Stratégie et optimisation de mise à jour de firmwares de LiveBox

Encadrant : **M. Stanislas Francfort**

Etudiant : **M. Enzo Escobar**

Remerciements

J'exprime ma plus grande gratitude à Orange Labs qui m'a accueilli chaleureusement au sein de ses locaux.

Je souhaite remercier aussi mon maître de stage, M. Stanislas Francfort, qui m'a accompagné tout au long de mon stage et a su me guider dans la bonne direction pour mener à bien mon projet.

Je souhaite enfin remercier toutes les personnes que j'ai pu rencontrer durant mon stage pour leur accueil et l'ambiance au sein de l'unité de recherche et développement.

Table des matières

1 Introduction

La recherche opérationnelle représente un enjeu important dans beaucoup de grandes entreprises actuelles. Les enjeux économiques sont tels et la taille des problèmes si importante qu'une approche mathématique de la décision est souvent nécessaire.

France Télécom n'échappe pas à la règle. Pour des activités au cœur de l'entreprise telles que le dimensionnement de réseau ou la gestion des coûts, la recherche opérationnelle a un rôle économique majeur.

1.1 Présentation de l'entreprise

France Télécom, née officiellement le 1er janvier 1991, offre des services de communication mobile, internet et fixe à 183,3 millions de clients dans 32 pays. En Europe, il est le troisième opérateur mobile et le deuxième fournisseur d'accès internet haut débit ADSL.

Le Groupe est également l'un des leaders mondiaux des services de communication aux entreprises et multinationales grâce à Orange Business Services. Sa marque Orange fédère près de 132 millions de clients en proposant des services intégrés innovants, simples d'utilisation et éco-responsables. Ceux-ci couvrent un large éventail de besoins qui va de l'accès au monde numérique jusqu'aux loisirs et divertissements, en passant par les services à la personne ou les réseaux sociaux et communautaires. Les chiffres de cette société sont le reflet de sa réussite :

- 123,7 millions de clients mobile
- 13,5 millions de clients haut débit ADSL en Europe
- 3e opérateur mobile en Europe
- 2e fournisseur d'accès internet ADSL en Europe
- 45,9 milliards d'euros de chiffre d'affaires réalisés en 2009

Pour rester toujours à la pointe l'innovation, le département de Recherche et Développement de France Télécom a un rôle majeur à jouer. Orange Labs est le nom donné à ce département qui est sous la direction de la division Innovation Marketing Group. Orange Labs est composé de 6 centres de Recherche et Développement (CRD) :

- BIZZ (Business Services)
- MAPS (MiddlewAre et PlateS-formes avancées)
- RESA (RESaux d'Accès)
- SIRP (Service Intégrés Résidentiels et Personnels)

- TECH (TECHnologies)
- CORE (COeur de REseau)

Ce dernier CRD nous intéresse plus particulièrement puisque c'est au sein de celui-ci que le stage s'est déroulé. Le but de CORE est de s'attaquer à toutes les problématiques "cœur de réseau" du groupe, que ce soit le réseau de téléphonie (fixe et mobile) ou internet. Le CRD CORE est divisé en 5 laboratoires :

- M2I (Multimedia networks for non-conversational fixed/mobile services : Image Internet)
- NAS (Network Architecture center for Services deployment)
- NIS (Network Integration center for Services deployment)
- TPN (Transport and Packet Network)
- M2V (Multimedia networks for conversational fixed/mobile services : Voice Visio)

Le stage a été réalisé au sein du laboratoire M2V, dont les problématiques de recherche tournent autour du réseau de communication directe, que ce soit par la voix ou par la vision. Cela inclut non seulement le réseau de téléphonie, mais aussi le réseau internet qui permet ce genre de services. Pour finir la descente de l'organigramme, M2V est composé de 6 équipes, appelées Unités de Recherche et Développement (URD) et affectées à des problématiques bien précises :

- CIMI (CORE network IP convergence for Mobile and IMS)
- ISN (Integration of conversational Services in Networks)
- IRI (Interconnexion des Réseaux pour les services conversationnels sur IP)
- DPC (Sécurité Ingénierie Développement de tests pour équipements et réseaux des services conversationnels)
- TSC (Traitement d'appel et de Session à l'accès pour les services Conversationnels)
- IODA (IMS and mobile Optimisation and Database Architecture)

C'est au sein de l'URD IODA qu'a été mis en place le projet d'optimisation de la mise à jour de LiveBox sur lequel j'ai effectué mon stage. IODA est une équipe d'une vingtaine de personnes aux profils variés. L'objet de mon stage est sous la responsabilité de M. Stanislas Francfort, maître de stage.

1.2 Présentation de l'équipe

IODA est une unité de Recherche et Développement rattachée au laboratoire M2V dont les activités portent sur les réseaux pour les services conversationnels. Cette URD est dirigée par M. Bertrand Decocq. IODA est spécialisée dans l'IP Multimedia Subsystem (IMS). C'est une architecture standardisée Next Generation Network (NGN) pour les opérateurs de téléphonie, qui permet de fournir des services multimédias fixes et mobiles. Ce système utilise la technologie VoIP basée sur une implémentation 3GPP standardisée de SIP fonctionnant sur un protocole standard IP. Ses activités autour de l'IMS sont concentrées majoritairement sur l'optimisation mobile et sur l'architecture de base de données.

Cette URD est chargée de la validation de base de données pour l'IMS et effectue les études technico-économiques environnantes. Une partie plus orientée vers la recherche de l'équipe se concentre sur les problèmes d'optimisation de la supervision du réseau et sur le traitement des données. En terme de recherche opérationnelle, IODA développe des axes de recherche dans les thèmes métiers suivant :

- l'optimisation des processus d'exploitation du Groupe France Télécom pour une diminution des coûts opérationnels
- l'optimisation du dimensionnement des réseaux avec un focus particulier sur les réseaux d'accès optique.

2 Présentation du stage

2.1 Présentation du sujet

De nos jours, la sécurité de l'information est essentielle. Beaucoup de pirates sévissent sur l'Internet afin par exemple de récupérer des données confidentielles ou compromettre des services. Les systèmes autonomes tels que les LiveBox n'échappent pas à la règle puisqu'ils comportent des failles susceptibles d'être exploitées. Ces pirates des temps modernes ont mis au point de nombreux virus et vers à cet effet. Les vers notamment se propagent et infectent les machines les unes après les autres tant qu'elles ne sont pas protégées.

Le combat contre les pirates touche en grande partie les entreprises telles qu'Orange car l'enjeu économique est de taille. C'est pour cette raison qu'une équipe de sécurité a été créée au sein d'Orange. Cette équipe élabore des défenses contre tout type d'attaque dès qu'ils en ont connaissance. Ces attaques touchent potentiellement tout type de matériel, y compris les LiveBox.

C'est dans ce contexte qu'intervient mon stage qui a pour sujet l'optimisation de la mise à jour du parc des Livebox pour lutter contre la propagation des vers. Cette étude est pertinente car même si certains vers se propagent très rapidement nous obligeant à réagir à posteriori, d'autres vers se propagent lentement pour ne pas être repérés. Ces derniers

se propagent suffisamment lentement pour donner réellement le temps à l'administrateur réseau de réagir en conséquence.

L'ordre de mise à jour des machines a alors son rôle à jouer dans la propagation d'un vers. En mettant à jour les machines de manière optimale, il est possible de ralentir la propagation du malware voir de la stopper. Cette volonté de réagir face à ce type d'attaque constitue le cœur de mon stage. Ce dernier s'articule autour de trois parties :

- Une étude de l'état de l'art du traitement d'une infection dans un réseau
- Une partie programmation et simulation pour obtenir des résultats dans le cas particulier du réseau de LiveBox
- Une étude mathématiques pour appuyer les nouveaux modèles et pour conclure d'après les résultats obtenus

Ce stage a une grande composante exploratoire car actuellement il n'existe pas de méthode particulière adressant les enjeux de sécurité pour mettre à jour les LiveBox. C'est un travail non seulement d'optimisation mais aussi de recherche. Il allie les mathématiques et la recherche opérationnelle à la programmation. Ce stage aboutira sur deux présentations au sein de l'entreprise. Une première présentation à destination de l'équipe d'optimisation afin de présenter mes trois mois de travail au sein de l'équipe. Une seconde présentation à destination de l'équipe impliquée dans la recherche de solution de sécurité qui représente l'utilisateur final de mes travaux. Un exemplaire de ce rapport sera donc fourni à un département opérationnel.

2.2 Ressources à disposition (humaines et matérielles)

Pour réaliser mon étude, Orange Labs a mis à ma disposition un ordinateur et un accès à un serveur de calcul. Cette procédure, pour des questions de sécurité, permet de conserver tout ce qui est fait au sein de l'entreprise.

Plus important et plus enrichissant, il m'a été permis de rencontrer des experts dans le domaine de l'optimisation et dans le domaine de la sécurité afin d'orienter mes recherches. Ainsi j'ai pu échanger avec des experts en optimisation qui m'ont permis de démarrer sur la bonne voie mon projet, et des experts en sécurité afin d'éprouver mon modèle dans les conditions les plus réelles possibles.

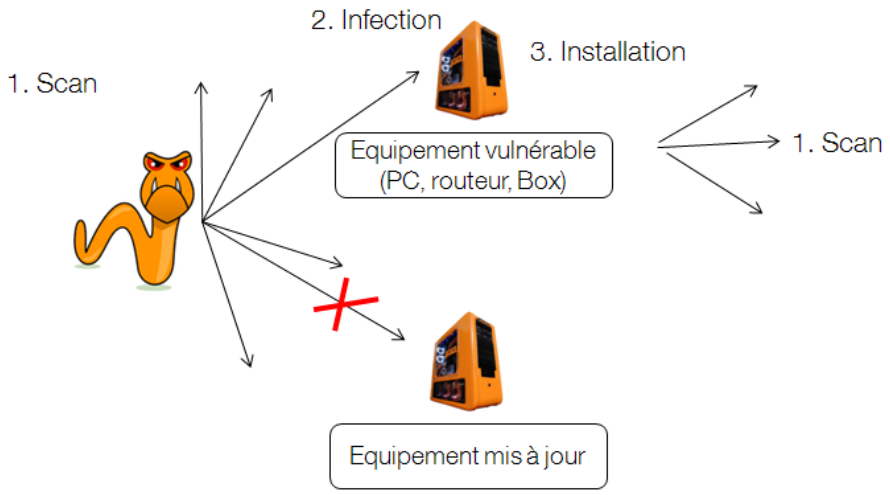
3 Etat de l'art

3.1 Etude des vers/virus

Comme pour tout nouveau projet, afin de travailler le plus efficacement possible, il est nécessaire de faire un état des lieux sur ce qui existe déjà. Les premiers jours de stage ont

donc été consacrés à l'étude des différents virus et vers existants. Cette étude m'a permis de prendre conscience des enjeux de la sécurité et d'avoir un aperçu de la diversité qui existe en terme d'attaque sur un réseau.

Il a tout d'abord été important de comprendre la différence entre un ver et un virus afin de bien délimiter le périmètre de l'étude. Ce que nous considérerons comme un ver est un logiciel malveillant se transmettant de machine en machine à travers le réseau. Contrairement à un virus, il n'a pas besoin d'un programme hôte pour se reproduire. Le but des vers est en général d'installer d'autres programmes nocifs tels que des spywares, keyloggers, backdoors ou trojan. Grâce à ces programmes, pour celui qui les contrôle d'espionner la machine infectée [7]. Une autre utilisation fréquente est le contrôle à distance de l'hôte pour des attaques de type déni de service.



Mon travail trouve naturellement une première limite en cette définition. Par la suite je n'étudierai que les logiciels malveillants se dupliquant de manière autonome, c'est-à-dire suivant une stratégie de propagation définie à l'avance. D'autre part nous considèrerons que la mise à jour nettoie et protège définitivement l'hôte du ver étudié.

Mon étude m'a ensuite conduit vers les infections les plus connues [5]. C'est par ce biais là que j'en ai appris plus sur les différentes stratégies de propagation des vers. Les rois de la contamination sont donc les vers suivants :

- Morris, le premier ver de l'histoire. Créé par un étudiant américain, il se propageait de machine en machine en cherchant sur chaque nouvel hôte ses futures victimes dans le fichier .rhosts. Une fois la liste de victime établie, il passait à l'attaque.
- Code Red II [1], un ver qui a paralysé tout le réseau internet pendant plusieurs heures. Il se propageait en attaquant majoritairement des ordinateurs qui étaient dans la même classe d'adresse IP que l'hôte. Il s'est propagé tellement vite et a

infecté tellement de machine que le seul fait de chercher des victimes potentielles a bloqué Internet.

- ADM Worm, il tire une adresse IP aléatoire, puis effectue un scan de manière incrémentale pour trouver ses victimes. Une fois sa victime trouvée, il utilise des failles IQUERY pour s'installer et augmenter ses privilèges afin de relancer l'infection.

- Nimda, le premier ver à utiliser plusieurs mécanismes de propagation à la fois. Il utilisait la propagation par IP (scan aléatoire d'adresses), par e-mail, par les navigateurs web et par création de fichier.

Un premier bilan sur les stratégies de propagation peut être tiré. En effet, la plupart du temps, la propagation de vers se fait de manière aléatoire. Il est tout de même intéressant de signaler que la stratégie de propagation de Code Red II est un petit plus évolué que les autres et que cela a sûrement joué dans la vitesse d'infection impressionnante qu'il a eu. Beaucoup d'analyses, d'études et de simulations se sont basées par la suite sur ce vers.

En complétant cette première liste à l'aide de stratégie de propagation moins connues mais non moins efficaces, il est possible de dresser un bilan sur le fonctionnement général des vers. Il est notamment possible de leur stratégie de propagation, ce qui nous intéresse au plus au point pour l'étude que nous allons mener. Voici donc les stratégies de propagation les plus répandues à ce jour [3] :

- Scan des adresses IP sur le réseau : ce type de ver scan un réseau pour trouver de nouveaux hôtes. Les scans sont souvent aléatoires car c'est une méthode très simple à mettre en œuvre. Cependant, il existe aussi des vers qui scannent de manière incrémentale le réseau. Du fait de sa simplicité, c'est la seconde méthode la plus utilisée par les pirates à l'heure actuelle.

- Liste interne d'adresses IP : au moment de sa création, le ver contient une liste d'hôtes potentiels. Il infectera uniquement les machines sur la liste.

- Liste extérieure d'adresses IP : le ver communique avec son créateur pour savoir dans quelle direction se tourner. Ainsi, le pirate coordonne l'attaque de tous ses vers. Cette technique est difficile à mettre en œuvre car le ver doit communiquer vers l'extérieur et donc crée du flux de données supplémentaire.

- Liste locale d'adresses IP : ce type de vers, une fois installé, récupère les informations de l'hôte concernant son environnement pour orienter ses futures attaques.

- Passif : ce type de ver est le plus lent à se propager mais le plus discret. Il ne cherche pas de nouveaux hôtes mais attend qu'une machine potentiellement infectable le contact. Il se greffe alors sur les échanges entre sa nouvelle victime et la machine déjà infectée pour se propager. La propagation à travers les réseaux peer2peer illustre parfaitement cette méthode de propagation puisque le ver attends les connexions d'autres machines pour se dupliquer.

Pour réagir face à ce type d'attaque, il a fallu mettre des stratégies de mise à jour en place.

3.2 Etude des différentes méthodes de mise à jour

La seconde partie nécessaire au lancement du projet est une étude des stratégies de protection. Le but dans notre cas précis est de réagir au mieux pour limiter l'infection auprès des clients de France Télécom.

Il faut tout d'abord définir avec précision l'objectif de notre réaction, car face à ce genre d'attaque il existe deux manières de réagir. La première consiste à orienter la protection pour diminuer le nombre d'individus infectés. Ceci répond à une volonté de protéger le maximum d'individu de l'infection. La seconde façon de réagir consiste à limiter la vitesse de propagation au maximum, quitte à avoir plus de machines infectées. Ceci réponds plutôt à une volonté de contrôler la situation, c'est-à-dire ne jamais avoir trop d'individus infectés au même moment. La décision prise a été de suivre la première approche qui consiste à protéger le maximum d'individus, c'est-à-dire à empêcher les vers de contaminer un maximum d'hôte.

Maintenant que notre champ d'action est délimité, il est possible de se consacrer à l'étude de l'existant. Le parallèle avec l'épidémiologie prend ici toute son ampleur. En effet, les enjeux humains lors d'une véritable épidémie sont tellement importants que le sujet a été approfondi par le passé.

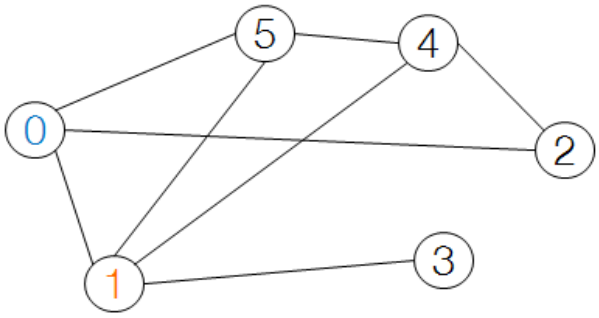
D'un autre côté, les enjeux économiques autour de l'information et notamment de sa virtualisation ont amené les gens à considérer les malwares avec le plus grand sérieux. Il existe aussi une littérature spécifique à la propagation de vers dans un réseau. Beaucoup ont d'ailleurs été motivé par Code Red II et sa propagation fulgurante à travers Internet.

Une première étude sur l'épidémiologie suggère une modélisation en réseau. L'idée principale à retenir est de vacciner les hôpitaux et les écoles en premiers. Ce sont des endroits où il y a beaucoup d'interaction entre les acteurs et avec lesquels la population entretient un lien étroit. A l'aide de la modélisation en graphe, cela revient à vacciner les noeuds de plus haut degrés si chaque nœud représente un individu et si chaque arête représente une mise en relation, un contact, entre deux individus.

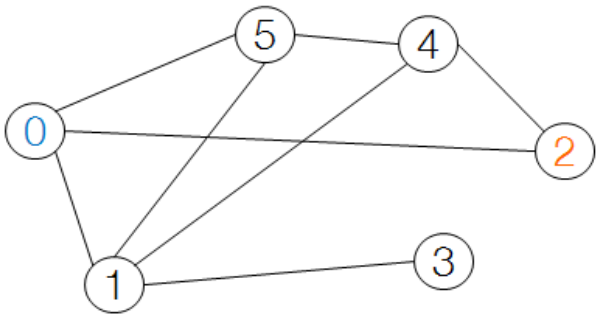
Cette idée importante a été reprise dans beaucoup de stratégie de vaccination dans un réseau informatique. La principale différence est qu'il est parfois difficile de connaître ces points névralgiques. Il a donc fallu adapter le modèle pour essayer de vacciner au mieux dans un graphe aléatoire.

Les méthodes qui sont ressorties de cette étude et qui vont être étudiées par la suite sont les suivantes [4] :

- Aléatoire : on vaccine pseudo-aléatoirement un individu.
- RNB (random neighbor) : on choisi pseudo-aléatoirement un individu et on vaccine un de ses voisins. Avec cette méthode, on tire partie du fait que l'individu que l'on va vacciner a un haut degrés car il est voisin d'un individu quelconque.
- DEG : on choisi aléatoirement un individu puis on vaccine son voisin de plus haut degrés. Par exemple, sur le graphe suivant on choisit le sommet 0 aléatoirement puis on applique la méthode pour trouver le nouvel individu à vacciner.



- OUT : on choisi aléatoirement un individu puis on vaccine le voisin avec lequel il a le moins de voisin en commun. Par exemple, sur le graphe suivant on choisit le sommet 0 aléatoirement puis on applique la méthode pour trouver le nouvel individu à vacciner.



De plus, des variations existent pour certaines de ces méthodes :

- Choisir le niveau du voisin : pour toutes les méthodes sauf la méthode aléatoire, il est possible de ne pas s'arrêter au voisin de niveau 1 mais plutôt de considérer un voisin de niveau n. Ainsi, ce sera un voisin de niveau n qui sera vacciné. Notons les stratégies correspondant DEG_n, RNB_n et OUT_n s'il y a ambiguité.
- Chaîner les vaccination : plutôt que de partir d'un sommet aléatoire à chaque fois, on part du sommet qui vient d'être vacciné si possible. Notons ces stratégies DEGC, RNBC, OUTC.

Le coeur de notre étude sera donc de comparer l'efficacité des ces différentes méthodes et de leurs déclinaisons en fonction du type de propagation.

3.3 Synthèse Propagation/Vaccination

Afin d'étudier des cas concrets et de pouvoir les comparer, nous allons étudier plusieurs couples de propagation/vaccination. Le but de faire une étude comparée est de pouvoir adapter la stratégie de vaccination en fonction de la stratégie de propagation. Comme le stage ne dure que trois mois, nous limiterons les simulations au niveau 1 pour les stratégies de vaccinations cherchant des voisins. On espère tout de même trouver des résultats intéressant qui orienteront les recherches futures.

Nous avons donc deux types de propagations qui représentent les deux stratégies les plus simples mais le plus utilisées : aléatoire et de voisins en voisins. Pour chacune de ces méthodes de propagation, nous confronterons nos sept méthodes de vaccinations (aléatoire, DEG, DEGC, RNB, RNBC, OUT, OUTC). Cela fera donc deux échantillons de sept couples à comparer. En plus d'être intéressante dans notre cas, aucune étude comparée aussi exhaustive n'existe actuellement dans la littérature.

4 Modélisation du problème

La question de la modélisation du problème est essentielle. Il faut trouver un modèle simple pour pouvoir effectuer des simulations sans trop de contraintes de temps (afin de garder un temps de calcul raisonnable). Nous avons fait le choix de modéliser le réseau par un graphe aléatoire. Chaque sommet du graphe représentera une machine et chaque arête un lien entre deux machine. La nature de ce lien n'a pas de réelle importance, il représente juste le vecteur de propagation du ver (cela peut être la classe de l'adresse IP par exemple).

La représentation du graphe choisie est une représentation très utilisée de graphe, c'est-à-dire une matrice NxN où N est le nombre de sommets du graphe (appelée aussi matrice sommet-sommet). Cette matrice M est symétrique et une arête entre le sommet i et j est représentée par un 1 pour les cœfficient m_{ij} et m_{ji} et l'absence d'arête par un 0. Cela revient à l'expression suivante :

$$m_{ij} = \begin{cases} 1 & \text{s'il existe une arête entre i et j} \\ 0 & \text{sinon} \end{cases}$$

La représentation des états est un vecteur V de taille N. Chaque sommets pourra être dans un des trois états suivants : infecté ($v_i = 1$), vacciné ($v_i = 2$), neutre ($v_i = 0$).

Nous allons aussi poser ici toutes les notations qui nous servirons dans toute la suite du rapport.

- Nombre de sommets du graphe : n
- Temps de doublement de population pour un ver : τ. Cela signifie qu'à chaque intervalle de temps valant τ, chaque ver va essayer d'infecter un nouvel hôte. Si à l'instant $\tau - 1$ il y a k machines infectées, à l'instant τ il y aura au plus $2k$ machines infectées.
- Intervalle entre deux vaccinations : λ. Cela signifie qu'à chaque intervalle de temps valant λ, une nouvelle machine est vaccinée.

Il est important de souligner que dans notre modèle, la probabilité que le ver tente d'infecter une adresse qui n'existe pas lors de sa propagation est négligée. Cela signifie qu'à chaque tentative d'infection, il tentera d'attaquer une machine qui existe et qui sera donc dans un état soit susceptible d'être infecté, soit déjà infecté, soit vacciné.

4.1 Modèle Susceptible-Infecté-Supprimé

Lorsqu'il est question de propagation de virus dans une population ou de propagation de virus sur un réseau, deux modèles s'imposent naturellement.

- Le premier modèle est utilisé plutôt dans l'épidémiologie et est le modèle Susceptible-Infected-Susceptible (SIS). Cela signifie que chaque individu passe d'un état à l'autre au cours de l'expérience. Au début, il est susceptible de contracter la maladie, autrement dit il est potentiellement infectable. Une fois infecté, il passe dans l'état I. Dans cet état il est porteur de la maladie et peut la transmettre. C'est à ce moment là qu'il est traité, mais comme pour la plupart des virus, le traitement n'est pas définitif. En effet, nous devons faire des rappels de vaccin pour rester protégé. Ici c'est ce phénomène que l'on veut modéliser. L'individu est protégé avant de repasser dans l'état S où il est susceptible d'être infecté.
- Le second modèle est utilisé dans les réseaux où une mise à jour contre un virus est un traitement définitif. Ce modèle porte le nom de Susceptible-Infected-Removed (SIR) [2]. En effet, une fois qu'une machine est traitée, elle ne pourra plus jamais être infectée. Ceci équivaut à la retirer du réseau des potentiellement infectables, d'où le terme Removed.

Dans notre cas, nous utiliserons le modèle SIR car c'est le modèle qui représente le mieux le système de mise à jour dans un réseau.

4.2 Propagation aléatoire

C'est le cas le plus utilisé par les vers car c'est le cas le plus simple à programmer par les pirates. Dans ce type de modèle, les arêtes entre les sommets du graphes n'ont aucune importance pour la propagation. Ceci équivaut à un graphe complet. Par exemple, ce type de graphe modélise un ver qui a pour stratégie d'infection de tirer une adresse IP

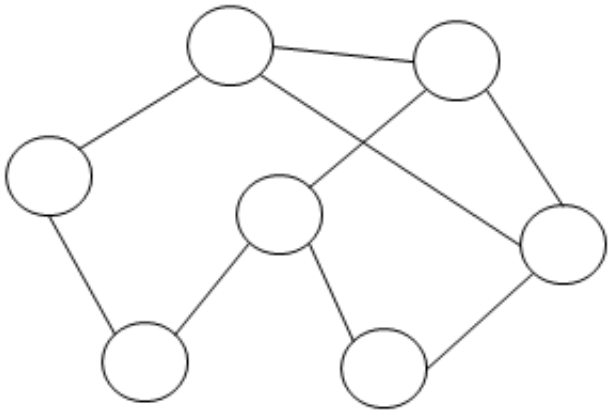
de manière aléatoire. Une fois une adresse tirée, il tente d'infecter la machine concernée. Nous faisons la conjecture que les différentes stratégies de vaccination ont dans ce cas le même effet.

4.3 Propagation non aléatoire

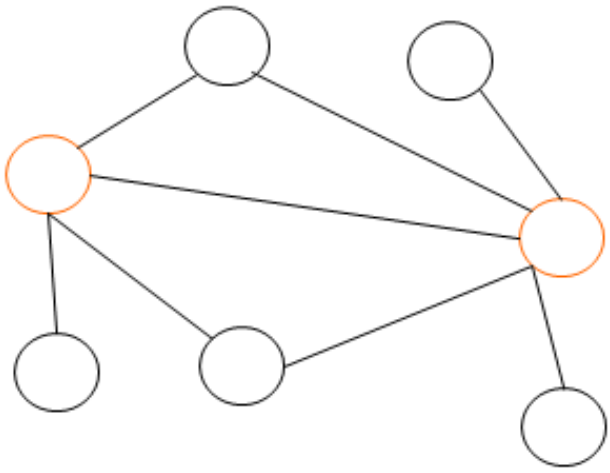
Dans ce cas, les arêtes du graphe ont leur importance puisque le ver va se propager de voisins en voisins. Les arêtes représentent les liaisons entre les machines et ces liens sont susceptibles d'être empruntés par le ver pour se propager dans le réseau. Dans notre cas, ces liens ne seront pas des liens physiques, mais dépendront de la manière dont le ver se propage. Ainsi, selon la topologie du graphe nous pourrons simuler différentes stratégies de propagation.

Cette approche nous amène à considérer les différents types de graphe suivants :

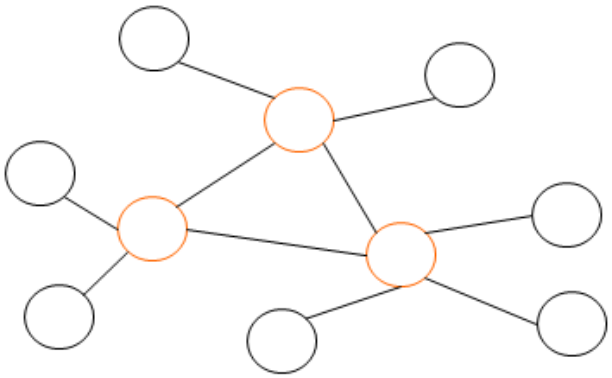
- aléatoire à probabilité uniforme : on ne fait pas de distinction entre les sommets, il y a autant de probabilité d'existence d'une arête entre n'importe quelles paires de sommets.



- aléatoire à probabilité non uniforme : on considère plusieurs classes de sommets. La probabilité d'existence d'une arête dépendra cette fois-ci des classes respectives des deux sommets qu'elle relie.



- aléatoire avec noeuds de raccordement : ce type de graphe représente un type de réseau utilisé dans les télécoms. Il y a peu de machines connectées entre elles qui représentent le fournisseur de service et de nombreux clients qui gravitent autour.



4.4 Graphiques et données à comparer

Le travail à effectuer possède une forte composante exploratoire puisque nous ignorons quels vont être les résultats. Pour conserver un maximum d'informations sur chaque simulation, les données numériques seront stockées dans des fichiers .csv. Les courbes que nous tracerons à partir des ces données nous servirons à visualiser ces données de manière synthétique.

Les paramètres que nous avons décidé d'étudier sont le nombre de vaccinés, le nombre d'infectés, le nombre d'infectés cumulés et la vitesse de propagation, tout ceci en fonction du temps. Nous nous intéresserons aussi au temps nécessaire pour atteindre le maximum d'infection.

5 Implémentation/algorithmique

Les graphes que nous étudierons sont de grande taille. Il est donc important d'optimiser au maximum la gestion de la mémoire pour augmenter la vitesse de calcul. C'est pour cette raison que nous nous sommes tournés vers la librairie Perl Data Langage (PDL) de Perl. Cette librairie permet de mettre toute la puissance du langage Perl à disposition de tableaux de données multidimensionnels. De plus, un vrai travail a été effectué par le créateur de la librairie pour optimiser tous les traitements au cours d'opérations complexes. En effet, cette librairie a été créée à la base par des astronomes pour divers calculs sur des données de grande taille comme le traitement d'image ou le calcul du mouvement des astres.

Pour la partie graphique, PDL possède un module spécial permettant d'interfacer PGPLOT avec la structure de donnée particulière qu'est un piddle (un objet de PDL).

5.1 Vaccination

Lorsque la vaccination est non aléatoire, le topologie du graphe joue un rôle essentiel. La vaccination est dépendante des liens entre sommets et notamment de sa structure. Par exemple, dans le cas où le graphe de départ n'est pas connexe et la stratégie de mise à jour chaînée, elle ne s'effectuera que dans son sous-graphe connexe.

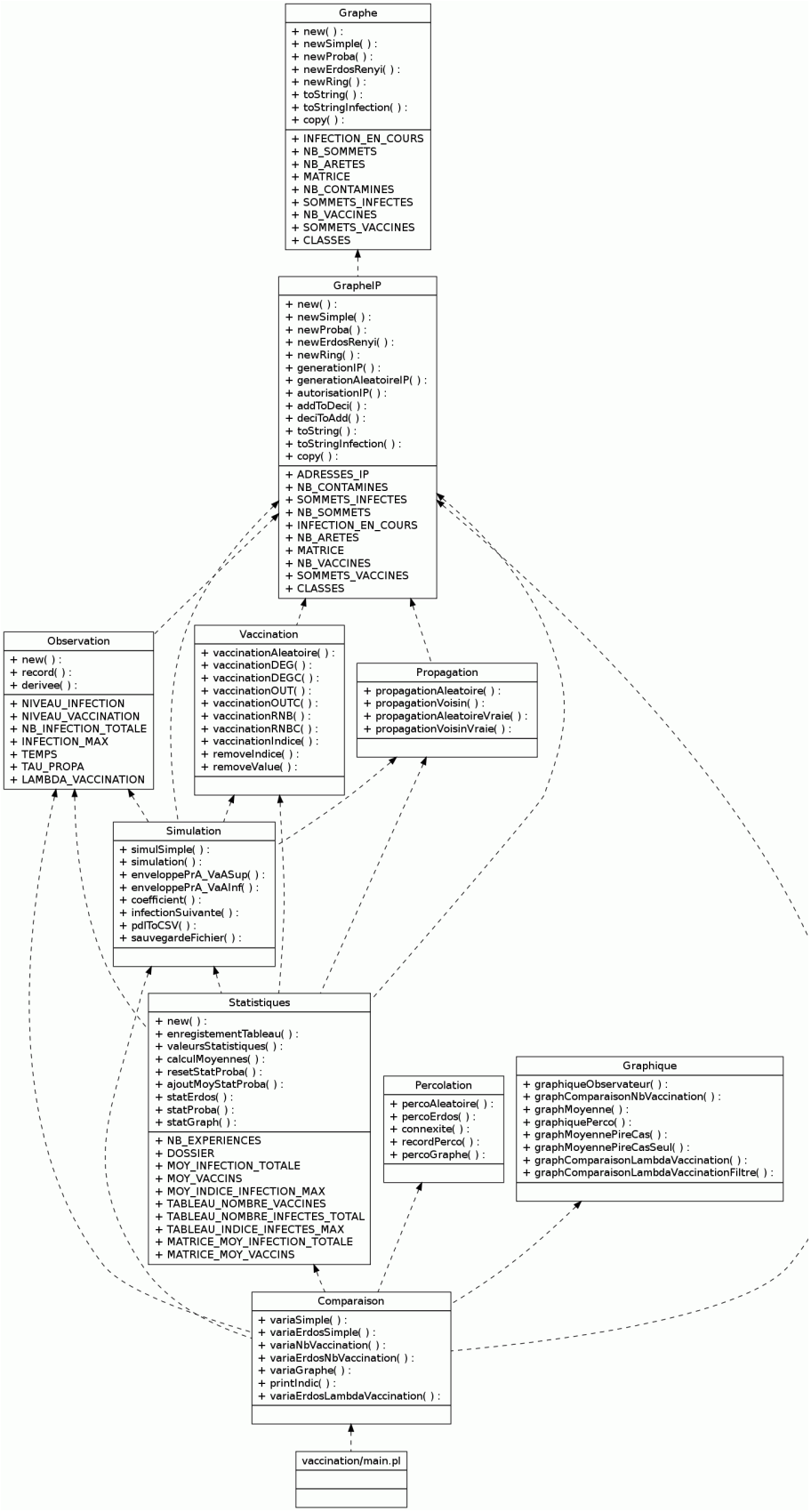
Dans la pratique, la vaccination ne peut être stoppée par la structure du graphe. Si la stratégie de mise à jour ne parvient pas à trouver de nouvelles machines à vacciner, la personne chargée de la mise à jour va recommencer la même stratégie dans un autre endroit du graphe. Afin de rendre compte de ce comportement, nous avons utilisé une métaheurisitique triviale de recherche locale. Cette métaheurisitique consiste à essayer la stratégie pendant un certain nombre de fois, et s'il n'y a eu que des cas d'échec, à recommencer ailleurs.

Dans notre cas particulier, si les tentatives de vaccination ont échoué (c'est-à-dire qu'on essaie de vacciner un individu déjà immunisé) 10 fois de suite, alors le programme effectue une vaccination aléatoire parmi les sommets non vaccinés.

5.2 Présentation des modules

Le choix d'une programmation modulaire a été nécessaire pour une réutilisation ultérieure. En effet, les choix que nous avons dû faire pour limiter notre étude font que notre vision du phénomène de propagation et vaccination est tronquée. Pour une meilleure efficacité face à une propagation de vers, il est nécessaire de tester encore d'autres méthode de vaccinations. De même, nous n'avons pas modélisé tous les types de propagations de vers dans notre étude.

Le diagramme UML suivant montre la répartition du code en module, avec les fonctions de chaque module, les variable de classe et les dépendances entre les modules. Nous avons utilisé Autodia pour générer ce diagramme. Ce logiciel permet de générer des diagramme de classe à partir de fichiers sources pour de multiples langages, et notamment pour PERL. Le seul inconvénient rencontré est qu'il ne prend pas en compte les paramètres des fonctions lors de la création du diagramme.


Graphe
+ new( ) :
+ newSimple( ) :
+ newProba( ) :
+ newErdosRenyi( ) :
+ newRing( ) :
+ toString( ) :
+ toStringInfection( ) :
+ copy( ) :
+ INFECTION_EN_COURS
+ NB_SOMMETS
+ NB_ARETES
+ MATRICE
+ NB_CONTAMINES
+ SOMMETS_INFECTES
+ NB_VACCINES
+ SOMMETS_VACCINES
+ CLASSES
GrapheIP
+ new( ) :
+ newSimple( ) :
+ newProba( ) :
+ newErdosRenyi( ) :
+ newRing( ) :
+ generationIP( ) :
+ generationAleatoireIP( ) :
+ autorisationIP( ) :
+ addToDeci( ) :
+ deciToAdd( ) :
+ toString( ) :
+ toStringInfection( ) :
+ copy( ) :
+ ADRESSES_IP
+ NB_CONTAMINES
+ SOMMETS_INFECTES
+ NB_SOMMETS
+ INFECTION_EN_COURS
+ NB_ARETES
+ MATRICE
+ NB_VACCINES
+ SOMMETS_VACCINES
+ CLASSES
Observation
+ new( ) :
+ record( ) :
+ derivee( ) :
+ NIVEAU_INFECTION
+ NIVEAU_VACCINATION
+ NB_INFECTION_TOTALE
+ INFECTION_MAX
+ TEMPS
+ TAU_PROPA
+ LAMBDA_VACCINATION
Vaccination
+ vaccinationAleatoire( ) :
+ vaccinationDEG( ) :
+ vaccinationDEGC( ) :
+ vaccinationOUT( ) :
+ vaccinationOUTC( ) :
+ vaccinationRNB( ) :
+ vaccinationRNBC( ) :
+ vaccinationIndice( ) :
+ removeIndice( ) :
+ removeValue( ) :
Propagation
+ propagationAleatoire( ) :
+ propagationVoisin( ) :
+ propagationAleatoireVraie( ) :
+ propagationVoisinVraie( ) :
Simulation
+ simulSimple( ) :
+ simulation( ) :
+ enveloppePrA_VaASup( ) :
+ enveloppePrA_VaAInf( ) :
+ coefficient( ) :
+ infectionSuivante( ) :
+ pdfToCSV( ) :
+ sauvegardeFichier( ) :
Statistiques
+ new( ) :
+ enregistementTableau( ) :
+ valeursStatistiques( ) :
+ calculMoyennes( ) :
+ resetStatProba( ) :
+ ajoutMoyStatProba( ) :
+ statErdos( ) :
+ statProba( ) :
+ statGraph( ) :
+ NB_EXPERIENCES
+ DOSSIER
+ MOY_INFECTION_TOTALE
+ MOY_VACCINS
+ MOY_INDICE_INFECTION_MAX
+ TABLEAU_NOMBRE_VACCINES
+ TABLEAU_NOMBRE_INFECTES_TOTAL
+ TABLEAU_INDICE_INFECTES_MAX
+ MATRICE_MOY_INFECTION_TOTALE
+ MATRICE_MOY_VACCINS
Percolation
+ percoAleatoire( ) :
+ percoErdos( ) :
+ connexite( ) :
+ recordPerco( ) :
+ percoGraphe( ) :
Graphique
+ graphiqueObservateur( ) :
+ graphComparaisonNbVaccination( ) :
+ graphMoyenne( ) :
+ graphiquePerco( ) :
+ graphMoyennePireCas( ) :
+ graphMoyennePireCasSeul( ) :
+ graphComparaisonLambdaVaccination( ) :
+ graphComparaisonLambdaVaccinationFiltre( ) :
Comparaison
+ variaSimple( ) :
+ variaErdosSimple( ) :
+ variaNbVaccination( ) :
+ variaErdosNbVaccination( ) :
+ variaGraphe( ) :
+ printIndic( ) :
+ variaErdosLambdaVaccination( ) :
vaccination/main.pl

6 Premiers résultats

6.1 Choix des paramètres

Nous avons essayer de trouver des paramètres qui représentent au mieux le réseau pour nos premières simulations. Nous avons donc sélectionné les paramètres suivants :

- nombre de sommets : n = 500
- temps de doublement : $\tau = 5$
- temps entre deux vaccination : $\lambda = 1$, $\lambda = 3$, $\lambda = 5$ et $\lambda = 8$
- probabilité d'existence d'une arête pour le graphe uniforme : p = 0.05
- probabilité d'appartenance à une classe et matrice des arêtes inter-intraclasse :

$$p_c = \begin{pmatrix} 0.3 \\ 0.6 \\ 0.1 \end{pmatrix} \text{ et } p_{inter-intra} = \begin{pmatrix} 0.025 & 0 & 0.1 \\ 0 & 0.01 & 0.02 \\ 0.1 & 0.02 & 0.15 \end{pmatrix}$$

Ces paramètres permettent de représenter de grand réseaux, uniforme dans un cas, hétérogène dans l'autre, tout en gardant en temps de calcul raisonnable.

6.2 Données numériques

Afin d'avoir un maximum de précision, nous avons conservés les données sous forme de tableau dans un fichier .csv. Grâce à ces tableaux, il est possible d'avoir le maximum d'infection qui correspond à la valeur maximum du nombre d'infectés. Il est aussi intéressant de savoir si ce maximum d'infection est atteint rapidement ou pas. Ce temps d'infection est à comparer à la durée de la décontamination de toute la population.

Voici un extrait du fichier .csv que l'on obtient, centré au maximum d'infection :

Temps	501									
Infection	0	1	1	...	416	421	420	...	1	0
Infection totale	0	1	1	...	436	441	441	...	443	443
Vaccination	0	0	1	...	74	75	76	...	499	500

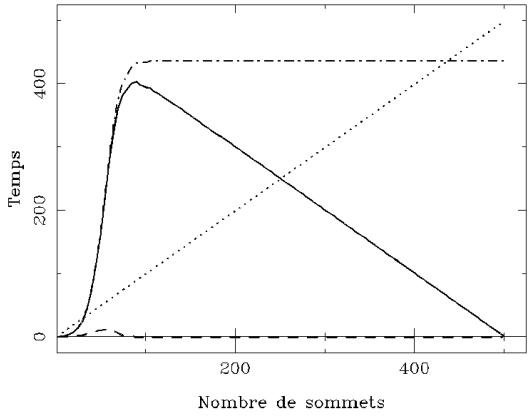
6.3 Analyse graphique

Ces différentes paramètres de simulations nous donnent un premier jeu de données. En effet, pour chaque couple de stratégie de propagation/simulation, nous avons un quatre courbes , une pour chaque valeur de lambda. Vu le nombre conséquent de graphiques obtenus, nous avons décidé de ne présenter que la stratégie de propagation par voisin et

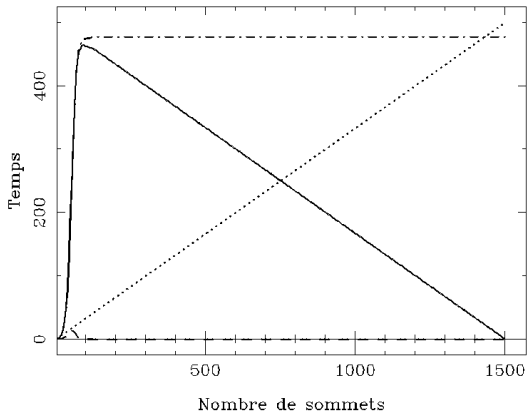
la stratégie de vaccination RNB dans un réseau uniforme.

Voici le jeu des quatre trois courbes obtenues pour les différents paramètres de lambda.

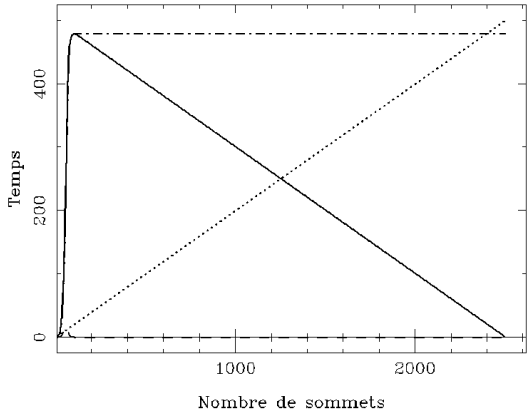
Lambda = 1



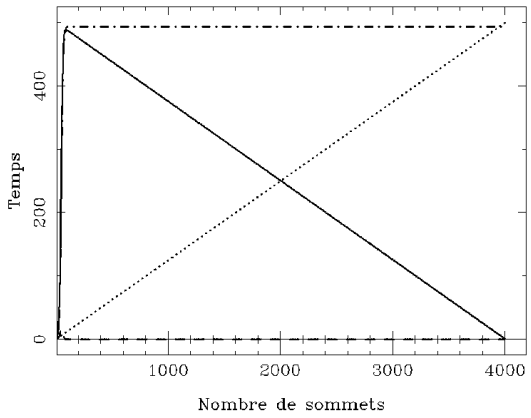
Lambda = 3



Lambda = 5



Lambda = 8



6.4 Direction prise en raison de ces premiers résultats

Ces résultats nous amènent à faire une étude statistique comparée. D'après ce qu'on obtient, le rapport vitesse de vaccination/temps de doublement a une réelle importance sur l'allure des courbes obtenues. De plus, les temps du maximum d'infection ne sont pas atteints au même instant. Il est aussi intéressant de noter que lorsque le temps entre deux vaccination est bien inférieur au temps entre deux vagues de propagation, l'allure des courbe varie beaucoup, alors que lorsque la vaccination est trop lente, il a y peu de changements entre les courbes. Nous allons donc chercher les résultat en fonction du rapport vitesse de vaccination / temps de doublement, mais en nous concentrons sur des valeurs majoritairement élevées.

Une fois les résultats acquis, pour pouvoir les comparer entre eux de manière fiable, nous allons devoir mettre en place des indicateurs. Une étude théorique appuiera la pertinence de nos indicateurs et nous orientera tout au long du reste de l'étude.

7 Etude théorique

7.1 Calcul statistique

Comme énoncé précédemment, nous allons effectuer une étude statistique du problème. Il est important de s'assurer que le nombre d'expériences effectuées soit suffisamment élevé pour que la moyenne observée soit proche de la moyenne théorique.

7.1.1 Hypothèses sur les expériences : expériences indépendantes et identiquement distribuées

Nous allons dans un premier temps effectuer des hypothèses sur notre jeu d'expériences et prouver leur véracité. Ceci est valable à une stratégie de propagation et vaccination fixées, ainsi que pour un type de graphe donné. Pour résumer, nous supposons τ, λ et n fixés. Nous supposons de plus connaître la probabilité qu'une arête existe entre chaque paire de sommets du graphe.

- Toutes nos expériences sont indépendantes : pour chaque nouvelle expérience, nous générons un nouveau graphe de manière pseudo-aléatoire selon une distribution uniforme et nous choisissons aussi un sommet de manière aléatoire comme foyer de la propagation du vers. Cette manière de procéder nous assure de l'indépendance entre deux expériences.
- Toutes nos expériences sont identiquement distribuées : notre échantillon contiendra les résultats obtenus à une stratégie de vaccination et propagation fixée, ainsi que pour un type de graphe aléatoire donné. Cela signifie que pour chaque expérience, le graphe aléatoire sera généré sur le même modèle et que la propagation tout comme la vaccination suivront la même stratégie à la même vitesse. Cette manière de procéder nous assure une distribution identique pour chaque expérience.

Nous pouvons donc conclure que nos expériences sont bien indépendantes les unes des autres qu'elles sont identiquement distribuées (i.i.d.).

Nous pouvons même aller plus loin en disant que notre échantillon est uniformément distribué.

7.1.2 Calcul de l'erreur

Afin de savoir si notre expérience reflète suffisamment bien la théorie, il est intéressant d'avoir un indicateur de l'erreur. Nous allons donc essayer de quantifier cette erreur pour permettre une représentation correct du problème. Nos expériences suivant une loi uniforme et en appliquant le théorème Central Limite (ce qui est possible car notre échantillon est i.i.d.), nous pouvons majorer l'erreur de la manière suivante :

Soit N la taille de l'échantillons et X_n les observations. La moyenne observée μ_N et la variance observée σ_N^2 valent respectivement :

$$\mu_N = \frac{1}{N}\sum_{n=1}^{N} X_n$$

et

$$\sigma_N^2 = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu_N)^2$$

Par le théorème Central Limite on a :

$$\sqrt{N}\frac{|\mu_N - \mu|}{\sigma_N} \sim N(0,1)$$

On en déduit l'intervalle de confiance suivant :

$$|\mu_N - \mu| \leq z_{1-\frac{\alpha}{2}}\frac{\sigma_N}{\sqrt{N}}$$

Soit :

$$|e_N| \leq z_{1-\frac{\alpha}{2}}\frac{\sigma_N}{\sqrt{N}}$$

où $|\mu_N - \mu| \leq |e_N|$ est la majoration de l'erreur pour l'intervalle de confiance $(1-\alpha)$ et où $z_{1-\frac{\alpha}{2}}$ est le quantile pour la loi normale centrée réduite.

D'après la table des quantiles calculée avec la limite de la loi de Student, si on veut un niveau de confiance de 95% (c'est-à-dire $\alpha = 0,05$) il faut prendre $z_{1-\alpha} = z_{0,095} = 1,960$, d'où :

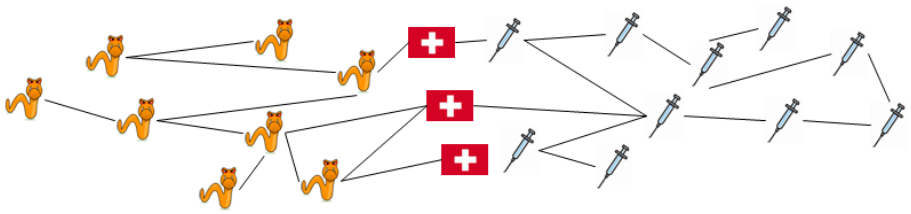
$$|e_N| \leq 1,960\frac{\sigma_N}{\sqrt{N}}$$

Nous allons nous tenir à ce niveau de confiance dans la suite de nos calculs statistiques.

7.2 Courbe maximale

Il est intéressant en préliminaire au reste de l'étude de chercher la courbe correspondant au pire des cas lors d'une propagation. Le pire des scénarios est que la vaccination ne permette ni de soigner des individus malades (vaccination d'individus sains), ni d'empêcher la progression de la propagation (chaque tentative d'infection est une infection réussie). Ce scénario se poursuit jusqu'à ce que tous les sommets soient soit infectés soit vaccinés. Il sera donc par la suite nécessaire de vacciner toute la population pour éradiquer l'épidémie.

On peut imaginer que la propagation se produit d'un côté du graphe, tandis que la vaccination se produit de l'autre côté. L'ensemble des éléments susceptibles d'être infectés diminue à chaque étape jusqu'à ce qu'il n'y en ait plus.



Cette courbe du pire des cas correspond au minimum entre la courbe de propagation sans vaccination et la courbe de potentiellement infectable. Nous avons besoin des équations de chacune de ces courbes pour nos calculs.

Les potentiellement infectables (dans le cas discret) ont pour équation [1] :

$$i_d(t) = n - E\left(\frac{t}{\lambda}\right)$$

La propagation sans vaccination (dans le cas discret) a pour équation :

$$p_d(t) = 2^{E\left(\frac{t}{\tau}\right)}$$

Pour plus de simplicité dans les calculs, nous allons lisser ces courbes en les dédiscrétisant.

La propagation sans vaccination dans le cas continue a pour équation :

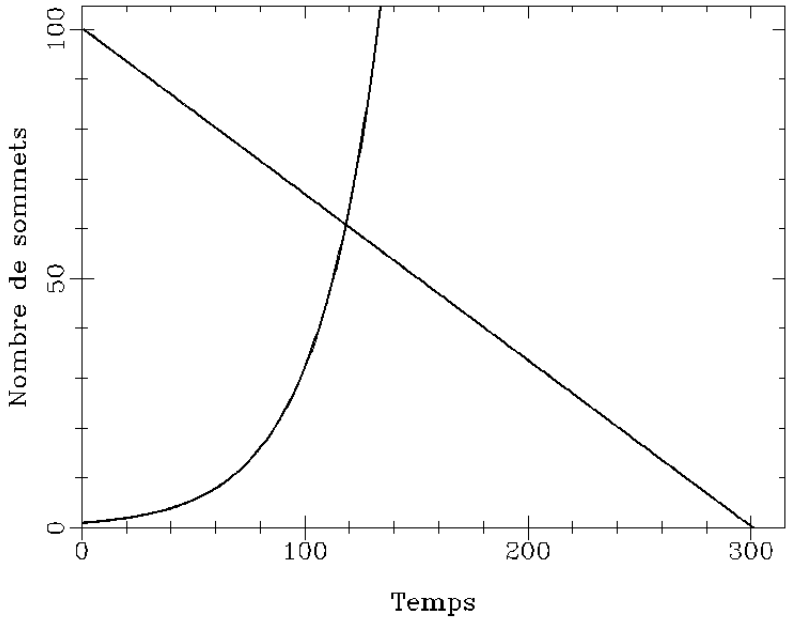
$$p(t) = 2^{\frac{t}{\tau}} = e^{\frac{tln2}{\tau}}$$

Les potentiellement infectables dans le cas continue ont pour équation :

$$i(t) = n - \frac{t}{\lambda}$$

Le graphique représentant ces deux courbes est le suivant a l'allure suivante :

[1] ici et dans la suite, E(x) fait référence à la partie entière de x



Le point qui va nous intéresser par la suite est l'intersection entre ces deux courbes. Il représente le nombre maximum de machine que le ver peut infecter lors de sa propagation. Appelons t_{max} l'abscisse de ce point qui est le maximum de la courbe. t_{max} est la solution de l'équation $p(t) = i(t)$. Résolvons cette équation pour $t > 0$:

$$\begin{aligned} & p(t) = i(t) \\ \Leftrightarrow \quad & n - \frac{t}{\lambda} = 2^{\frac{t}{\tau}} \\ \Leftrightarrow \quad & n - e^{\frac{t}{\tau} ln2} - \frac{t}{\lambda} = 0 \end{aligned}$$

Ce type d'équation n'a pas de solution sous forme close. Pour avoir une évaluation de ce point, il nous faut utiliser la décomposition de l'exponentielle en série entière afin d'avoir un polynôme et approximer ce point en cherchant la racine de ce polynôme. De plus, grâce à cette décomposition, nous aurons aussi une majoration de l'erreur.

$$
\begin{aligned}
& n - e^{\frac{t}{\tau}ln2} - \frac{t}{\lambda} = 0 \\
\Leftrightarrow \quad & n - \frac{t}{\lambda} - \sum_{k=0}^{\infty} \frac{(\frac{tln2}{\tau})^k}{k!} = 0 \\
\Leftrightarrow \quad & n - \frac{t}{\lambda} - \sum_{k=0}^{\infty} \left(\frac{ln2}{\tau}\right)^k \frac{t^k}{k!} = 0 \\
\Leftrightarrow \quad & n - \frac{t}{\lambda} - 1 - \frac{tln2}{\tau} - \sum_{k=2}^{\infty} \left(\frac{ln2}{\tau}\right)^k \frac{t^k}{k!} = 0 \\
\Leftrightarrow \quad & n - 1 - t\frac{\tau - \lambda ln2}{\lambda\tau} - \sum_{k=2}^{\infty} \left(\frac{tln2}{\tau}\right)^k \frac{1}{k!} = 0
\end{aligned}
$$

En faisant un développement limité à partir de cette série entière à un indice $m \geq 2$ donné, nous somme dans le cas du calcul de racine d'un polynôme de degrés m. En outre, nous avons une majoration de l'erreur pour le développement limité car la série converge (la série de l'exponentielle converge, et ici nous avons la série de l'exponentielle moins un nombre de fini de terme, donc la série converge). L'expression de ce majorant est :

$$\varepsilon \leq \frac{t^m}{m!}$$

Il est donc possible de calculer une valeur approchée de t_{max} à une erreur ε donnée. Cependant, cette erreur ne converge rapidement que proche de 0 (en effet, pour que cette mesure de l'erreur diminue, il faut $m \geq t$). Ici, t_{max} est de l'ordre de plusieurs dizaine de sommets, c'est à dire que pour calculer une valeur approchée suffisamment précise de t_{max}, il faut résoudre des polynômes de degrés très élevés. Pour résumer, nous pouvons dire que :

- Comme nous ne connaissons pas t_{max}, il est impossible de faire un changement de variable en t_{max}.
- ε est borné par $\frac{t_{max}^m}{m!}$ en t_{max}. Pour avoir une erreur inférieur à 1, il nous faut $m \geq t_{max}$. Ceci revient à résoudre un polynôme de degrés supérieur à t_{max}.

Ceci ne correspond pas à ce que nous recherchons, nous n'exploiterons donc pas cette méthode pour trouver une valeur approchée de t_{max}.

Nous pouvons par contre tirer la confirmation que le nombre maximum d'infectés à un lien direct avec le rapport temps de vaccination/temps de propagation. En effet :

Prenons $\lambda' = k\lambda$ et $\tau' = k\tau$ et montrons que le nombre maximum d'infectés n_{max} est le même.

On a t_{max} solution de :

$$n - e^{\frac{t}{\tau} ln2} - \frac{t}{\lambda} = 0 \tag{1}$$

On note alors t'_{max} la solution de

$$n - e^{\frac{t}{\tau'} ln2} - \frac{t}{\lambda'} = 0 \tag{2}$$

En prenant $t'_{max} = kt_{max}$ on tombe sur l'égalité :

$$n - e^{\frac{t_{max}}{\tau} ln2} - \frac{t_{max}}{\lambda} = 0$$

qui est vrai puisque t_{max} est solution de 1. On peut donc affirmer que $t'_{max} = kt_{max}$ est une solution de l'équation 2, et que cette solution est unique car la fonction $n - e^{\frac{t}{\tau'} ln2} - \frac{t}{\lambda'}$ est strictement monotone.

Nous avons donc :

$$\begin{aligned} n'_{max} &= n - \frac{t'_{max}}{\lambda'} \\ &= n - \frac{kt_{max}}{k\lambda} \\ &= n - \frac{t_{max}}{\lambda} \\ &= n_{max} \end{aligned}$$

Nous pouvons en conclure que seul le rapport entre le temps de propagation et le temps de vaccination est important pour mesurer le nombre maximum d'infectés.

7.3 Courbe de probabilité

A chaque instant t, il y a une probabilité $P_i(t)$ qu'un sommet soit infecté ou non. Pour rester cohérent, nous allons chercher cette probabilité dans le cas que nous avons décrit comme cas de référence, c'est-à-dire une propagation aléatoire et une vaccination aléatoire. Nous avons décidé de calculer cette probabilité en fonction de t. Pour trouver cette probabilité, nous considérons trois ensembles :

- l'ensemble des sommets infectés à l'instant t $C_i(t)$
- l'ensemble des sommets vaccinés à l'instant t $C_v(t)$
- l'ensemble des sommets dans leur état d'origine à l'instant t $C_p(t)$

Sur ces trois ensembles, nous savons que :

- $C_v(t) = E\left(\frac{t}{\tau}\right)$
- $C_p(t) = n - C_i(t) - C_v(t)$
- $P_i(t) = \frac{C_i(t)}{n}$

Ce que nous voulons trouver dans un premier temps, c'est l'espérance du nombre d'individus infectés à l'instant t. Pour plus de simplicité, nous assimilerons $C_v(t)$ qui est normalement un entier à son espérance.

Pour trouver cette espérance, nous allons d'abord nous placer dans un cas simple puis le généraliser. Supposons qu'à l'instant t_i il y ait une et une seule infection et non un doublement de population comme il se produit dans le pire des cas (on garde en considération qu'il puisse y avoir une vaccination entre t_i et $t_i + 1$). Nous avons alors la relation suivante :

- $C_i(t_i + 1) = C_i(t_i) + \frac{C_p(t_i}{n}$
- $C_i(t_i + 1) = C_i(t_i) + 1 - \frac{C_i(t_i)+C_v(t_i)}{n}$

Notons f la fonction associée à cette propagation d'un seul élément : $f : (x, y) \mapsto x + 1 - \frac{x-C_v(y-1)}{n}, \forall (x, y) \in R^2$. Nous pouvons alors modéliser un doublement de population par des infections successives d'un seul élément. Dans ce but, nous définissons la notation f^k à deux variables par $f^k(x, y) = f(f(\dots f(x, y), y), y \dots, y), \forall k \in N$

L'expression de $C_i(t + 1)$ en fonction de $C_i(t)$ est donc la suivante :

$$C_i(t + 1) = f^k(C_i(t), t + 1)$$

où k est le nombre d'individu infectés à l'instant t.

Un premier problème nous apparaît, que nous ne pourrons résoudre que par des approximations. En effet, k doit être un entier, alors que l'espérance du nombre d'individus infectés à l'instant t ne l'est pas forcément. A chaque étape nous devrons donc approximer la partie $C_i(t)$ par sa partie entière $E(C_i(t))$ ou par sa partie entière $+1$ $E(C_i(t)) + 1$. Nous espérons ainsi obtenir deux courbes qui envelopperont nos observations dans cette stratégie de propagation et de vaccination.

En développant les calculs, nous obtenons l'expression suivante (et en considérant la partie entière de $C_i(t)$) :

$$C_i(t+1) = C_i(t) - (C_v(t+1) - C_v(t))\frac{C_i(t)}{n} + \left(E\left(\frac{t+1}{\tau}\right) - E\left(\frac{t}{\tau}\right)\right) * \left(E(C_i(t)) - \sum_{i=0}^{E(C_i(t))-1} \frac{i}{n} + \sum_{j=0}^{E(C_i(t))-2} \sum_{i=1}^{E(C_i(t))-j-1} \left(-\frac{1}{n}\right)^{E(C_i(t))-i-j+1} ij - \frac{(C_i(t) + C_v(t))}{n} - \left(E(C_i(t)) - \sum_{i=0}^{E(C_i(t))-1} \frac{i}{n} + \sum_{j=0}^{E(C_i(t))-2} \sum_{i=1}^{E(C_i(t))-j-1} \left(-\frac{1}{n}\right)^{E(C_i(t))-i-j+1} ij\right)\right)$$

Posons à n fixé :

$$g : (x, y) \mapsto 1 - \frac{x+y}{n}$$

et

$$a_k = k - \sum_{i=0}^{k-1} \frac{i}{n} + \sum_{j=0}^{k-2} \sum_{i=1}^{k-j-1} \left(-\frac{1}{n}\right)^{k-i-j+1} ij, \forall k \geq 2$$

avec $a_1 = 1$.

Maintenant, nous avons notre expression approchée de $C_i(t)$ par récurrence.

$$C_i(t+1) = C_i(t) - \left(1 - \frac{C_v(t+1) - C_v(t)}{n}\right) + \left(E\left(\frac{t+1}{\tau}\right) - E\left(\frac{t}{\tau}\right)\right) * \left(a_{E(C_i(t))} - \frac{(C_i(t) + C_v(t))a_{E(C_i(t))}}{n}\right)$$

C'est cette expression que nous utiliserons pour la courbe minimale de nos courbes enveloppes. La courbe enveloppe supérieure sera la même, en changeant $E(C_i(t))$ par $E(C_i(t)) + 1$.

Par la suite nous allons comparer ces deux courbes de $C(t)$ obtenus à nos simulations. Nous espérons que ces deux courbes enveloppes seront de véritables courbes enveloppes malgré les approximations (c'est-à-dire que nos simulations seront toujours entre les deux courbes). Nous espérons aussi que ces deux courbes seront proches de la simulation, car si elle en sont trop éloignées nous ne pourrons en tirer aucune information.

7.4 Percolation

La percolation est un phénomène physique connu depuis 1957. Il consiste en un changement d'état à partir d'une certaine valeur de seuil. Le phénomène tire son nom du passage

de l'eau à travers le percolateur de la machine à café. [6]

Un parallèle a été fait dans l'épidémiologie. En effet, il existe une certaine valeur critique appelée seuil de percolation, à partir de laquelle la propagation d'un virus dans une population est très rapide. La façon de vacciner est faite de sorte à maintenir le nombre de personne contaminées en dessous de ce seuil de percolation en vaccinant d'abord les gens fréquentant les hôpitaux ou les écoles.

Comme l'étude de la vaccination dans une population et les stratégies de mise à jour de machines dans un réseau infecté sont très proches, nous pouvons transposer cette idée de percolation à notre problème. Nous définirons le seuil de percolation comme le plus petit nombre de sommets à enlever dans un graphe pour avoir une chance sur deux de déconnecter le graphe. Autrement dit, notre seuil de percolation est
$\min_k(\mathbb{P}(G - S\text{non connexe avec}|S| = k) > 0,95)$. La stratégie qui apparaît la plus simple pour trouver ce seuil est d'enlever des sommets de manière aléatoire dans un graphe donné et tester la connexité à chaque sommet enlevé.

Afin de s'approcher au mieux du résultat théorique, nous allons faire une étude statistique de la percolation. Pour cela nous allons générer des graphes aléatoires avec les mêmes caractéristiques et étudier la déconnexion du graphe. La mesure de l'erreur est donnée par l'étude sur la loi normale effectuée précédemment. Nous conservons donc le même intervalle de confiance.

Il ne faut pas perdre de vue non plus que les ressources informatiques dont nous disposons sont limitées. Pour optimiser la vitesse de calcul, nous avons mis au point un algorithme particulier qui trouve le sommet recherché en un seul parcours du graphe. Cela permet de ne pas avoir à tester la connexité à chaque sommet enlevé. L'algorithme décrit ci-après se base sur une hypothèse importante et vérifiée dans notre cas : enlever les sommets dans un ordre prédéfini dans un graphe aléatoire équivaut à les enlever aléatoirement dans un graphe aléatoire.

En effet, si nous décomposons ce qu'il se passe en deux étapes, nous obtenons ceci :

- Génération d'un graphe de manière pseudo-aléatoire
- Génération d'un ordre de manière pseudo-aléatoire

Ce qui est important est que chez nous, ces deux étapes sont indépendantes et donc commutent. La génération du graphe n'a donc pas d'influence sur l'ordre choisi et inversement. Dans ces conditions, nous pouvons choisir de ne faire varier qu'une seule des deux étapes de manière pseudo-aléatoire. Nous avons fait le choix ici de nous cantonner à la génération du graphe de manière pseudo-aléatoire à un ordre fixé.

7.4.1 Algorithme

Nous allons décrire ici l'algorithme utilisé pour trouver le seuil de percolation. Le principe est simple, puisqu'il s'agit d'enlever des sommets les uns après les autres du graphe de départ. A la fin de cette algorithme, nous obtenons le premier sommet enlevé qui déconnecte le graphe, en suivant un ordre prédéfini. Cet algorithme n'a d'intérêt que lorsqu'il est utilisé à plusieurs reprises en changeant l'ordre à chaque fois.

Description de l'algorithme en boîte noire :

Input : G(S,A) avec $|S| = n \in N, S = (s_1, s_2, \ldots, s_n)$
Output : k_0, avec k_0 valant l'indice du premier sommet qui déconnecte le graphe si on les enlèves en commençant par s_n et en finissant par s_1, 0 si le graphe G n'est pas connexe.

Maintenant que les entrées/sorties de l'algorithme est décrit, posons les notations qui nous servirons par la suite :

- σ une permutation de $\{1, \ldots, n\}$ telle que :
  - $\sigma(1) = 1$
  - $\forall i \in \{2, \ldots, n\}, \sigma(i) = min\ \{k \in \mathbb{N}; s_k \in N(S_{i-1})\}$
- On note $S_1 = s_{\sigma(1)} = s_1$ et $S_k = s_{\sigma(1)} \cup \ldots \cup s_{\sigma(k)}, \forall k \in [1, n]$
- On note $N(s)$ les voisins du sommet s et $N(S)$ l'intersection entre les voisins des sommets de S et le complémentaire de S (noté $\bar{S}$) : $N(S) = (N(S_1) \cup N(S_2) \ldots \cup N(S_m)) \cap \bar{S}$.

Déroulement de l'algorithme :

Au début on pose $k_0 = 0$.
Pour k = 1 à n-1

- si $N(S_k) = \emptyset$ alors $k_0 = 0$ puis STOP
- sinon $S_{k+1} = S_k \cup s_{\sigma(k)}$ et si $k > k_0$ ET $k \neq \sigma(k)$ alors $k_0 = k$

7.4.2 Preuve de l'algorithme

Pour nous assurer que cet algorithme fonctionne, nous avons décidé de prouver mathématiquement cet algorithme. Le prouver nous assurera non seulement de sa justesse, mais aussi de sa terminaison. Pour prouver cet algorithme, il nous faut démontrer quelques propriétés et lemmes qui en découlent, en supposant que le graphe de départ est connexe

(en effet, s'il ne l'est pas, notre algorithme retournera toujours la même chose) :

Propriété 1 : terminaison de l'algorithme.

Démonstration. Cet algorithme termine car, comme le nombre de sommets du graphe est fini, il existe k tel que $N(S_k) = \emptyset$. A chaque itération, le nombre de sommet du graphe de départ diminue, donc au pire des cas en n-1 itération, nous avons un graphe réduit à un seul sommet, c'est-à-dire que nécessairement $N(S_n) = \emptyset$.

□

Propriété 2 : A la fin de l'algorithme, $k_0 = max\ \{i \leq n; i \neq \sigma(i)\}$.

Démonstration. Comme le graphe de départ est supposé connexe, $\exists k$ tel que $1 \leq k \leq n$ vérifiant $k \neq \sigma(k)$ (propriété nécessaire pour affecter cette valeur à k_0). D'un autre côté, comme notre algorithme impose la condition $k > k_0$ pour affecter une nouvelle valeur à k_0, k_0 est toujours le plus grand indice vérifiant cette propriété. A fortiori, cette propriété est vraie à la fin de l'algorithme, c'est-à-dire lorsque $k = n$. De plus, i étant la dernière valeur de k_0, on a bien $i \neq \sigma(i)$.

□

Propriété 3 : $\forall k, S_k$ est connexe.

Démonstration. Par récurrence :

- Initialisation : $S_1 = s_1$ donc S_1 est connexe
- Récurrence : $\forall k \in [1, n]$, supposons que S_k est connexe et montrons que S_{k+1} l'est aussi. On a $S_{k+1} = S_k \cup s_{\sigma(k)}$. Or $s_{\sigma(k)} \in N(S_k)$. Comme S_k est connexe, lui rajouter un voisin d'un de ses sommets donne un nouveau graphe qui reste connexe. Ainsi, on a bien S_{k+1} qui est connexe.

Nous pouvons conclure grâce à cette démonstration par récurrence que S_k connexe $\forall k$.

□

Propriété 4 : le 1er sommet qui déconnecte le graphe est s_{k_0} si on les enlève dans l'ordre $s_n, \ldots s_1$.

Nous posons une nouvelle notation : $\tilde{S}_k = s_1, \ldots, s_k$ Il en découle deux lemmes que nous allons montrer et qui reviennent à la propriété 4 :

Lemme 1 : $\tilde{S}_{k_0}$ est connexe.

Démonstration. k_0 vérifie $max\ \{i \leq n; i \neq \sigma(i)\}$, donc on a $\forall k > k_0, s_k = s_{\sigma(k)}$. De plus, on a :

$$\begin{aligned} & \tilde{S}_{k_0} = \{s_1, \ldots, s_{k_0}\} \\ \Leftrightarrow\ & \tilde{S}_{k_0} = S - \bigcup_{k=k_0+1}^{n} s_k \\ \Leftrightarrow\ & \tilde{S}_{k_0} = S - \bigcup_{k=k_0+1}^{n} s_{\sigma(k)} \text{ car } k > k_0 \\ \Leftrightarrow\ & \tilde{S}_{k_0} = S_{k_0} \end{aligned}$$

Or d'après la propriété 3, S_{k_0} est connexe donc $\tilde{S}_{k_0}$ est connexe.

□

Lemme 2 : $\tilde{S}_{k_0-1}$ n'est pas connexe.

Démonstration. On rappelle que $\tilde{S}_{k_0-1} = \{s_1, \ldots, s_{k_0-1}\}$ et $S_{k_0-1} = \{s_{\sigma(1)}, \ldots, s_{\sigma(k_0-1)}\}$.

En démontrant le lemme 1, nous avons montré que $S_{k_0} = \tilde{S}_{k_0}$. On en déduit que $S_{k_0-1} \neq \tilde{S}_{k_0-1}$ (car par définition de k_0, $\sigma(k_0) \neq k_0$).

D'un autre côté, comme $s_{k_0} \in S_{k_0}$ et $s_{\sigma(k_0)} \neq s_{k_0}$, alors $s_{k_0} \in S_{k_0-1}$. On note alors i l'indice tel que $S_i = S_{i-1} \cup s_{k_0}$.

Prenons $A = \{s_k; k < k_0\} \cap \bar{S}_{i-1}$. Démontrer que $\tilde{S}_{k_0-1}$ n'est pas connexe revient maintenant à démontrer qu'il une partition en deux sous-ensembles non vides distincts de $\tilde{S}_{k_0-1}$ qui ne sont pas relié par des arêtes. :

1. $N(S_{i-1}) \cap A = \emptyset$
2. $A \cup S_{i-1} = \tilde{S}_{k_0-1}$
3. $A \neq \emptyset$ et $S_{i-1} \neq \emptyset$

Montrons 1 par l'absurde :

Supposons qu'il existe un sommet de A, par exemple s_j tel que $s_j \in N(S_{i-1})$. Alors comme $j < k_0$ (puisque $s_j \in A$), on a un sommet d'indice inférieur à
$k_0 = min\ \{k; s_k \in N(S_{i-1})\}$ ce qui est absurde. Donc $N(S_{i-1}) \cap A = \emptyset$.

Montrons 2 :

$$\begin{aligned} A \cup S_{i-1} &= (\{s_k; k \leq k_0 - 1\} \cap \bar{S}_{i-1}) \cup S_{i-1} \\ &= \{s_k; k \leq k_0 - 1\} \cup S_{i-1} \\ &= \{s_k; k \leq k_0 - 1\} \\ &= \tilde{S}_{k_0-1} \end{aligned}$$

Montrons 3 :

- $i > 0$ donc $s_0 \in S_{i-1}$ c'est-à-dire $S_{i-1} \neq \emptyset$
- $k_0 \in A$ donc $A \neq \emptyset$

1, 2 et 3 assurent $\tilde{S}_{k_0}$ non connexe.

□

D'après les lemmes 1 et 2, nous avons $\tilde{S}_{k_0}$ connexe alors que $\tilde{S}_{k_0-1} = \tilde{S}_{k_0} - s_{k_0}$ non connexe. Nous pouvons donc conclure que la Propriété 4 est vérifiée, c'est-à-dire que le premier sommet qui déconnecte le graphe est s_{k_0} si on enlève les sommets dans l'ordre $s_n, \ldots, s_1$.

Propriété 5 : la complexité de l'algorithme est en O(n + |E| log n).

Démonstration. A chaque itération on effectue les opérations suivantes :

- on teste si $N(S_k) = \emptyset$. Dans le cas d'une structure en liste chaînée, la complexité de cette comparaison est $O(1)$.
- on calcule $s_{\sigma(k)}$. Il faut dans un premier calculer $\sigma(k) = min\ \{i; s_i \in N(S_k)\}$. Avec la bonne structure de données, la complexité est en $O(1)$. Ensuite il faut mettre à jour $N(S_k)$ de la manière suivante : $N(S_{k+1}) = N(S_k) - s_{\sigma(k)} \cup N(s_{\sigma(k)})$. La complexité de l'opération $N(S_k) - s_{\sigma(k)}$ est en $O(1)$ tandis que la complexité de $N(S_k) \cup N(s_{\sigma(k)})$ est en $d(s_{\sigma(k)} * log\ \big(N(S_k) \cup N(s_{\sigma(k)})\big)$.
- on rajoute un sommet à un ensemble de sommets : $S_{k+1} = S_k \cup s_{\sigma(k)}$. L'opération d'ajout a une complexité en $O(1)$.
- on affecte une nouvelle valeur à k_0 si les comparaisons $k > k_0$ et $k \neq \sigma(k)$ sont concluantes. Ces opérations ont une complexité en $O(1)$.

La complexité de l'ensemble de l'algorithme est donc la suivante :

$$\begin{aligned} C &= O(n) + \sum_{k=1}^{n} O(k) + d(s_{\sigma(k)}) * log\left(N(S_k) \cup N(s_{\sigma(k)})\right) \\ &\leq O(n) + \sum_{k=1}^{n} O(k) + d(s_{\sigma(k)}) * log(n) \\ &\leq O(n) + log(n) * |A| \\ &= O(n + |A|log(n)) \end{aligned}$$

□

7.4.3 Intérêt de l'algorithme

Le principal intérêt de cet algorithme est sa faible complexité en terme de temps de calcul. Nous travaillons sur des jeux de données de grande taille, il est donc important de pouvoir minimiser le temps de calcul. Cet algorithme nous permet d'effectuer des calculs en $O(n + |A|log(n))$, valeur que l'on peut comparer à $O(n^2 + n|A|)$, temps de calcul que nous aurions sans cet algorithme. [2]

Il faut rappeler que ceci est possible car enlever des sommets dans un ordre aléatoire d'un graphe aléatoire revient au même que d'enlever des sommets de manière ordonnée dans un graphe aléatoire.

8 Simulation statistiques

Nous avons effectué un grand nombre de simulations pour obtenir des résultats rigoureux. Afin de pouvoir faire suffisamment de calculs en un temps raisonnable nous avons choisi de prendre des graphes à 250 sommets, $\tau = 10$ et λ variant de 1 à 15. De même, nous avons fixé le nombre de simulations à paramètres fixés à 15 (valeur choisie après plusieurs tests pour avoir un erreur raisonnable à un niveau de confiance de 95%). Cela signifie que nous allons faire s'affronter 15 fois chaque couple de stratégie avant de calculer la moyenne.

Nous avons choisi de présenter uniquement les courbes correspondant à la stratégie de vaccination aléatoire et la stratégie de vaccination RNB, face à une propagation par voisin. En effet, la stratégie aléatoire est un cas de base à étudier pour pouvoir se situer par rapport à lui, tandis que la stratégie de vaccination RNB est une stratégie intéressante car assez simple à mettre en œuvre (donc demander peu de temps pour le calcul) mais tout

[2] Cette complexité correspond à la méthode qui consiste à enlever les sommets un par un et à tester la connexité à chaque fois

de même efficace (en général, elle se situe dans les 5 meilleures solutions). La simplicité de cette méthode se situe en grande partie dans le fait qu'il n'y a besoin de connaître que le voisinage direct pour l'appliquer. En se projetant dans une utilisation réelle, c'est une technique plus facilement applicable. De même, nous avons sélectionné des valeurs de lambda représentative : 1, 3, 5, 10, 15. Ces valeurs nous donnerons une bonne idée de l'évolution des différentes courbes en fonction de lambda.

8.1 Cas de base : Vaccination Aléatoire

La vaccination aléatoire est utilisée actuellement par France Télécom pour mettre à jour son parc de LiveBox. Pour cette raison, ce cas constituera notre mesure de référence.

8.1.1 Propagation Aléatoire

Dans ce cas la structure du graphe importe peu. En effet, nous avons vu tout à l'heure que dans ce cas, il n'y a pas de stratégie de propagation particulière et donc pas de possibilité de contrer cette stratégie véritablement.

L'intérêt d'étudier ce couple de stratégie de propagation et de vaccination est de situer la simulation par rapport au pire des cas théoriques d'une part et de vérifier que nos autres couples de stratégies ne sont pas **pires que le hasard** d'autre part. En effet, si une stratégie particulière ne fait pas mieux que le hasard, on pourra conclure qu'il est inutile de l'utiliser.

8.1.2 Propagation par Voisin

Les différentes stratégies de vaccination autre qu'aléatoire s'appuient fortement sur la topologie du graphe. Pour pouvoir comparer ces stratégies de vaccinations, il est nécessaire de considérer une stratégie de propagation où la structure a son importance, c'est-à-dire par voisin.

Il est alors important de vérifier que les différentes stratégies de vaccination sont **meilleures qu'une stratégie aléatoire** et à quel point elles le sont pour savoir si l'implémentation de ces stratégies de mise à jour est un investissement rentable pour France Télécom.

8.2 Graphe aléatoire uniforme

Un graphe aléatoire uniforme est un graphe où la probabilité d'existence d'une arête entre deux sommets est toujours la même, peu importe les sommets à ses extrémités. Ceci pourrait correspondre par exemple à un réseau uniforme, sans appareils jouant un rôle

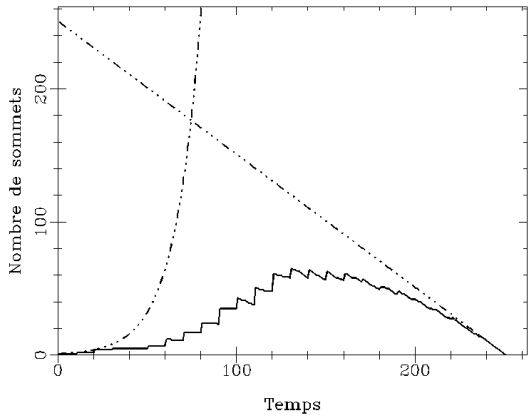
prépondérant dans le réseau.

Les résultats présentés ont été obtenus avec les paramètres suivants :

- Nombre de sommets : 250
- Probabilité d'existence d'une arête : 0.04 (ce qui donne un degrés moyen de 10 dans le graphe)
- Lambda Vaccination (λ) : variation entre 1 et 15
- Tau Propagation (τ) : 10

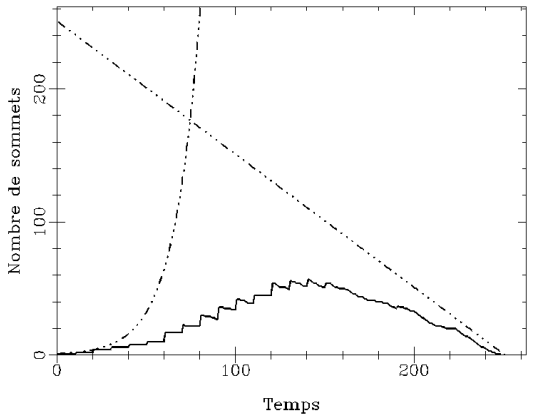
Nous obtenons le jeu de courbe suivant :

Vaccination aléatoire ($\lambda = 1$)



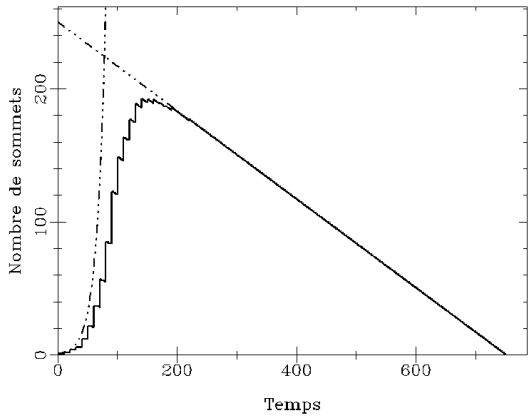
Nombre d'infectés maximum : 74,5
Erreur : 2,5

Vaccination RNB ($\lambda = 1$)



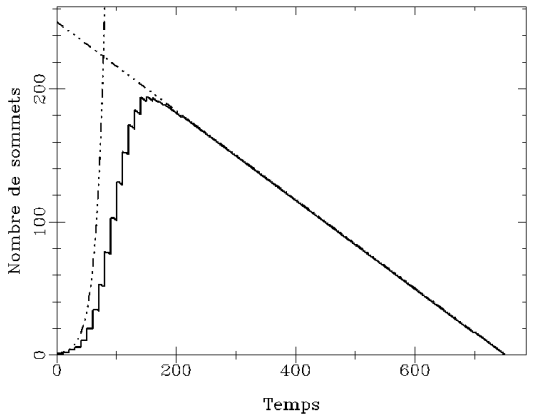
Nombre d'infectés maximum : 57,0
Erreur : 5,6

Vaccination aléatoire ($\lambda = 3$)



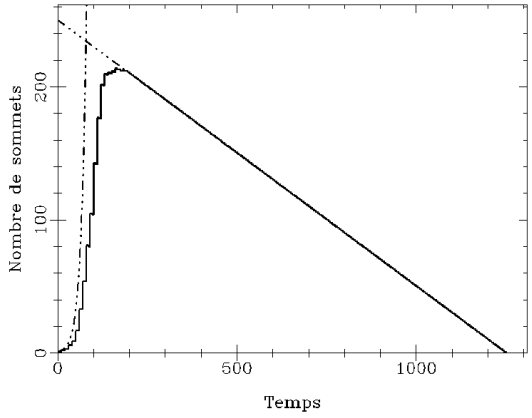
Nombre d'infectés maximum = 191,1
Erreur : 1,4

Vaccination RNB ($\lambda = 3$)



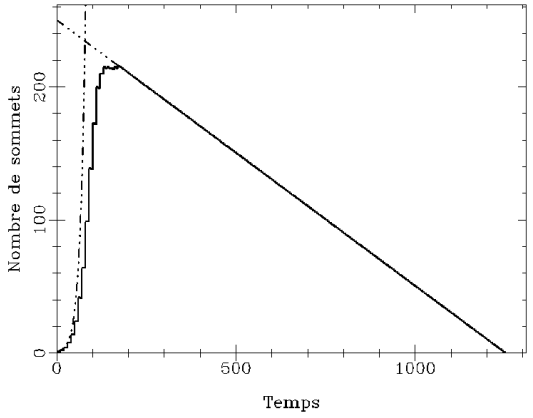
Nombre d'infectés maximum : 190,5
Erreur : 1,7

Vaccination aléatoire ($\lambda = 5$)



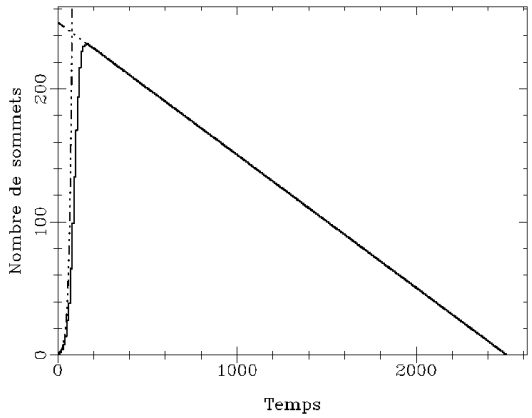
Nombre d'infectés maximum : 214,2
Erreur : 0,4

Vaccination RNB ($\lambda = 5$)



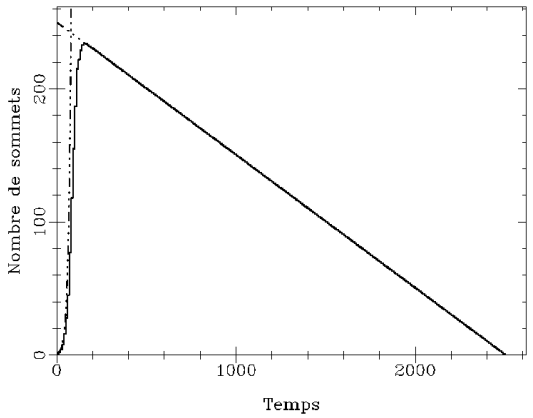
Nombre d'infectés maximum : 199,2
Erreur : 1,4

Vaccination aléatoire ($\lambda = 10$)



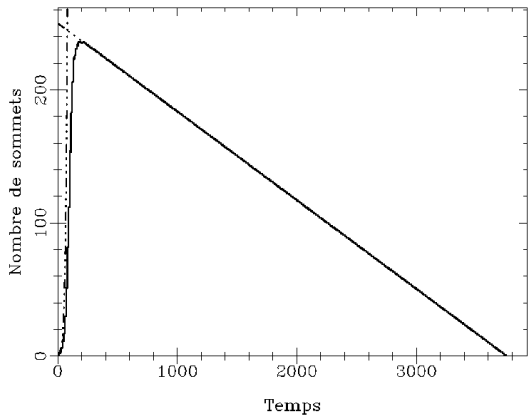
Nombre d'infectés maximum : 232,4
Erreur : 0,3

Vaccination RNB ($\lambda = 10$)



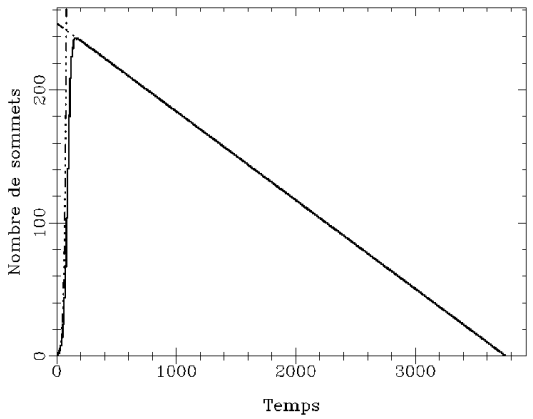
Nombre d'infectés maximum : 231,0
Erreur : 0,4

Vaccination aléatoire ($\lambda = 15$)



Nombre d'infectés maximum = 237,7
Erreur : 0,6

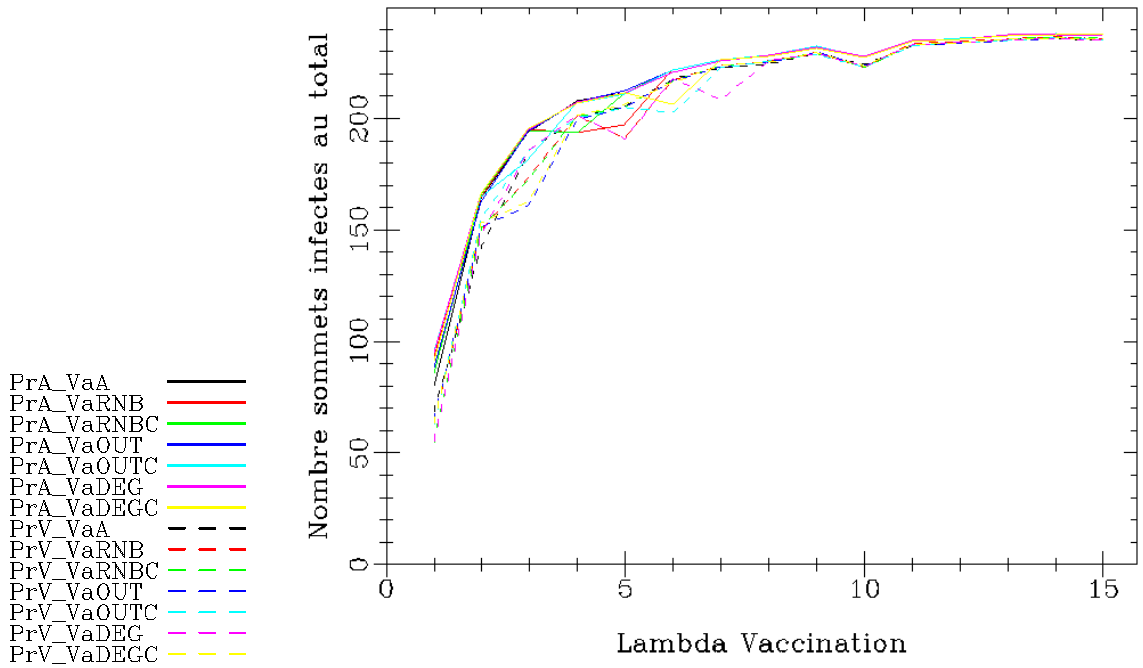
Vaccination RNB ($\lambda = 15$)



Nombre d'infectés maximum : 237,2
Erreur : 0,2

Lorsque la vitesse de propagation est rapide par rapport à la vitesse de vaccination, il est clair que nous ne pouvons pas réellement réagir. Dans ce cas, notre stratégie n'est pas réellement meilleure que le hasard. Par contre, si le ver nous donne l'opportunité de réagir et nous laisse mettre en place une stratégie de propagation, nous constatons que nous pouvons faire mieux que le hasard.

Enfin, pour rendre compte de l'efficacité des différentes méthodes, nous avons tracé les courbes du nombre maximum d'infectés en fonction de lambda.



Ces courbes confirment la première intuition que l'on avait, c'est-à-dire que lorsque la propagation est aléatoire, les stratégies de vaccination sont toutes équivalentes.

Ces courbes confirment aussi la tendance que nous avions repérée juste au-dessus, à savoir que lorsque la vitesse de propagation est trop importante, il n'est pas réellement possible de réagir. Dans ces conditions, les différentes stratégies que l'on peut adopter sont à peu près équivalentes.

On constate aussi que même dans le cas de propagation par voisin, les différentes stratégies ont des résultats qui se ressemblent. Cela s'explique par le fait que le graphe est uniforme. En effet, par construction, les sommets sont tous équivalents en terme de probabilité. Il n'y a donc aucun sommets qui serait un point remarquable dans le graphe. Il n'y a donc pas une grande différence entre une stratégie avancée et une stratégie aléatoire de vaccination dans ce cas.

8.3 Graphe avec classes et probabilités inter-intra classe

Ce type de graphe a été choisi pour modéliser un graphe non uniforme. En faisant varier les différents paramètres, il est possible d'obtenir des graphes différents correspondant aux différents cas qui nous intéressent.

8.3.1 Graphe type concentrateur

Il y a deux classe, une avec peu de sommets mais beaucoup de connexions, donc une classe de sommets à degrés élevés, et une autre avec beaucoup de sommets mais peu de connexions, donc une classe a degrés assez faibles.

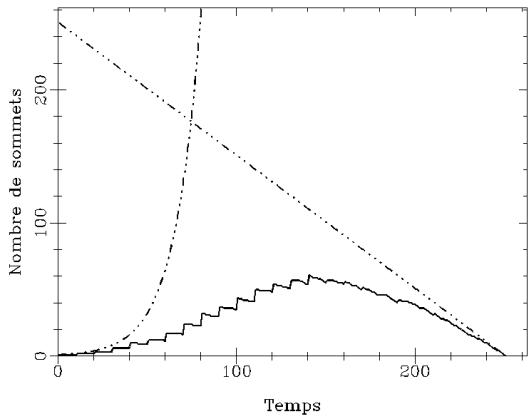
Concrètement, la première classe représente les concentrateurs, la seconde classe représente les clients. Le degrés élevé signifie au niveau de la propagation que les appareils offrants les services ont plus de chances d'être infectés, ce qui paraît logique car ils échangent plus avec le reste du réseau.

Les paramètres utilisés sont les mêmes que pour l'exemple précédent. Le seul paramètre qui n'est plus valable est la probabilité d'existence d'une arête puisque cette probabilité n'est plus uniforme. Ce paramètre est remplacé par :

- Probabilité d'appartenance aux classes : $p_c = \begin{pmatrix} 0.05 \\ 0.95 \end{pmatrix}$
- Matrice des probabilités d'existence des arêtes inter et intra classe : $p_{inter-intra} = \begin{pmatrix} 0.2 & 0.4 \\ 0.4 & 0.02 \end{pmatrix}$

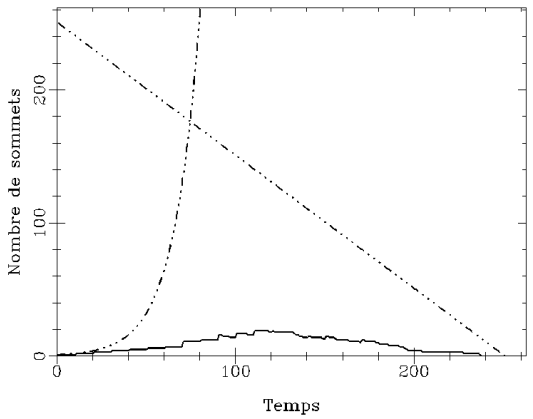
Nous obtenons le jeu de courbes suivant :

Vaccination aléatoire ($\lambda = 1$)



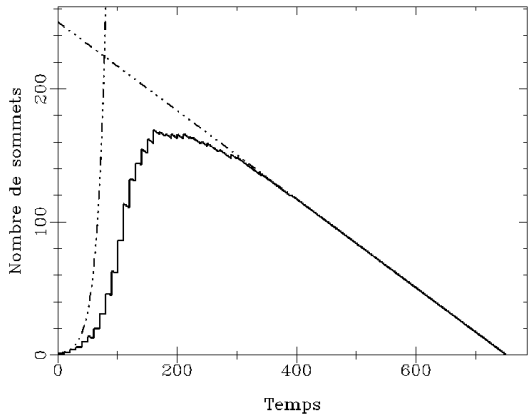
Nombre d'infectés maximum : 51,9
Erreur : 4,3

Vaccination RNB ($\lambda = 1$)



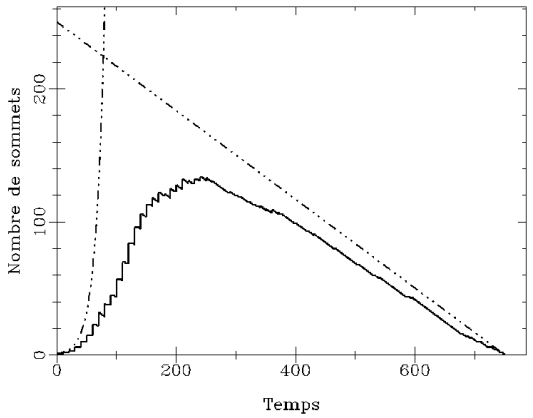
Nombre d'infectés maximum : 12,3
Erreur : 2,1

Vaccination aléatoire ($\lambda = 3$)



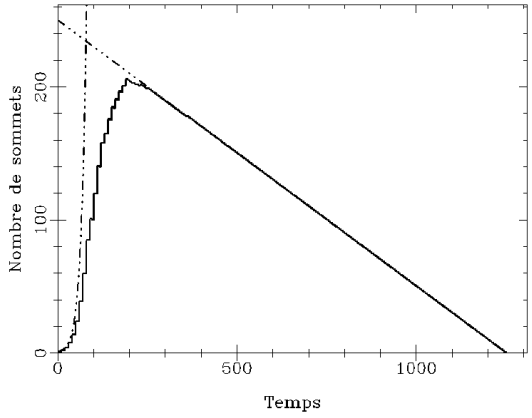
Nombre d'infectés maximum : 169,5
Erreur : 1,0

Vaccination RNB ($\lambda = 3$)



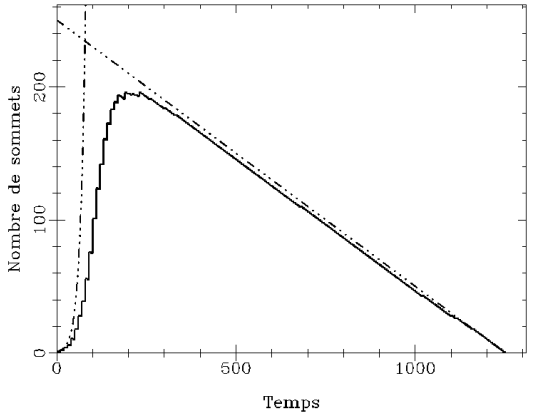
Nombre d'infectés maximum : 137,2
Erreur : 2,9

Vaccination aléatoire ($\lambda = 5$)



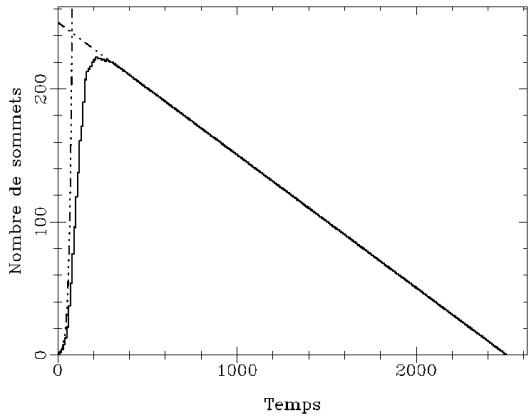
Nombre d'infectés maximum : 199,3
Erreur : 1,0

Vaccination RNB ($\lambda = 5$)



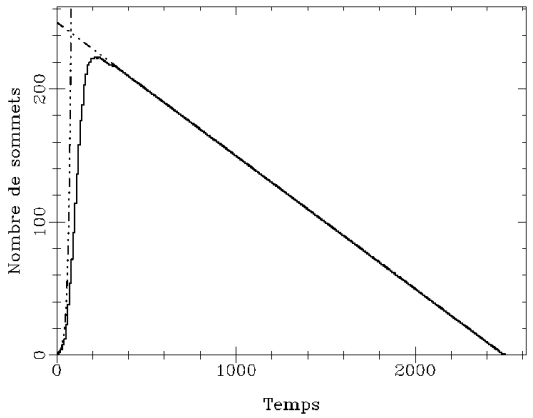
Nombre d'infectés maximum : 184,7
Erreur : 2,3

Vaccination aléatoire ($\lambda = 10$)



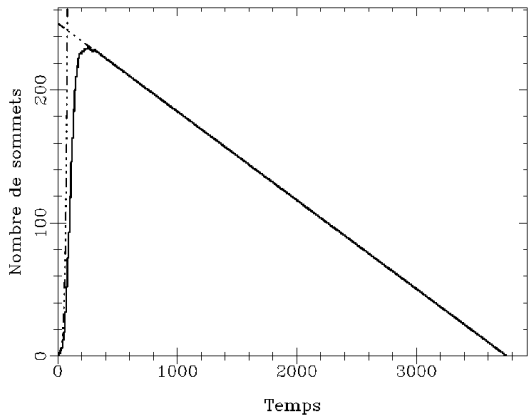
Nombre d'infectés maximum : 222,0
Erreur : 0,7

Vaccination RNB ($\lambda = 10$)



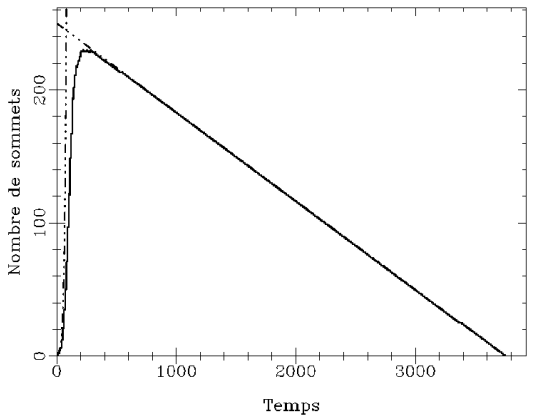
Nombre d'infectés maximum 219,3
Erreur : 0,6

Vaccination aléatoire ($\lambda = 15$)



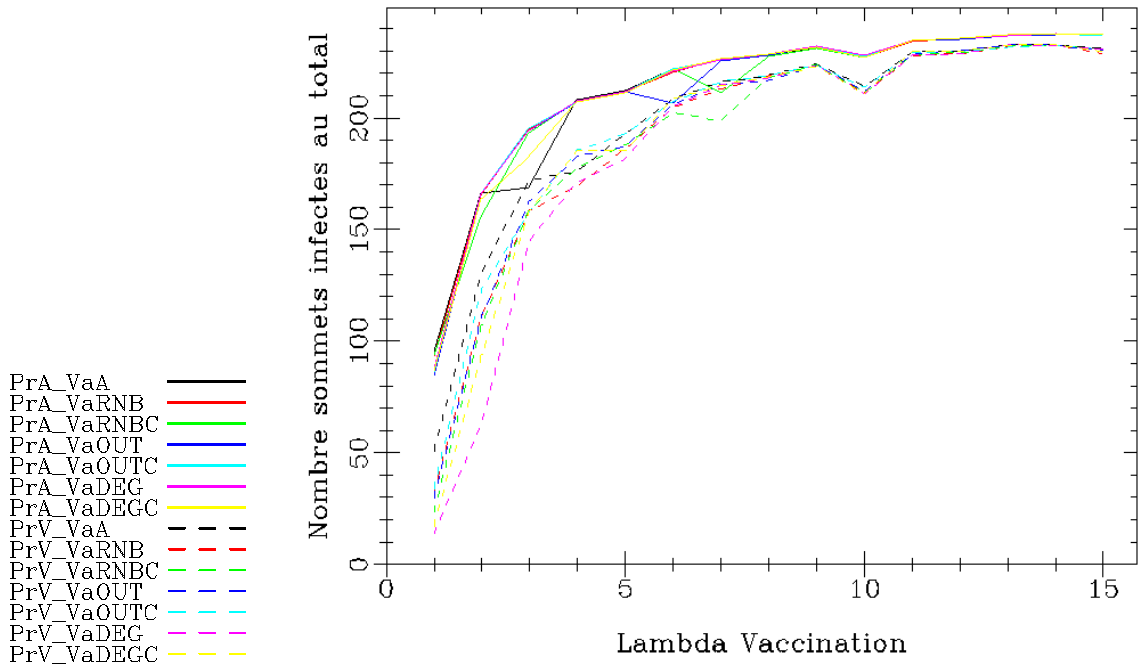
Nombre d'infectés maximum : 231,5
Erreur : 0,5

Vaccination RNB ($\lambda = 15$)



Nombre d'infectés maximum : 230,4
Erreur : 0,4

Enfin, pour rendre compte de l'efficacité des différentes méthodes, nous avons tracé les courbes du nombre maximum d'infectés en fonction de lambda.



Ces courbes confirment la première intuition que l'on avait, c'est-à-dire que lorsque la propagation est aléatoire, les stratégies de vaccination sont toutes équivalentes.

Ces courbes confirment aussi la tendance que nous avions repérée juste au-dessus, à savoir que lorsque la vitesse de propagation est trop importante, il n'est pas réellement possible de réagir. Dans ces conditions, les différentes stratégies que l'on peut adopter sont à peu près équivalentes.

Ce qui tranche avec le cas précédent est l'efficacité des deux méthodes. Elles sont toutes les deux plus efficaces sur ce type de graphe que sur un graphe aléatoire. De plus, l'écart du nombre de sommets infectés est plus prononcé ici. Lorsque la vitesse de vaccination est plutôt lente, on constate une grande différence entre les deux méthodes de vaccination, la méthode RNB s'imposant comme beaucoup plus efficace qu'une vaccination aléatoire.

Cette tendance générale s'observe avec les autres méthodes de vaccination. Elles sont plus efficaces que la vaccination aléatoires. On observe aussi que les stratégies de vaccination sont plus efficaces contre une propagation par voisin que contre une propagation aléatoire. Cette différence est significative contrairement à la différence dans le cas d'un graphe uniforme.

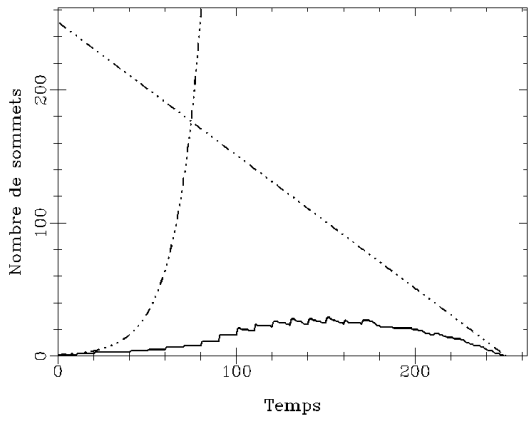
8.3.2 Graphe type concentrateur séparé

Ce type de graphe a presque la même structure que le graphe précédent, à ceci près que les concentrateurs ne sont pas reliés entre eux. Dans ce cas, la matrice des probabilités inter et intra classes n'est plus la même et ressemble à :

- Probabilité d'appartenance aux classes : $p_c = \begin{pmatrix} 0.05 \\ 0.95 \end{pmatrix}$
- Matrice des probabilités d'existence des arêtes inter et intra classe : $p_{inter-intra} = \begin{pmatrix} 0 & 0.4 \\ 0.4 & 0.02 \end{pmatrix}$

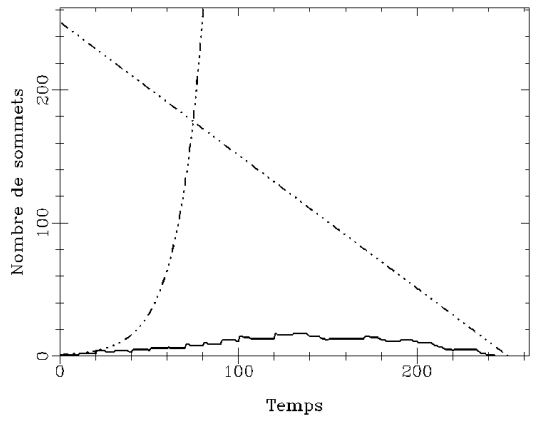
Nous obtenons le jeu de courbes suivant :

Vaccination aléatoire ($\lambda = 1$)



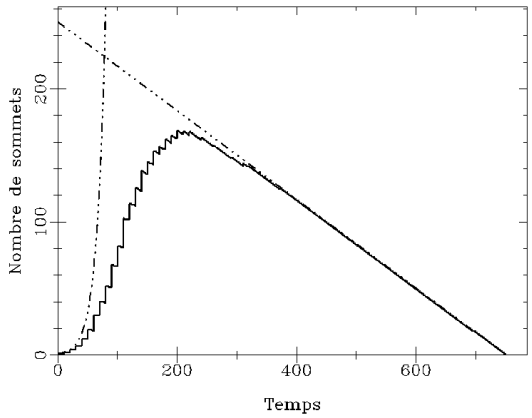
Nombre d'infectés maximum : 50,1
Erreur : 4,7

Vaccination RNB ($\lambda = 1$)



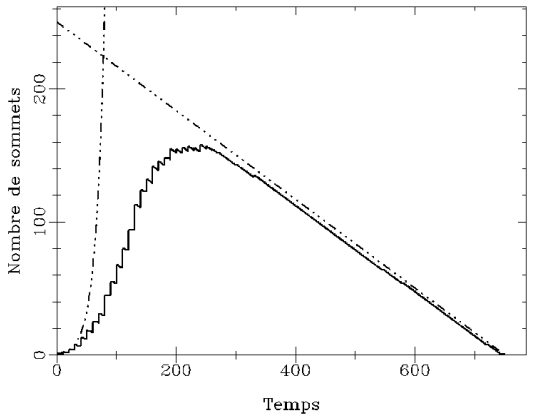
Nombre d'infectés maximum : 11,1
Erreur : 2

Vaccination aléatoire ($\lambda = 3$)



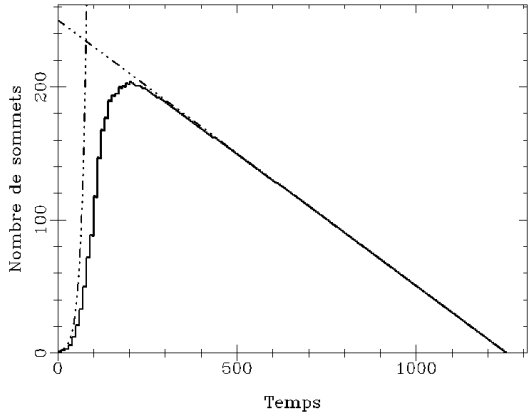
Nombre d'infectés maximum : 158,9
Erreur : 11,2

Vaccination RNB ($\lambda = 3$)



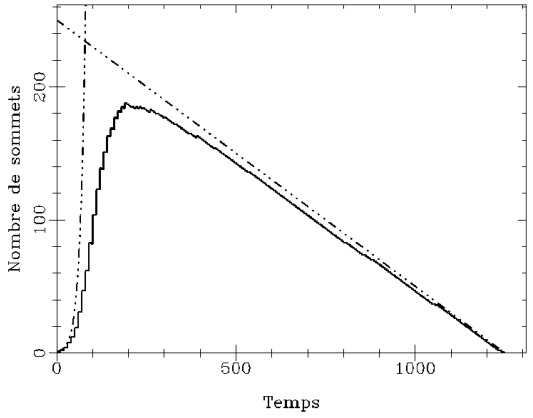
Nombre d'infectés maximum : 137,6
Erreur : 3,4

Vaccination aléatoire ($\lambda = 5$)



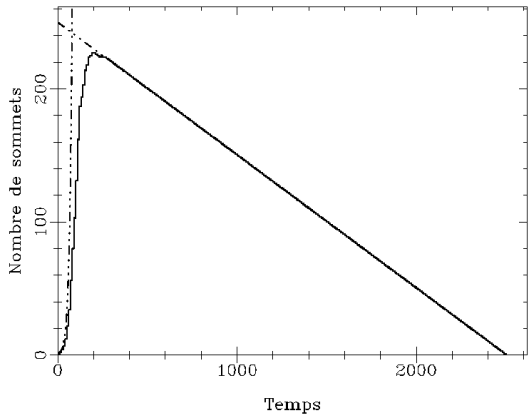
Nombre d'infectés maximum : 199,7
Erreur : 1,2

Vaccination RNB ($\lambda = 5$)



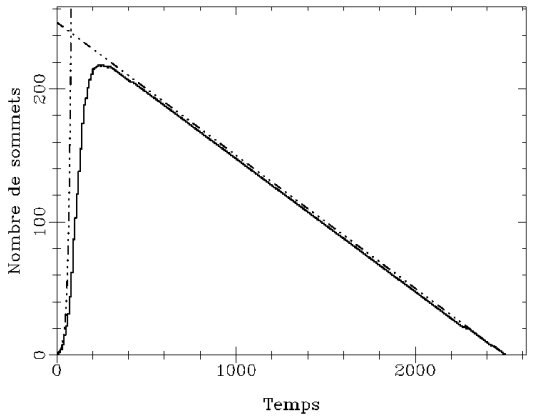
Nombre d'infectés maximum : 186
Erreur : 1,8

Vaccination aléatoire ($\lambda = 10$)



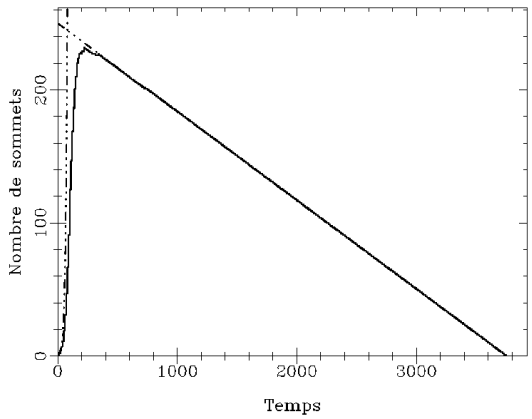
Nombre d'infectés maximum : 222,1
Erreur : 0,6

Vaccination RNB ($\lambda = 10$)



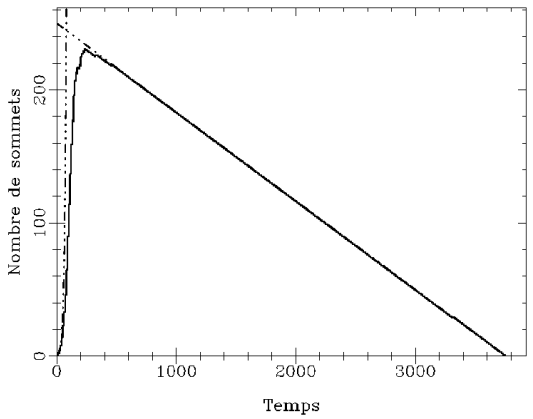
Nombre d'infectés maximum : 218,5
Erreur : 0,6

Vaccination aléatoire ($\lambda = 15$)



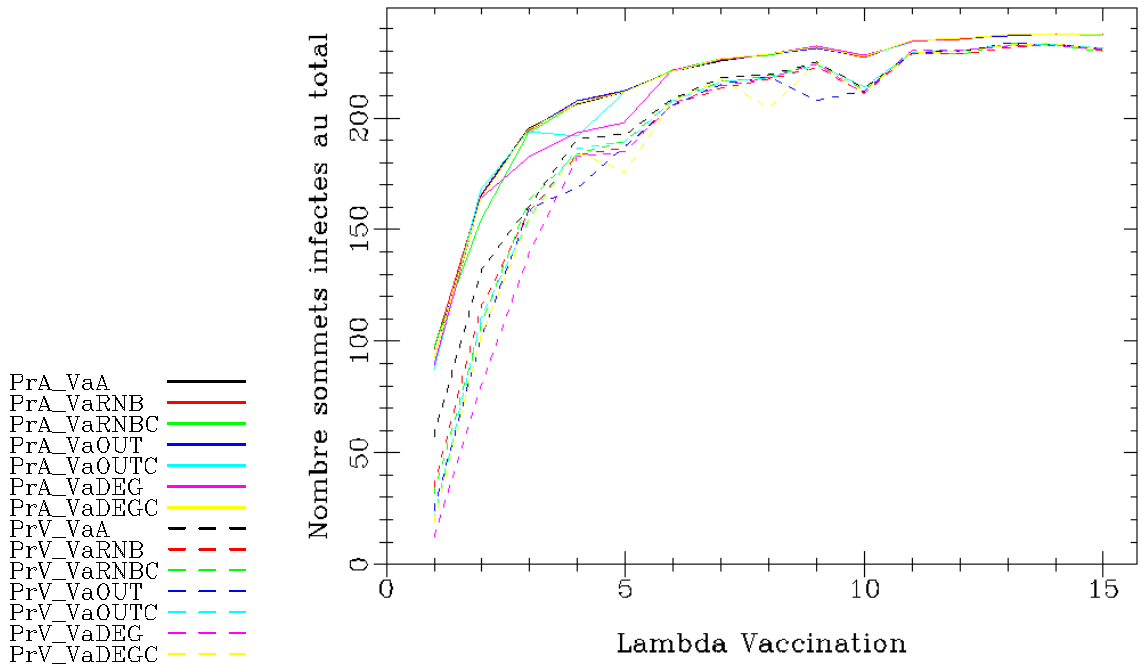
Nombre d'infectés maximum : 230,2
Erreur : 0,5

Vaccination RNB ($\lambda = 15$)



Nombre d'infectés maximum : 228,4
Erreur : 0,7

Enfin, pour rendre compte de l'efficacité des différentes méthodes, nous avons tracé les courbes du nombre maximum d'infectés en fonction de lambda.



Ces courbes confirment la première intuition que l'on avait, c'est-à-dire que lorsque la propagation est aléatoire, les stratégies de vaccination sont toutes équivalentes.

Ces courbes confirment aussi la tendance que nous avions repérée juste au-dessus, à savoir que lorsque la vitesse de propagation est trop importante, il n'est pas réellement possible de réagir. Dans ces conditions, les différentes stratégies que l'on peut adopter sont à peu près équivalentes.

En reprenant les résultats précédents, la méthode RNB n'est pas significativement meilleure que lorsque les concentrateurs n'étaient pas séparés. Par contre la vaccination aléatoire est ici plus efficace. L'écart du nombre de sommets infectés entre les deux méthodes reste important pour des petites valeurs de lambda, mais pas autant que précédemment.

Cette tendance est confirmée pour les autres stratégies de vaccination. Il n'y a que peu de différences avec le cas précédent.

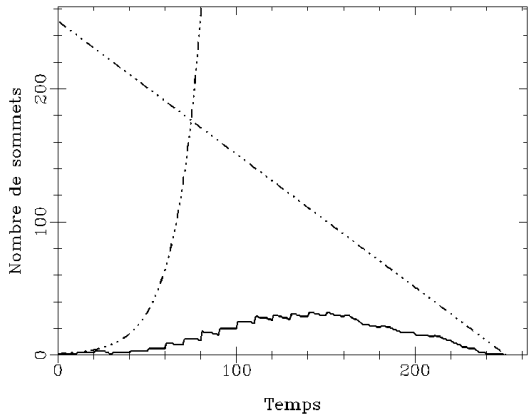
8.3.3 Graphe type concentrateur séparation client

Ce type de graphe ressemble au graphe de type concentrateur, à ceci près que les clients n'ont pas de lien entre eux. Dans ce cas, la matrice des probabilités inter et intra classes n'est plus la même et ressemble à :

- Probabilité d'appartenance aux classes : $p_c = \begin{pmatrix} 0.05 \\ 0.95 \end{pmatrix}$
- Matrice des probabilités d'existence des arêtes inter et intra classe : $p_{inter-intra} = \begin{pmatrix} 0.2 & 0.4 \\ 0.4 & 0 \end{pmatrix}$

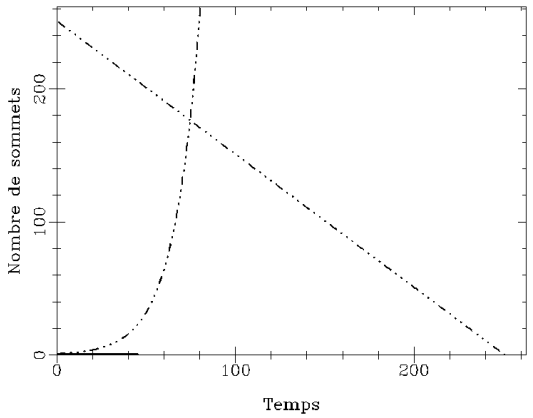
Nous obtenons le jeu de courbes suivant :

Vaccination aléatoire ($\lambda = 1$)



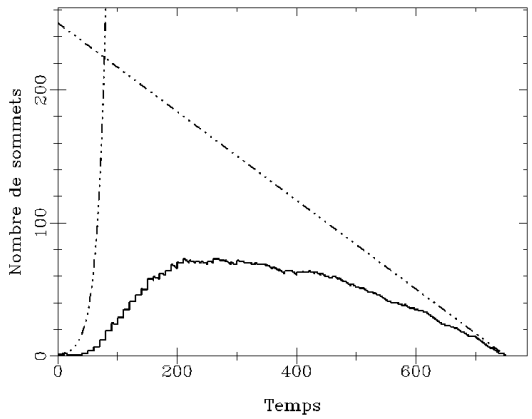
Nombre d'infectés maximum : 30,4
Erreur : 1,6

Vaccination RNB ($\lambda = 1$)



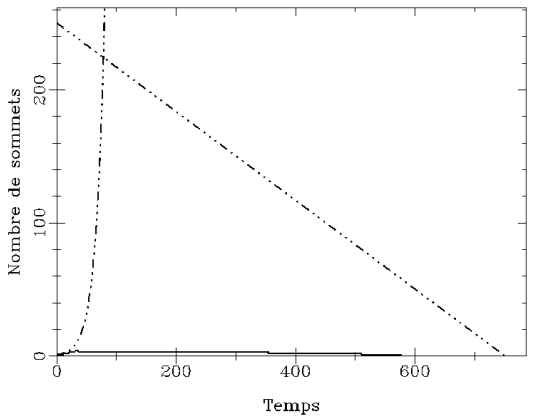
Nombre d'infectés maximum : 1,3
Erreur : 0,2

Vaccination aléatoire ($\lambda = 3$)



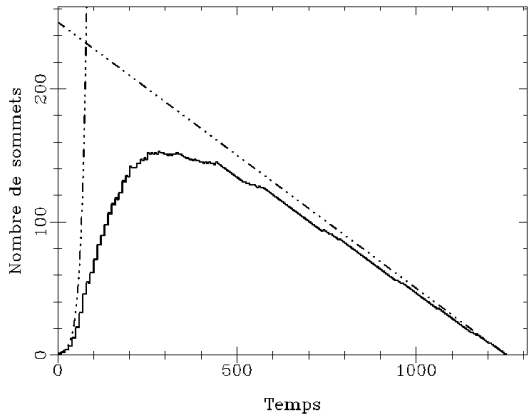
Nombre d'infectés maximum : 96,3
Erreur : 4,4

Vaccination RNB ($\lambda = 3$)



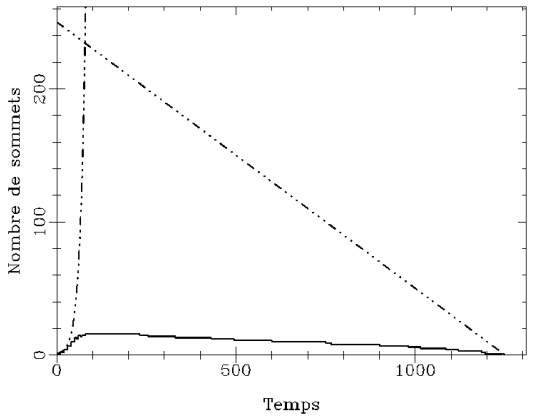
Nombre d'infectés maximum : 2,9
Erreur : 0,6

Vaccination aléatoire ($\lambda = 5$)



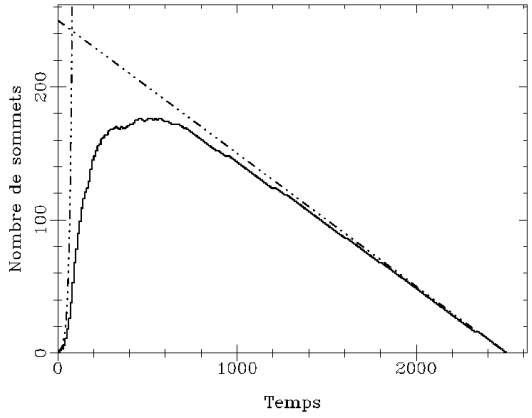
Nombre d'infectés maximum : 122,5
Erreur : 3,5

Vaccination RNB ($\lambda = 5$)



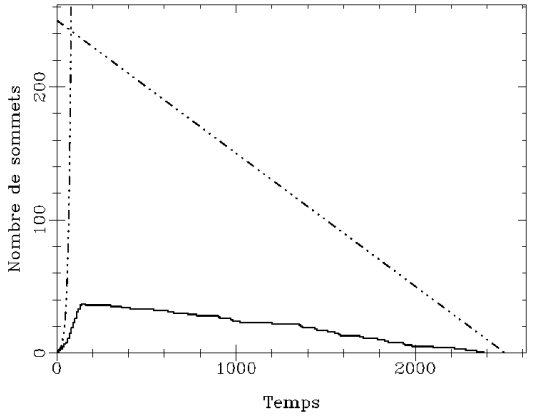
Nombre d'infectés maximum : 7,7
Erreur : 2,1

Vaccination aléatoire ($\lambda = 10$)



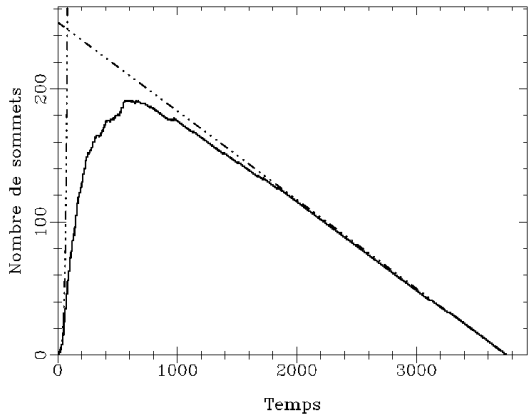
Nombre d'infectés maximum : 165,9
Erreur : 4,3

Vaccination RNB ($\lambda = 10$)



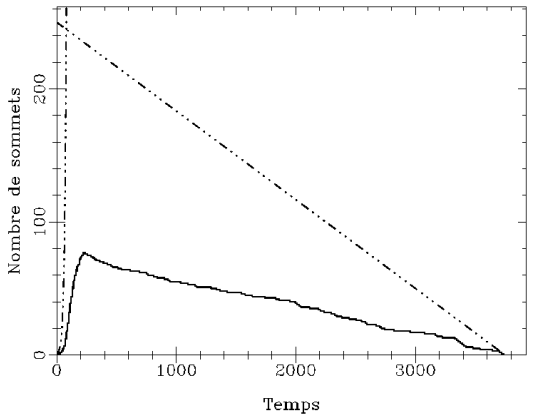
Nombre d'infectés maximum : 35,6
Erreur : 5,4

Vaccination aléatoire ($\lambda = 15$)



Nombre d'infectés maximum : 184,3
Erreur : 4,9

Vaccination RNB ($\lambda = 15$)

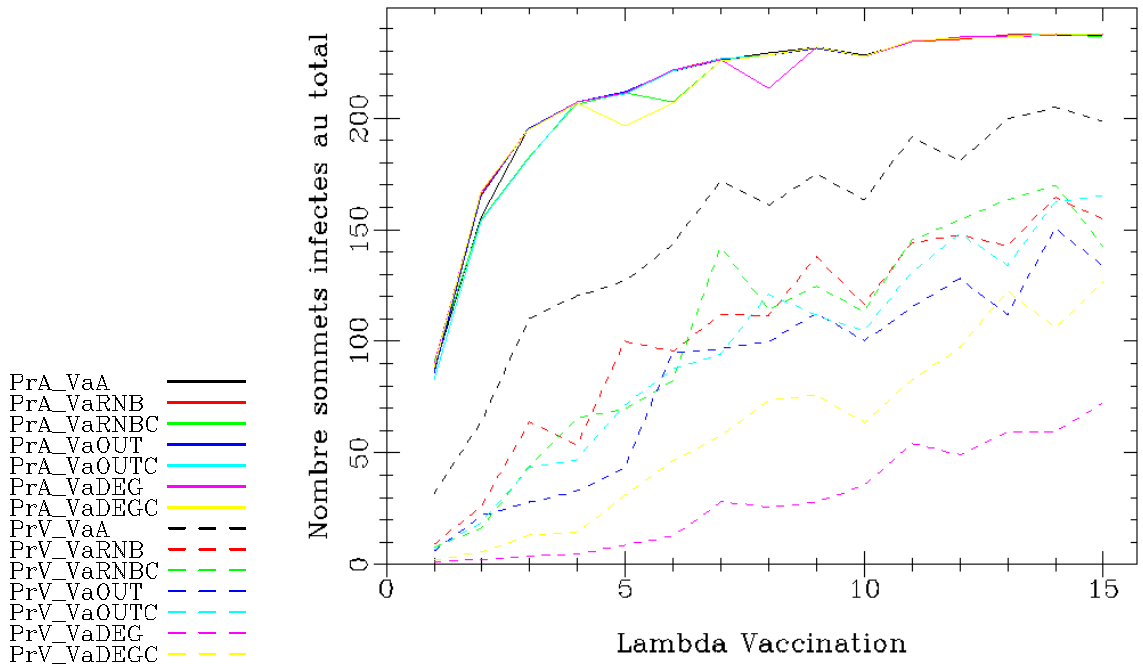


Nombre d'infectés maximum : 68,8
Erreur : 6,8

A travers ces premières courbes, nous pouvons constater que la stratégie mise en place est particulièrement efficace pour enrayer la propagation. Il est aussi possible que, vu les paramètres choisis, la plupart des graphes générés ne soient pas connexes, ce qui expliquerait une difficulté pour le ver à se propager. Nous pouvons quand même constater que l'allure des courbes est assez différente des courbes obtenues précédemment, et que donc, dans ce type de graphe, la stratégie de propagation est payante même à des valeurs de lambda élevées.

Pour étayer cette hypothèse, nous pouvons regarder les dernières courbes de la série. Dans le cas RNB comme dans les autres stratégies de vaccination, nous essayons d'enlever les sommets clés. Ici, il se peut que comme le graphe de départ n'est pas connexe, le nombre de sommets clés soit assez faible. La stratégie testée donne de bons résultats car les sommets clés sont vite atteints et diminue encore le champ de propagation possible pour le vers. Pour s'assurer de ce que l'on avance, il faudrait dans des travaux ultérieurs calculer la connexité du graphe de départ.

Enfin, pour rendre compte de l'efficacité des différentes méthodes, nous avons tracé les courbes du nombre maximum d'infectés en fonction de lambda.



Ces courbes confirment la première intuition que l'on avait, c'est-à-dire que lorsque la propagation est aléatoire, les stratégies de vaccination sont toutes équivalentes.

Par contre, ces courbes ont une allure assez différentes des cas étudiés précédemment. En effet, la stratégie RNB s'avère très efficace, même pour des valeurs de λ relativement élevées. En effet, même pour λ valant 15, la stratégie de vaccination est bien plus efficace qu'une vaccination aléatoire.

En comparant ce résultat avec les autres stratégies de vaccination, on peut voir des écarts significatifs entre les différentes stratégies de vaccination. En effet, les courbes dans le cas d'une propagation par voisin sont très écartées les unes des autres. Cela signifie que dans ce type de graphe, nous pouvons agir avec beaucoup de liberté sur λ pour combattre l'infection et qu'il peut être intéressant d'investir du temps et de l'énergie pour mettre en place la meilleure stratégie de vaccination.

8.4 Graphe supplémentaires à étudier

Durant l'étude, il est apparu d'autres structures de graphes intéressantes, représentant par exemple un certain type de réseau ou une certaine stratégie de propagation, sur lesquels nous n'avons pas eu le temps d'effectuer de simulations.

8.4.1 Graphe de type propagation locale majoritaire

Cette propagation part du postulat que si le ver a réussi à infecter une machine, les autres machines proches en terme d'IP doivent être plus facilement infectables (car peut-être similaires). En effet, des machines avec des IP proches appartiennent souvent au même sous-réseau et sont donc administrées par la même personne. Dans ce cas, nous pouvons comprendre la logique de l'attaque, car si c'est la même personne qui gère le réseau, il n'y a pas de raison qu'il ait effectué des mise à jour de manière non uniforme sur son parc. Par exemple, si sur un réseau local, il est possible d'exploiter une faille particulière, on suppose que les autres machines du réseau ne sont pas non plus protégées vis à vis de cette faille.

Ceci pourrait se modéliser par des classes (autant que de petits réseaux locaux) où la probabilité d'existence d'arête intra-classe est élevée et la probabilité d'existence d'arête inter-classe est faible. Un exemple à 4 réseaux (par exemple les 4 classes d'adresse IP dont se servait le ver CodeRed) pourrait se modéliser avec les informations suivantes :

- Probabilité d'appartenance aux classes : $p_c = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}$
- Matrice des probabilités d'existence des arêtes inter et intra classe :
  $p_{inter-intra} = \begin{pmatrix} \frac{4}{10} & \frac{3}{10} & \frac{2}{10} & \frac{1}{10} \\ \frac{3}{10} & \frac{4}{10} & \frac{1}{10} & \frac{2}{10} \\ \frac{2}{10} & \frac{1}{10} & \frac{4}{10} & \frac{3}{10} \\ \frac{1}{10} & \frac{2}{10} & \frac{3}{10} & \frac{4}{10} \end{pmatrix}$

8.4.2 Graphe de type ring

Ce type de graphe représente une architecture réelle utilisée chez France Télécom. Il est constitué dans un premier temps de serveur relié entre eux de manière à former un cercle. Dans un second temps ces serveurs sont reliés à de nombreux clients qui peuvent être relié à plusieurs serveurs ou reliés entre eux. En reprenant ce qui a été fait avant, ceci revient à un graphe de type concentrateur mais avec les serveurs reliés de façon à former un cercle.

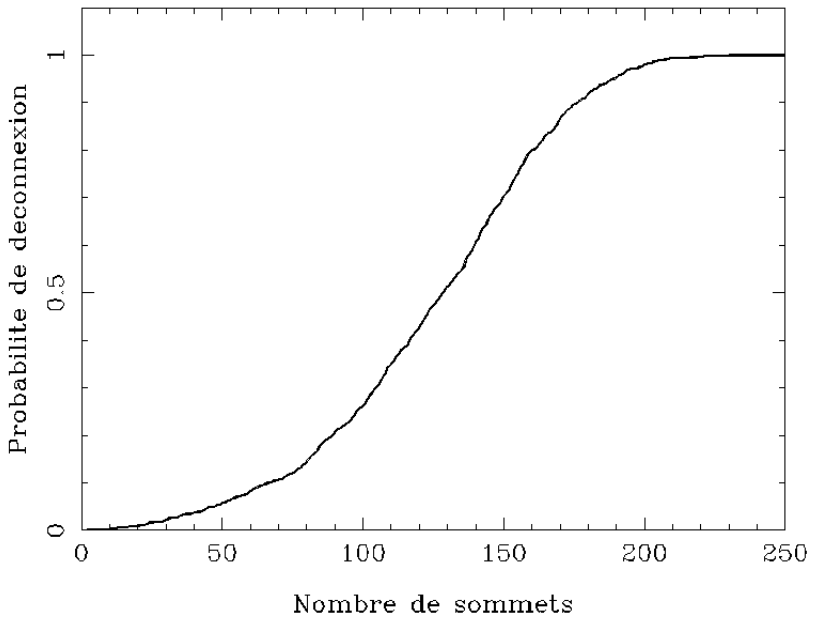
9 Analyse à l'aide de nos indicateurs

Nous avons maintenant tout en main pour comparer les différentes méthodes et essayer d'évaluer leur efficacité. Je rappelle que cette étude n'a d'intérêt que s'il est possible de réellement améliorer la performance d'une vaccination aléatoire.

9.1 Taux de percolation

Nous allons comparer le taux de percolation au nombre de sommets vaccinés à l'instant où le maximum de sommets infectés est atteint. Nous espérons qu'un des mécanisme mis en jeu pour contrer une propagation est le mécanisme de déconnexion dans un graphe. En effet, lors d'une propagation de voisin en voisin, si nous arrivons à déconnecter le graphe, nous pourrons peut-être contenir l'infection dans une portion du graphe et protéger l'autre partie.

Nous avons fait une simulation pour 250 sommets afin d'obtenir une courbe de percolation que nous pourrons comparer aux résultats obtenus jusqu'ici :

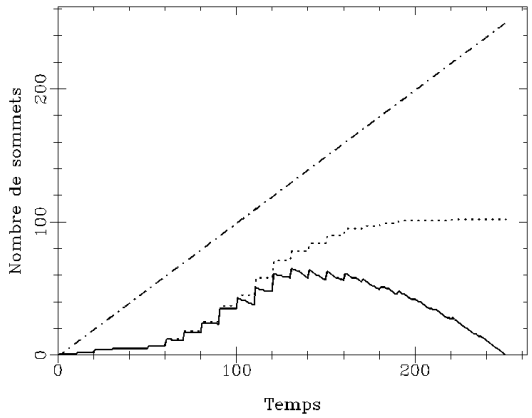


La courbe de percolation varie peu selon la structure du graphe. Sur cette courbe nous retrouvons bien l'idée du seuil de percolation. En effet, jusqu'à une valeur seuil, la probabilité de déconnecter un type de graphe en enlevant des sommets aléatoirement est faible. Au-delà de cette valeur seuil, la probabilité de déconnecter ce même type de graphe augmente très rapidement.

Nous rappelons que ce que nous appelons seuil de percolation est le nombre moyen de sommet à enlever pour déconnecter un type de graphe donné. Pour le situer sur la courbe, le seuil de percolation équivaut au nombre de sommets à enlever pour avoir 50% de chances de déconnecter le graphe (toujours à un niveau de confiance de 95%).

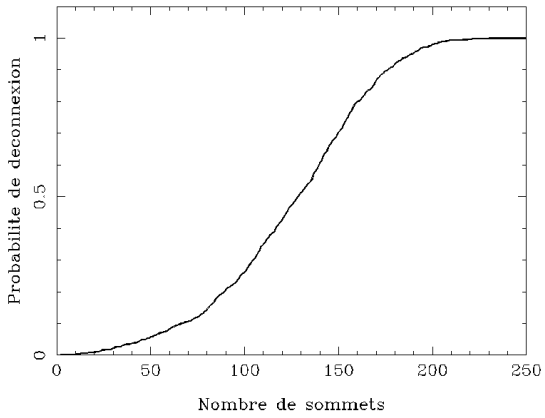
Concrètement, l'idée est de situer le nombre de machines mises à jour lorsque l'infection diminue sur la courbe de percolation. Nous avons choisi de confronter les courbes précédentes dans le cas d'un graphe uniforme et d'une vaccination aléatoire. La propagation est là aussi par voisin pour que la percolation ait un réel sens. En effet, si la propagation se fait de manière totalement aléatoire, il ne sert à rien d'essayer de déconnecter le graphe.

Courbe d'infection (lambda valant 1)



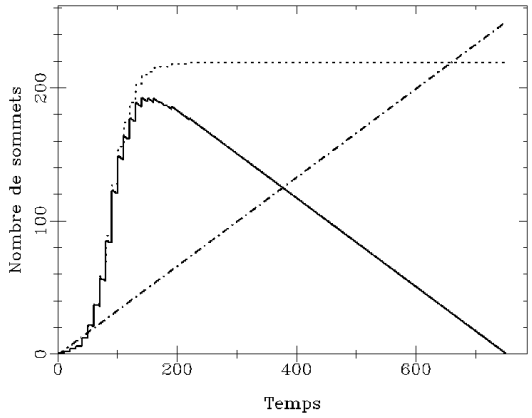
Nombre de vaccinés au maximum d'infection : 92,5

Courbe de percolation



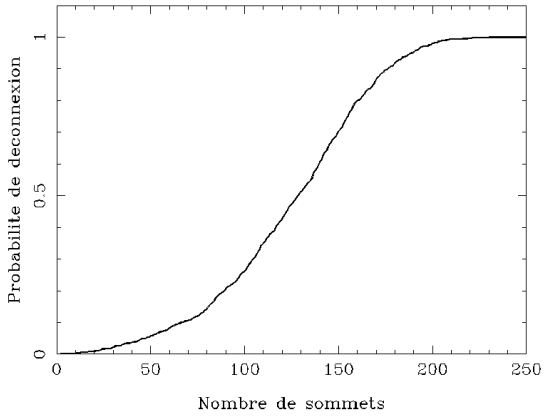
Probabilité de déconnexion à 92,5 : 0,22

Courbe d'infection (lambda valant 3)



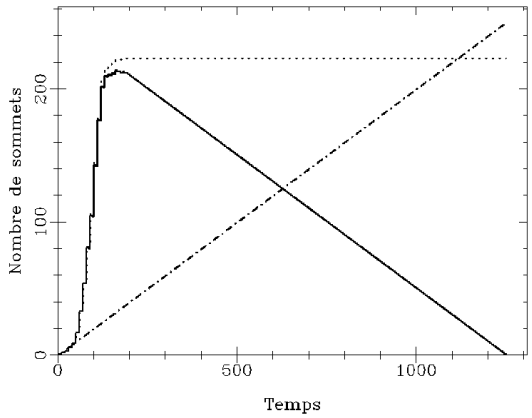
Nombre de vaccinés au maximum d'infection : 183,3

Courbe de percolation



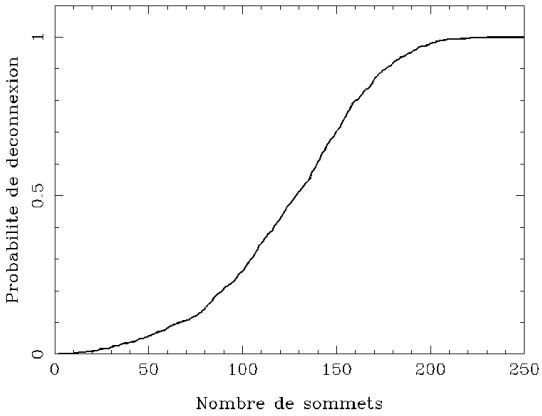
Probabilité de déconnexion à 92,5 : 0,93

Courbe d'infection (lambda valant 5)



Nombre de vaccinés au maximum d'infection : 220,6

Courbe de percolation



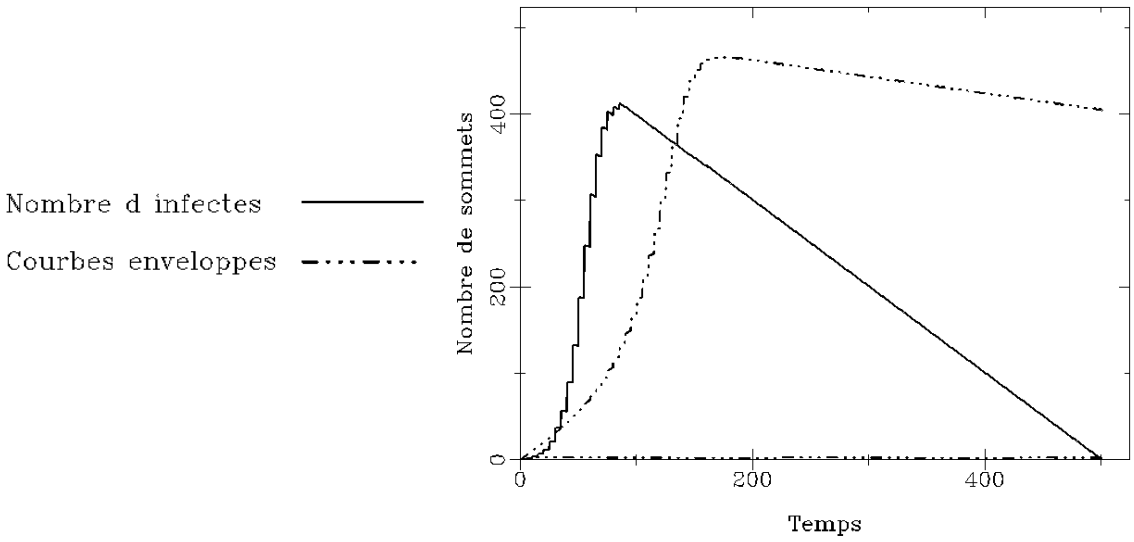
Probabilité de déconnexion à 220,6 : 0.99

J'ai choisi de présenter ces trois valeurs de lambda pour bien montrer que la percolation, c'est-à-dire la déconnexion dans le graphe, ne semble pas être le phénomène mis en jeu ici.

En effet, nous atteignons très vite des valeurs élevées sur la courbe de percolation, bien au-delà du seuil de percolation. Cela nous amène à penser que la déconnexion du graphe n'a pas un rôle prépondérant dans notre combat contre l'infection. Il y a cependant la connexité locale que nous n'avons pas du tout étudié. Une étude ultérieure pourrait s'intéresser plus particulièrement à ce phénomène.

9.2 Courbes enveloppes

Une simple superposition des courbes enveloppes obtenues montre que ces dernières n'enveloppent pas réellement nos simulations comme nous le pensions.



La courbe du dessous et bien trop en dessous pour pouvoir en tirer quelque chose. La courbe qu'on espère toujours au dessus ne l'est pas. Ces résultats non concluant s'expliquent par l'enchaînement des approximations (à chaque étape il y a approximation) dans un premier temps, puis par le fait que ce sont des courbes de probabilités dans un second temps. Les courbes enveloppes sont inexploitables pour la suite de notre étude.

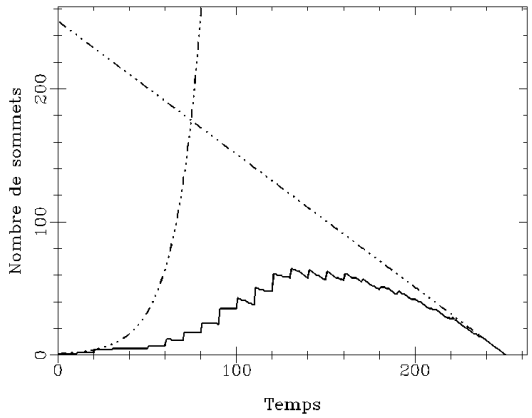
9.3 Courbe maximale

Dans un premier temps, nous souhaitons savoir si nous sommes éloignés du pire des cas dans l'ensemble. En effet, si même notre meilleure stratégie est proche du pire des cas, il ne servira à rien de la mettre en place sur le réseau.

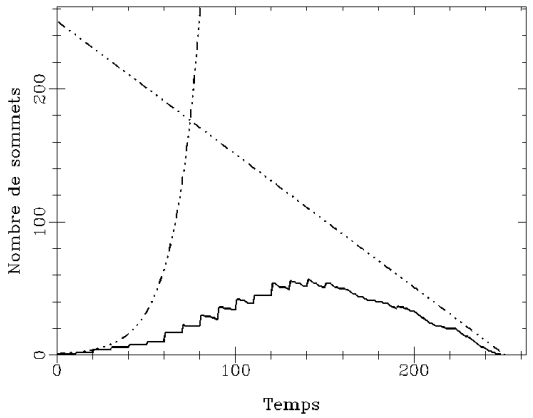
Dans un second temps, nous souhaitons savoir si la méthode utilisée par France Télécom donne un résultat proche de ce cas et à quel point nous pouvons nous en éloigner en changeant de stratégies. En d'autres termes, nous voulons savoir si l'efficacité relative de nos stratégies varie en fonction du rapport temps de propagation/temps de vaccination.

Nous avons choisi, de ne montrer que les courbes obtenus sur des graphes uniformes avec les mêmes paramètres que précédemment. En effet, ces courbes représentent bien la tendance général, et comme nous n'avons pas d'information réelle sur le réseau, le choix du graphe uniforme est pertinent. Pour les mêmes raisons que l'étude statistique, nous avons choisi de présenter uniquement la stratégie de vaccination aléatoire et la stratégie de vaccination RNB.

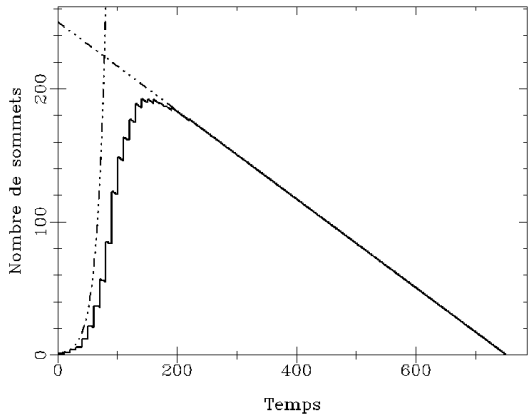
Vaccination aléatoire ($\lambda = 1$)



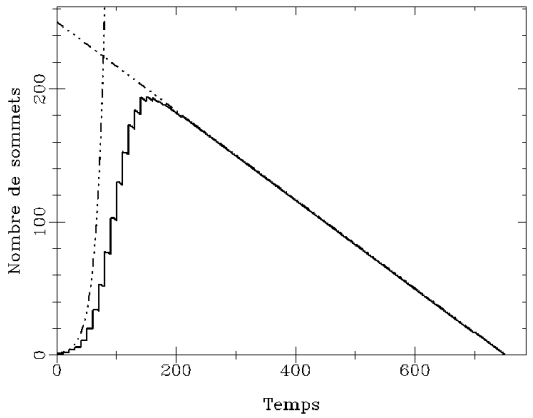
Vaccination RNB ($\lambda = 1$)



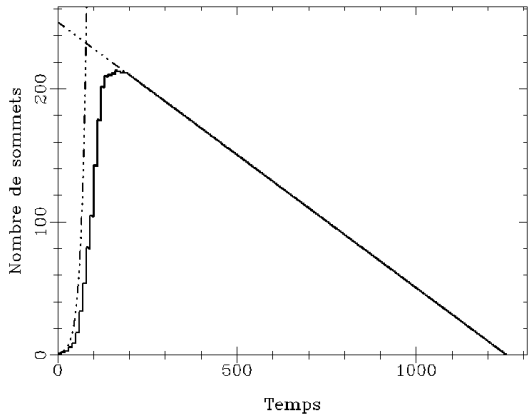
Vaccination aléatoire ($\lambda = 3$)



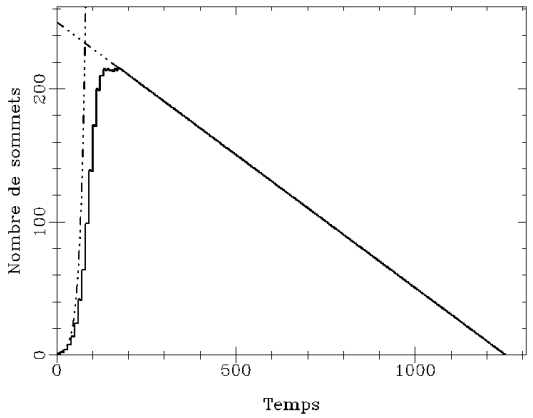
Vaccination RNB ($\lambda = 3$)



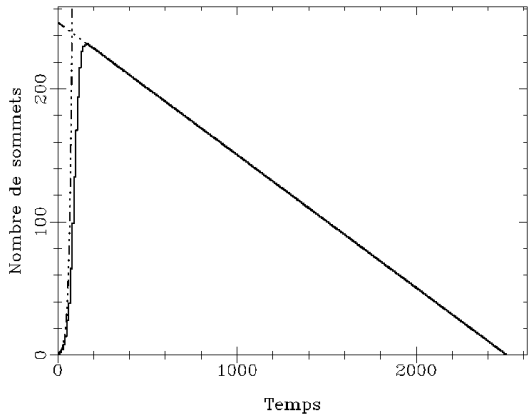
Vaccination aléatoire ($\lambda = 5$)



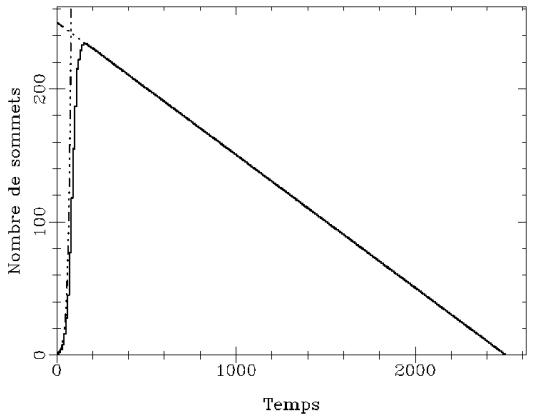
Vaccination RNB ($\lambda = 5$)



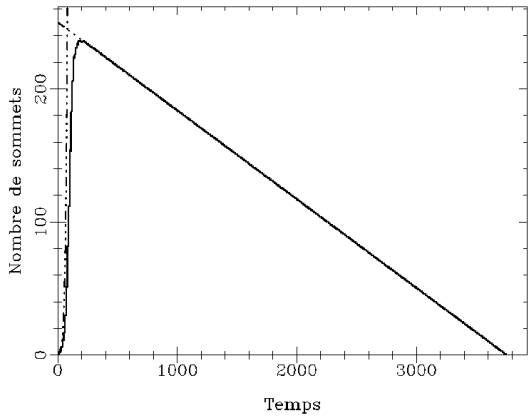
Vaccination aléatoire ($\lambda = 10$)



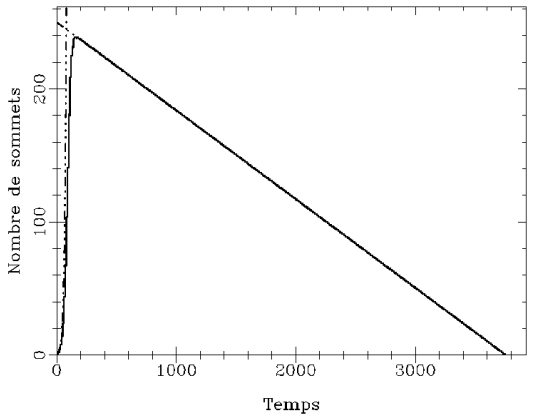
Vaccination RNB ($\lambda = 10$)



Vaccination aléatoire ($\lambda = 15$)



Vaccination RNB ($\lambda = 15$)



Nous constatons que lorsque la vitesse de propagation est rapide, quoi qu'il arrive nous sommes proches du pire des cas. Par contre, lorsque nous avons la possibilité de réagir, nous nous éloignons de cette courbe de plus en plus. Il apparaît un intérêt certain dans les différentes stratégies de vaccination, car plus l'administrateur vaccine vite son parc de machine, plus la différence entre une stratégie aléatoire et une stratégie donnée est grande. En d'autres termes, plus nous réagissons vite en mettant une stratégie précise en place, plus nous protégeons de machines. L'efficacité relative augmente avec l'augmentation de la vitesse de vaccination.

9.4 Indicateurs supplémentaires à étudier

Il est possible d'étudier des indicateurs supplémentaires qui nous semblent pertinents à la fin de notre étude. En effet, nous avons vu que la connexité du graphe joue un rôle important dans la propagation (comme par exemple pour la propagation dans un graphe de type concentrateur avec séparation des clients). D'un autre côté, d'après les résultats obtenus par la percolation, ce n'est pas juste le phénomène de déconnexion du graphe qui est mis en jeu.

Il paraît donc intéressant de se tourner vers des indicateurs tels que la connexité dans le graphe de départ dans un premier temps déjà, mais aussi vers par exemple la connexité locale, le degré moyen ou le diamètre.

10 Conclusion

Mon travail durant ce stage a donc été découvrir si des stratégies de vaccinations sont intéressantes à mettre en place pour la société Orange. J'ai dû effectuer un travail de recherche dans un premier temps pour découvrir les stratégies existantes, puis j'ai dû mener

à bien la partie développement pour enfin effectuer des simulations.

Mon travail a une importance cruciale puisque selon mes résultats, certains employés du services vont continuer les recherches dans certaines directions, ou vont simplement décider d'abandonner le projet pour le moment. Mon travail est donc un préliminaire pour diriger dans la bonne direction des études futures.

Il faut aussi rappeler que le travail effectué ici, c'est-à-dire déterminer la meilleure stratégie de vaccination possible, n'est qu'un complément de ce qui existe déjà pour se protéger des malwares. En effet, il existe aussi des services de sécurité dédiés à la surveillance des réseaux, avec par exemple des sondes IDS/IPS, des firewall et des anti-virus. Tout ceci forme déjà une solide défense contre les attaques extérieures.

A travers cette étude, nous avons apporté des premiers éléments de réponses à la problématique posée. En effet, nous avons testé différentes stratégies de vaccination dans différents cas et avons constaté que nous pouvions faire mieux que le hasard dans la majorité des cas.

Pour évaluer au mieux l'efficacité relative des différentes méthodes nous avons aussi cherché à mettre en place des indicateurs. Ces indicateurs nous ont permis entre autre de nous situer par rapport au pire des cas. Nous situer par rapport à ce cas extrême nous permet entre autre de savoir s'il est intéressant de mettre en place une réelle stratégie de vaccination ou si nous ne pouvons qu'être spectateur de l'infection et réagir à posteriori.

A ce titre, la vitesse relative de vaccination par rapport à la vitesse de propagation est absolument cruciale. En effet, si nous ne pouvons réagir plus vite que le ver, il infectera très rapidement tout notre parc de machine avant que nous n'ayons protégé les éléments clés du réseau.

Ce travail n'est cependant pas suffisant, car il existe encore de nombreux cas à traiter, que ce soit en terme de stratégie de propagation et de stratégie de vaccination, ou en terme de scénario.

En effet, dans nos exemples nous avons fait débuter la propagation du ver et la vaccination au même moment. Dans la réalité, c'est un cas plutôt rare. En général le ver commence à infecter le réseau, se fait repérer par des experts en sécurité, une équipe de développeur crée un patch contre ce vers, et enfin, le réseau commence à être traité. La vaccination commence donc bien plus tard que la propagation, ce qui n'est pas le cas ici.

Un autre scénario que nous n'avons pas testé est une vaccination avant le début d'infection. Ce cas correspond à la découverte d'une faille non exploitée et sa correction avant que des vers exploitant cette faille ne soit lancés sur le réseau. Lorsqu'une nouvelle faille est découverte, elle n'est pas nécessairement exploitée par les pirates tout de suite. Dans

ce scénario, l'équipe de développement corrige donc cette faille et commence à mettre à jour les machines du réseau. C'est durant cette phase de mise à jour qu'un pirate lance un ver sur le réseau. Dans ce cas, la vaccination aura commencé avant l'infection.

Enfin, nous pouvons selon la structure du graphe dégager une ou deux méthodes qui se démarquent. Le tableau suivant est une facette du bilan de mon stage. Il montre la stratégie de vaccination qui fonctionne le mieux contre une stratégie de propagation par voisin, d'après les résultats obtenus avec nos simulations statistiques.

Type de graphe	Meilleure stratégie de vaccination
Uniforme	OUTC
Concentrateur simple	DEG
Concentrateurs séparés	DEG
Concentrateurs avec clients séparés	DEG

Ce tableau met en évidence que la stratégie de vaccination DEG est la plupart du temps la meilleure. Il peut donc être intéressant de pousser les recherches ultérieures vers cette stratégie en particulier.

Références

[1] Weibo Gong Cliff Changchun Zou and Don Towsley. Code red worm propagation modeling and analysis. *CCS*, Novembre 2002. Une étude détaillé de CodeRed permière et seconde version.

[2] Jesus Gomez Garden Pablo Echenique and Yamir Moreno. Immunization of real complex communication networks. *European Physical Journal B*, 49, Janvier 2006. Dans cet article, le modèle Susceptible-Infecté-Supprimé est expliqué et détaillé.

[3] Glenn Gebhart. Worm propagation and countermeasures. *Information Security Reading Room*, Février 2004. Une description comparative des vers les plus connus de l'histoire.

[4] Petter Holme. Efficient local strategies for vaccination and network attack. *Europhys. Lett.*, 68, Mars 2004. Dans cet article est décrit un bon nombre de stratégie. Nos stratégies testées proviennent principalement de cet article.

[5] Stuart Staniford Nicholas Weaver, Vern Paxson and Robert Cunningham. A taxonomy of computer worms. *Security and Protection Invasive Software*, Octobre 2003. Une description détaillée des différents types de vers.

[6] Wikipedia. *Percolation.*

[7] Wikipedia. *Ver informatique.*