitant le minimum de ressources pour stopper la propagation d'une entité nocive. Le choix des nœuds à protéger pour bloquer la diffusion dépond de la nature de la diffusion et de la meilleur façon pour la contrôler, ceci est assurer par la variante CNDP appliquée. Malgré tout ce qui est assuré par cette application, il reste des lacunes que nous envisageons comme perspectives.

Notre application a besoin d'être enrichie par d'autres variantes CNDP, car nous avons implémenté seulement les trois principales. Aussi, les algorithmes implémentés peuvent être optimisés de plus. Des paramètres nécessaires pour l'analyse des réseaux, à titre d'exemple nous citons les différentes mesures de centralité, doivent être ajoutées à l'application pour plus de performance, ainsi qu'un outil de comparaison de différentes approches est à intégrer pour une meilleure décision de la solution à prendre.

Bibliographie

1 Saglietto, Laurence. *Quelques points de repères dans l'étude des réseaux par la théorie des graphes*. 2006.

2 Muller, Didier. *Introduction a la théorie des graphes.*

3 Solnon, Christine. *Théorie des graphes et optimisation dans les graphes*.

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

5 Topart, Hélène. *Etude d'une nouvelle classe de graphes: les graphes hypotriangulés*. 2011.

6 Blanchard, Nicolas K. *Le théorème des graphes parfaits*. 2015.

7 Brax, Nicolas. *Modélisation de réseaux, de l'auto-réparation à l'auto-organisation*. Université Paul Sabatier. 2008.

8 *https://fr.wikipedia*.

9 Levorato, Vincent. *Contributions à la Modélisation des Réseaux Complexes: Prétopologie et Applications*. Université paris , 2008.

10 Trondi, Fatiha. *Résolution du problème de l'emploi de temps. Proposition d'un algorithme évolutionnaire multi objectif*. Université Mentouri – Constantine, 2005-2006.

11 Mancel, Catherine. *Modelisation et resolution de problemes d'optimisation combinatoire issus d'applications spatiales*. l'Institut National des Sciences Appliquées de Toulouse. 2005.

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

13 Barrat and Autre. *Dynamical Processes on Complex Networks*. 2008.

14 Orman, Gunce Keziban and autre, et. *Relation entre transitivité et structure de communauté dans les réseaux complexes*.

15 Allard, Antoine. *Modélisation mathématique en épidémiologie par réseaux de contacts*. Université Laval. 2008.

16 Noel, Pirre-Andre. *Dynamoque stochastique sur réseaux complexe*. 2012.

17 Labatut, Vincent. *Étude de l'omniprésence des propriétés petit-monde et sans-échelle*. 2014.

18 Dubé, C and Autre. *L'utilisation de modéles épidémiologique pour la gestion des maladies animales*. 2007.

19 Noel, Andre Pirre. *Dynamique de modéles épidémiologiques Applications au cas du virus du Nil occidental*. Universite Laval Quebec. 2007.

20 Albano, Alice. *Dynamié des graphes de terrain analyse en temps intrinseé*. 2014.

21 Lewis, J. M and Autre. *The node-deletion problem for hereditary properties is np-complete*.

22 Lalou, M and Autre. *The Critical Nodes Detection Problem in Networks*.

23 Borgatti, S P. *Identifying sets of key players in a network, Computational, Mathematical and Organizational Theory*. 2006.

24 Shen, S and Autre. *Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs.* 2012.

25 Shen, S and Autre. *Exact interdiction models and algorithms for disconnecting networks via node deletions. Discrete Optimization*. 2012.

26 Arulselvan, Ashwin and Autre. *Cardinality- constrained critical node detection problem. In Performance models and risk management in communications systems*. 2010.

27 Veremyev, Alexander and Autre. *An integer programming framework for critical elements detection in graphs*. 2014.

28 Aruleslvan, Ashwin and autre. *Detecting critical nodes in sprse graphs*.

29 Boginski, Vladimir and autre. *Identifying critical nodes in protein-protein interaction networks*.

30 Dinh, T N and Autre. *Precise structural vulnerability assessment via mathematical programming*. 2011.

31 Ovelgonne, M and autre. *Covertness centrality in networks. In Advances in Social Networks Analysis and Mining(ASONAM)*. 2012.

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

33 Belattar, Brahim. *Modélisation & Simulation sur Ordinateur*. 2003/2004.

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

35 aissani, Amar. *Modélisation et simulation*. 2007.

36 BOURAS, Mohamed Boudaroua. *Etude des Propriétés Thermodynamiques, Structurales et de Transport du NaCl fondu à haute température par Dynamique Moléculaire.*. Université d'ORAN. 2007.