Réduction de dettes mutuelles entre entreprises : heuristiques sur des graphes de paiements

Rapport de stage de L3

Balthazar Patiachvili

Sous la direction de
Sylvain Contassot-Vivier et de Nazim Fatès

Table des matières

Résumé

Le règlement de dettes entre différentes entreprises d'une même économie est un enjeu important pour son bon fonctionnement. Pour se faire, l'une des méthodes possible consiste à introduire un acteur extérieur venant financer une partie des dettes, ce qui entraînera un effet d'entraînement, permettant à une entreprise de rembourser une à qui elle est redevable, etc. L'objectif est de savoir, à partir d'un montant de financement maximal, quelles dettes rembourser pour que cela soit le plus profitable à l'économie en général. Ce problème est NP-complet, il faut donc trouver des heuristiques pour obtenir une bonne approximation rapide du résultat. Deux méthodes ont été étudiées et implémentées : un algorithme glouton polynomial et déterministe, ainsi qu'un algorithme génétique probabiliste. Des tests ont été effectués sur des graphes générés aléatoirement ainsi que sur un graphe de paiements tiré de données réelles anonymisées. Il apparaît alors que l'algorithme glouton apporte une solution satisfaisante pour une annulation de la moitié de la dette totale, tant temporellement que qualitativement, mais est moins bonne qu'une autre heuristique proposée dans un autre article pour une somme plus faible annulée. L'algorithme génétique, quant à lui n'apporte dans aucun cas une solution satisfaisante comparé à celui glouton, et ce peu importe les paramètres testés. Aussi continuer à améliorer les heuristiques proposées ne semble-t-il pas très pertinent, certaines améliorations ayant déjà été testées sans fournir des changements drastiques dans les résultats. Il semble donc que l'implémentation de ces heuristiques a permis d'avoir des résultats de référence qui pourront être utiles dans la recherche d'amélioration d'heuristiques existantes ou de nouveaux algorithmes.

1 Introduction

1.1 Contexte

Dans les échanges financiers entre entreprises en zone euro, le délai moyen de paiement est de trois mois. Cependant, durant cette période, de nombreuses factures peuvent s'accumuler, ce qui peut mettre en difficulté certains acteurs économiques, en particulier les plus vulnérables, voire les amener à faire faillite.

Pour éviter cela, deux solutions sont généralement mises en place : un prêt ou affacturage par une banque, ou la création d'une économie parallèle, telle que le WIR en Suisse, ou encore Sardex en Sardaigne [1]. Ce dernier est ainsi un réseau de petites et moyennes entreprises où les échanges effectués ne peuvent se faire qu'en dépensant les dettes dues par d'autres entreprises du réseau.

Nous allons étudier ici une potentielle troisième solution : la réduction des dettes mutuelles par aide initiale d'un acteur extérieur, généralement une banque ou un état. L'idée générale est que la somme injectée permettra de débloquer les situations où une entreprise doit attendre d'être payée avant de rembourser une dette grâce à un effet d'entraînement : une première pourra alors rembourser une seconde, qui pourra à son tour rembourser une troisième, etc.

Le problème général consiste donc à trouver où injecter cette somme – qui peut être séparée en plusieurs petits montants – pour maximiser la dette réduite, c'est-à-dire pour rembourser le plus grand montant possible.

1.2 Connaissances actuelles du problème

Le problème de la réduction de la dette mutuelle peut se diviser en deux sous-problèmes : les réductions partielle et intégrale.

La réduction partielle consiste à supposer que l'on peut financer seulement une partie d'une dette, et que le reste sera payé par la suite. On s'autorise ainsi à réécrire des dettes pour qu'elles soient financées partiellement. Ce problème peut se résoudre de manière exacte en temps polynomial (voir propriété 1). Cependant, cette solution n'est pas satisfaisante en pratique : avec le mode de fonctionnement classique de l'économie actuelle, il est complexe de modifier une dette déjà existante. Il faut donc trouver un autre modèle : la réduction intégrale.

La réduction intégrale quant à elle ne fait pas de telle supposition : le financement d'une arrête ne peut alors que se faire dans son intégralité. Ainsi posé, ce problème est NP-complet (voir propriété 2), et dans l'hypothèse où P est différent de NP, le problème ne peut donc pas être résolu en temps polynomial. Ce stage a donc pour objectif de trouver des heuristiques permettant de trouver une solution approchée à ce problème en un temps raisonnable pour des instances réelles de grande taille.

Il y a, à l'heure actuelle, peu de documentation sur le sujet : on peut citer un article de Massimo Amato, Nazim Fatès et Lucio Gobbi [2] — que l'on peut considérer comme le point de départ de ce stage — ou encore un article de Pătcaş Csaba György [3], mais qui s'autorise à ajouter des dettes en plus de celles déjà existantes, ce qui n'est pas réalisable en pratique. On peut également citer le travail effectué par deux précédents stagiaires s'étant penchés sur ce problème : Arthur Rousseau, s'est intéressé à la génération de graphes de paiements [4], ainsi que Marie Vela-Mena qui a étudié une heuristique basée sur le recuit-simulé.

Remarque On considère qu'une grosse instance est un échantillon de plusieurs dizaines de milliers d'entreprises pour plusieurs centaines de milliers voire millions d'échanges commerciaux. L'objectif serait dans ce cas de trouver une solution satisfaisante en une durée raisonnable.

Pour ce stage, on débutera par bien définir mathématiquement le problème ainsi que ses propriétés. Par la suite, on étudiera une première heuristique gloutonne que nous comparerons à un algorithme génétique, donnant de bons résultats pour des problèmes NP-complets. Chacun de ces deux heuristiques seront testées sur des graphes générés aléatoirement ainsi que sur un graphe pris à partir de données réelles.

1.3 Outils utilisés

Les heuristiques développées ont été programmées en Python 3.9 typé. L'entièreté des codes et des exemples utilisés pendant ce stage sont disponibles sur le gitlab INRIA suivant : https://gitlab.inria.fr/contasss/debt_reduction.

À noter que bien que la plupart des exemples soient générés, certains sont tirés de données réelles anonymisées. C'est sur ces derniers que les principaux tests d'évaluation des heuristiques ont été effectués.

2 Définition formelle du problème

2.1 Multigraphe

Définition 1: Multigraphe

Un *multigraphe orienté pondéré* est un couple $(\mathcal{V}, \mathcal{E})$ où :
- $\mathcal{V}$ est un ensemble quelconque dont les éléments sont appelés *sommets* ou *nœuds* ;
- $\mathcal{E}$ est un multi-ensemble de triplets (u, v, y) appelés *arcs* ou *arêtes* où $u, v \in \mathcal{V}$, et $y \in \mathbb{N}^*$. On dit que y est *le poids* de cet arc.

On notera dans toute la suite : $n = |\mathcal{V}|$ et $m = |\mathcal{E}|$.

D'un point de vue économique, les nœuds peuvent être vus comme des acteurs (généralement des entreprises), et les arcs comme des dettes entre entreprises. Ainsi, un arc (u, v, y) correspond à une dette de valeur y que l'entreprise u doit à l'entreprise v. $\mathcal{E}$ est donc bien un multi-ensemble et non un ensemble simple car il peut y avoir plusieurs dettes de même valeur entre deux entreprises.

Remarque Par ailleurs, si ce n'est pas précisé, il est sous-entendu dans toute la suite que « polynomial » signifie polynomial en fonction de n et m. De plus, on supposera dans toute la suite que les poids sont indépendants de n et de m.

2.2 Notions liées au problème

Soient $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe pondéré orienté tel que $\mathcal{E} \neq \emptyset$, $S \subseteq \mathcal{E}$ et $v \in \mathcal{V}$.

Définition 2: Somme réglée

On appelle *somme réglée* ou *settlement* de S, que l'on note $\Sigma(S)$ la somme des poids des arcs de S. On a ainsi : $\Sigma(S) = \sum_{(u,v,y)\in S} y$.

Définition 3: Position nette

On appelle *position nette de v par rapport à S*, que l'on note $\pi_S(v)$, la somme algébrique totale due à v : $\pi_S(v) = \sum_{(u,v,y)\in S} y - \sum_{(v,u,y)\in S} y$.

La position nette de v est positive si v reçoit plus d'argent qu'il n'en donne, et négative dans le cas contraire. On remarque par ailleurs que $\sum_{v\in\mathcal{V}} \pi_{\mathcal{E}}(v) = 0$.

Définition 4: Financement

On appelle *financement* de S, que l'on note $\phi(S)$, la somme totale à débourser pour annuler intégralement la dette dans le sous-graphe induit par S.
Cela se définit formellement par : $\phi(S) = \sum_{v \in \mathcal{V}} \max(0, -\pi_S(v))$.

Définition 5: Facteur d'amplification

On appelle *facteur d'amplification* de S, que l'on note $\alpha(S)$, le rapport entre la somme annulée et le financement. On a donc : $\alpha(S) = \dfrac{\Sigma(S)}{\phi(S)}$.
On prend pour convention :
- $\alpha(\emptyset) = 0$
- $\phi(S) = 0$ et $\Sigma(S) > 0 \implies \alpha(S) = +\infty$

Définition 6: Facteur d'inclusion

On appelle *facteur d'inclusion* de S, que l'on note $\iota(S)$, le rapport entre la somme annulée de S et la somme totale des arêtes de $\mathcal{E}$. On a donc : $\iota(S) = \dfrac{\Sigma(S)}{\Sigma(\mathcal{E})}$.

Définition 7: Problème de réduction partielle de dettes mutuelles

On appelle *problème de réduction partielle de dettes mutuelles* le problème suivant.
Soient $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe et $Q \in \mathbb{N}^*$.
Trouver $S \subseteq \mathcal{V} \times \mathcal{V} \times \mathbb{N}$ tel que $\forall (u, v, y) \in \mathcal{E}$, $\exists y' \in [\![0, y]\!] \mid (u, v, y') \in S$, $\phi(S) \leqslant Q$ et $\Sigma(S)$ est maximal parmi les sous-ensembles S de $\mathcal{V} \times \mathcal{V} \times \mathbb{N}^*$ vérifiant les deux conditions ci-dessus.

Dans cette définition, $\mathcal{G}$ est le multigraphe représentant l'économie considérée, à savoir les acteurs économiques ainsi que leurs échanges, Q le financement maximal qu'un tierce injectera dans l'économie, et S est un ensemble d'arêtes, donc un ensemble de transactions, telles que le financement ne dépasse pas Q, et que la somme totale annulée est maximale.

La première condition représente ici le fait que l'on peut financer les arêtes de manière partielle et non pas de manière totale.

Définition 8: Problème de réduction intégrale de dettes mutuelles

On appelle *problème de réduction intégrale de dettes mutuelles* le problème suivant.
Soient $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe et $Q \in \mathbb{N}^*$.
Trouver $S \subseteq \mathcal{E}$ tel que $\phi(S) \leqslant Q$ et $\Sigma(S)$ est maximal parmi les sous-ensembles S de $\mathcal{E}$ vérifiant $\phi(S) \leqslant Q$.

Les variables ont la même signification que dans la définition précédente, à l'exception de S qui est désormais un sous-ensemble de $\mathcal{E}$, donc un ensemble de transactions, qui seront intégralement financées.

2.3 Propriétés du problème

Propriété 1: Résolution en temps polynomial

Le problème de réduction partielle de dettes mutuelles peut se résoudre en temps polynomial en fonction de $|\mathcal{V}|$, $|\mathcal{E}|$.

Idée de la preuve Preuve complète en annexe A.

On commence par éliminer les cycles du graphe, ce qui peut se faire par l'algorithme de Ford-Fulkerson en temps polynomial, puis l'on obtient alors un graphe acyclique.

Ensuite, tant que la limite de financement n'a pas été atteinte, on applique l'algorithme de Ford-Fulkerson à ce graphe.

Propriété 2: NP-complétude

Le problème de réduction intégrale de dettes mutuelles est NP-complet.

Idée de la preuve Preuve complète en annexe B.

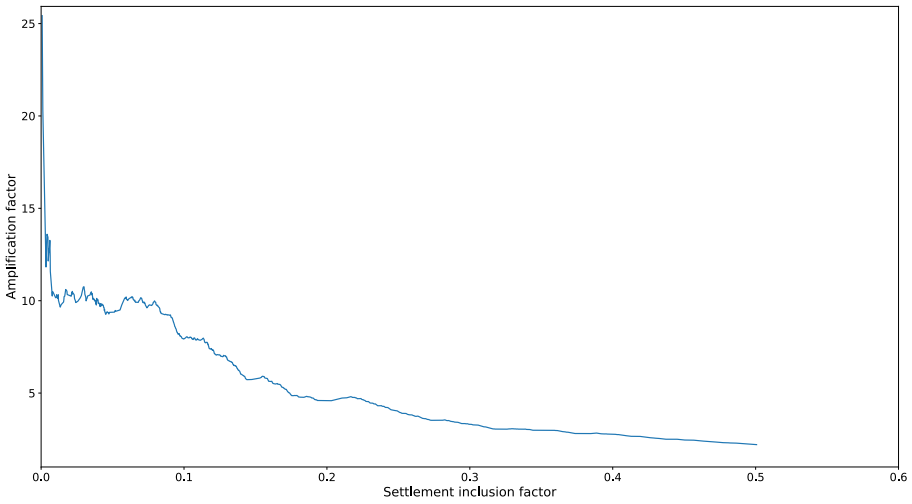
La preuve de la NP-difficulté se fait par réduction depuis le problème du sac à dos, et a déjà été formalisée dans la littérature [5].

Quant à la preuve de l'appartenance à NP, je n'ai trouvé qu'une seule occurrence de celle-ci dans la littérature [6], mais elle me semblait un peu lacunaire, aussi ai-je donc décidé d'en réécrire une nouvelle qui utilise le problème décisionnel associé.

Remarque En pratique, ce problème n'est pas exactement celui que nous allons optimiser. En effet, certains critères économiques sont plus intéressants dans la réalité : en l'occurrence, l'objectif exact est, à partir d'un multigraphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, trouver $S \subseteq \mathcal{E}$ tel que $\iota(S) \geqslant 0,5$ et $\alpha(S)$ maximal. Ce problème est également NP-complet ; la preuve est la même que pour le problème ci-dessus à quelques détails près.

Toute la difficulté de ce problème réside dans le compromis que l'on observe entre les facteur d'inclusion et d'amplification. En effet, lors d'une recherche de solution, on remarque que le facteur d'amplification décroît avec l'augmentation du facteur d'inclusion.

Voici un exemple de courbe du facteur d'amplification en fonction du facteur d'inclusion que l'on peut obtenir en construisant une solution.



3 Algorithme glouton

3.1 Description de l'approche

Cette méthode a été la première à être implémentée. Elle consiste à calculer un potentiel minimal de réduction pour chaque arête, puis à choisir de financer l'arête avec le potentiel le plus haut. On répète ensuite cette opération tant que l'objectif fixé initialement n'est pas atteint, en l'occurrence tant que le facteur d'inclusion n'a pas dépassé 50%.

Plus précisément, on cherche à trouver successivement les nœuds où un financement annulera le plus de dette possible. À partir d'un financement, on crée alors un chemin glouton initial avec l'algorithme 1, puis on lui ajoute des branches qui sont également des chemins gloutons, et ce tant qu'il est possible de le faire sans augmenter le financement. Pour cela, on sauvegarde sur chaque nœud son montant de financement restant, qui correspond à la différence entre ce qu'il donne et ce qu'il reçoit.

Il s'agit donc de trouver un moyen efficace de calculer la somme maximale de dette totale retirée après avoir injecté une somme Q dans un nœud v.

3.1.1 Calcul du potentiel et chemin glouton

Définition 9: Problème du calcul du potentiel

Soient $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe, $v \in \mathcal{V}$ et $Q \in \mathbb{N}^*$.
On cherche $S \subseteq \mathcal{E}$ tel que $\pi_S(v) \geqslant -Q$, $\forall u \in \mathcal{V} \setminus \{v\}$, $\pi_S(u) \geqslant 0$, et $\Sigma(S)$ maximal.

Propriété 3: NP-complétude

Le problème du calcul du potentiel est NP-complet.

Preuve Voir annexe C

Ce problème est également NP-complet, il faut donc trouver une méthode pour avoir une bonne approximation. Trois idées ont émergé, et toutes se basent sur le même principe : faire des chemins en prenant à chaque fois l'arête sortante de poids maximal tel qu'aucun financement supplémentaire ne soit nécessaire.

Algorithme 1: Calcul d'un chemin glouton

On part d'un sommet $v \in \mathcal{V}$ et d'un montant $Q \in \mathbb{N}$.

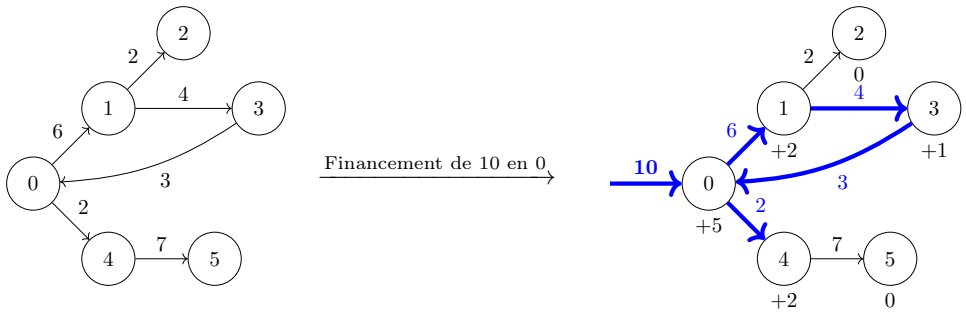
On choisit l'arête de poids maximal parmi les arêtes sortantes de v telle que son poids y est inférieure à Q. Ensuite, on prend cette arête que l'on ajoute à la solution qui va être renvoyée à la fin, et on sauvegarde le montant de financement restant $Q - y$ sur le nœud v.

On réitère cette opération sur le nœud au bout de l'arête choisie, en vérifiant que le poids de la nouvelle arête choisie est maximal parmi les poids ne dépassant pas la somme du poids de l'arête précédente et du montant de financement restant du nœud actuel.

On termine le processus quand il n'y a plus d'arête disponible.

Exemple 1 Exécution de l'algorithme 1

On applique ici l'algorithme 1 après injection d'un montant de 10 au nœud 0.



Avant financement — Après financement

On obtient alors le chemin $0 \to 1 \to 3 \to 0 \to 4$ en fin d'algorithme.

Propriété 4: Terminaison et correction de l'algorithme 1

L'algorithme 1 termine et renvoie un ensemble d'arêtes $S \subseteq \mathcal{E}$ tel que $\phi(S) \leqslant Q$.

Preuve Preuve complète en annexe D.

Propriété 5: Complexité temporelle de l'algorithme 1

L'algorithme 1 a une complexité temporelle de $\mathcal{O}(m^2)$.

Preuve À chaque étape de l'algorithme, on cherche une arête maximale pour le poids en $\mathcal{O}(m)$, et l'algorithme itère sur au plus toutes les arêtes, à savoir m.

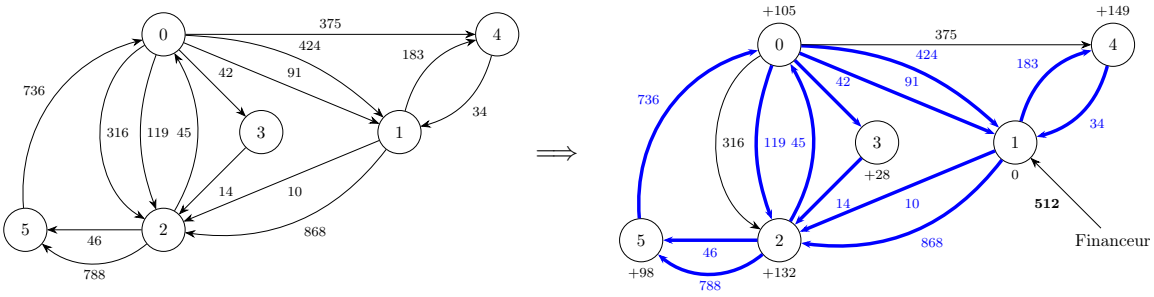
3.1.2 Calcul final du potentiel

Après une exécution de l'algorithme 1, on se rend compte que l'on peut continuer à choisir de nouvelles arêtes sans pour autant augmenter le financement. Pour éviter cela, trois solutions ont été testées. Celles-ci se basent sur le

même principe : parcourir les nœuds déjà visités pour tester si leur position nette par rapport à l'ensemble d'arêtes déjà prises leur permet de financer une nouvelle arête, auquel cas on réapplique l'algorithme 1 sur ce nœud.

Les algorithmes qui ont été testés sont les suivants :
— POT : sélection du nœud avec la plus grande position nette actuelle ;
— BFS : sélection des nœuds par ordre de parcours lors de la dernière exécution de l'algorithme 1 ;
— DFS : identique à BFS, à ceci près que l'on considère l'ordre inverse de parcours.

Exemple 2 Exécution de l'algorithme POT (exécution par étape en annexe E)



Avant financement — Après financement

En notant S l'ensemble des arêtes bleues de la configuration de droite, on a : $\Sigma(S) = 3\ 400$, $\phi(S) = 512$, donc $\alpha(S) = \dfrac{3\ 400}{512} \approx 6,64$. De plus, on a $\Sigma(\mathcal{E}) = 4\ 091$, d'où $\iota(S) = \dfrac{3\ 400}{4\ 091} \approx 83,1\%$.

Cet algorithme paraît alors satisfaisant, mais on peut déjà remarquer qu'il n'est pas optimal, c'est-à-dire qu'il ne maximise ni le facteur amplification ni le facteur d'inclusion pour un financement maximal donné, ici 512. En effet, pour le même graphe de départ, il est possible de trouver une solution S' telle que $\phi(S') = 480$ et $\iota(S') \approx 84,9\%$, et une autre S'' telle que $\phi(S'') = 353$ et $\alpha(S'') \approx 9,08$ (voir annexe F).

Propriété 6: Terminaison et correction de l'algorithme POT

L'algorithme POT termine et renvoie un ensemble d'arêtes $S \subseteq \mathcal{E}$ tel que $\phi(S) \leqslant Q$.

Preuve La preuve se fait exactement de la même façon que pour la propriété 4.

Propriété 7: Complexité temporelle de l'algorithme POT

La complexité temporelle de l'algorithme POT est de $\mathcal{O}(nm(\log(n) + m^2))$.

Preuve Lors d'un tour de boucle, l'algorithme POT trie tous les nœuds par position nette dans le sous-graphe solution, ce qui peut se faire en $\mathcal{O}(n\log(n))$, puis itère l'algorithme 1 en $\mathcal{O}(m^2)$ (voir propriété 5) tant qu'aucune arête n'a été ajoutée et qu'il reste un nœud non parcouru, c'est-à-dire au plus n.

Or, à chaque tour de boucle, on ajoute au-moins une arête, donc on itère au plus m fois cette opération de complexité $\mathcal{O}(n\log(n) + nm^2)$. La complexité totale de l'algorithme POT est donc de $\mathcal{O}(nm(\log(n) + m^2))$.

3.1.3 Algorithme complet

Maintenant que nous avons une méthode pour calculer le potentiel d'un nœud après injection d'un montant donné, nous pouvons définir l'algorithme glouton complet.

Algorithme 2: Algorithme glouton

Pour chaque arête du graphe, on calcule le potentiel si on la finançait, c'est-à-dire si l'on injectait un montant égal au poids de l'arête en son sommet source. Puis, on ajoute à la solution celle de l'arête avec le plus grand potentiel, et on recommence cette opération tant que l'on n'a pas atteint un critère donné et qu'il reste des arêtes disponibles.

Remarque Lors des tests, le critère donné était $\iota(S) \geqslant 50\%$, c'est-à-dire que le montant annulé total par le financement doit concerner au moins 50% du total. Ce critère est lié à des considérations économiques : annuler la moitié de la dette globale permettrait d'améliorer grandement la situation d'une grande partie des entreprises de l'économie concernée.

Propriété 8: Complexité temporelle de l'algorithme 2

La complexité temporelle de l'algorithme 2 est de $\mathcal{O}(nm^3(\log(n) + m^2))$.

Remarque Dans la mesure où l'on ne considère dans ce problème que les nœuds qui possèdent au moins une arête entrante ou sortante — car des nœuds isolés correspondent à des entreprises ne participant pas à l'économie — on a $m \geqslant n$, et donc la complexité de cet algorithme dans notre cadre peut se simplifier en $\mathcal{O}(nm^5)$.

Preuve À chaque tour de boucle, on teste pour au plus m arêtes l'algorithme POT, ce qui s'effectue donc en $\mathcal{O}(nm^2(\log(n) + m^2))$. De plus, on itère ce processus au plus m fois (dans le pire cas, on ajoute qu'une arête à la fois). La complexité temporelle globale est donc de $\mathcal{O}(nm^3(\log(n) + m^2))$.

3.2 Résultats des tests

3.2.1 Modifications de l'algorithme

Après de nombreux tests préliminaires, il apparaît que l'algorithme a deux problèmes corrigeables en l'état :

1. le temps d'exécution est trop long pour les moyennes instances (quelques centaines de nœuds) ;
2. le calcul du potentiel ne fait pas prendre les meilleures arêtes à l'algorithme.

Pour résoudre ces problèmes, les modifications suivantes ont été effectuées :

1. dans l'algorithme 2, au lieu de tester toutes les arêtes à chaque tour de boucle, on teste seulement les k arêtes de poids le plus élevé, avec k constant, ce qui peut se faire sans surcoût de complexité en triant initialement la liste de toutes les arêtes par poids décroissants ; la complexité de l'algorithme 2 devient alors $\mathcal{O}(nm(\log(n) + m^2))$;
2. le calcul du potentiel d'une arête n'est plus seulement la somme de tous les poids d'arêtes pouvant être financées sans apport supplémentaire, mais peut être n'importe quelle fonction prenant un graphe en entrée et renvoyant un flottant.

Il a également été essayé de tester les k arêtes avec le score initial le plus élevé, il fallait donc faire un calcul préliminaire pour toutes les arêtes, mais le résultat obtenu était en moyenne moins bon : le facteur d'amplification était plus bas, et le temps de calcul était plus élevé.

Remarque Après plusieurs tests, il est apparu que la fonction de score permettant d'évaluer une solution donnant le potentiel le plus efficace était celle ne prenant en compte que l'amplification qu'aurait la solution si elle incluait toutes les arêtes pouvant être financées par celle considérée.

Exemple 3 Exemple d'exécution de l'algorithme en annexe G

3.2.2 Résultats sur des graphes générés aléatoirement

Remarque La génération aléatoire de graphes pour ce problème demande la connaissance de la structure des graphes de paiements, qui est particulière. Une étude de cette structure a été menée par Arthur Rousseau, ancien stagiaire de l'équipe du Mocqua dans le même laboratoire, pour identifier les caractéristiques propres à ces graphes. C'est donc à partir de son travail que la génération aléatoire des graphes a été faite.

Les tests ont été effectués sur des graphes possédant les caractéristiques suivantes : 96 sommets et 333 arêtes, 447 sommets et 1 666 arêtes, 868 sommets et 3 325 arêtes (voir figure I), 1 689 sommets et 6 666 arêtes, 2 505 sommets et 10 032 arêtes, 4 137 sommets et 16 680 arêtes, et 8 272 sommets et 33 239 arêtes. Les courbes pour tous ces tests sont disponibles en annexe H.

La courbe ci-dessous représente le facteur d'amplification en fonction du facteur d'inclusion : chaque point bleu correspond à la solution partielle obtenue après une itération de l'algorithme 2. Le nombre d'arêtes testées correspond au k décrit dans la partie 3.2.1. Les tests ont tous duré entre 5 et 20 minutes, le temps augmentant avec la taille des graphes.

De plus, le point rouge correspond à la solution optimale pour ce graphe, c'est-à-dire la solution ayant le plus grand facteur d'amplification avec un facteur d'inclusion d'au moins 50%. La méthode d'obtention de ces solutions optimales est détaillée en annexe I.

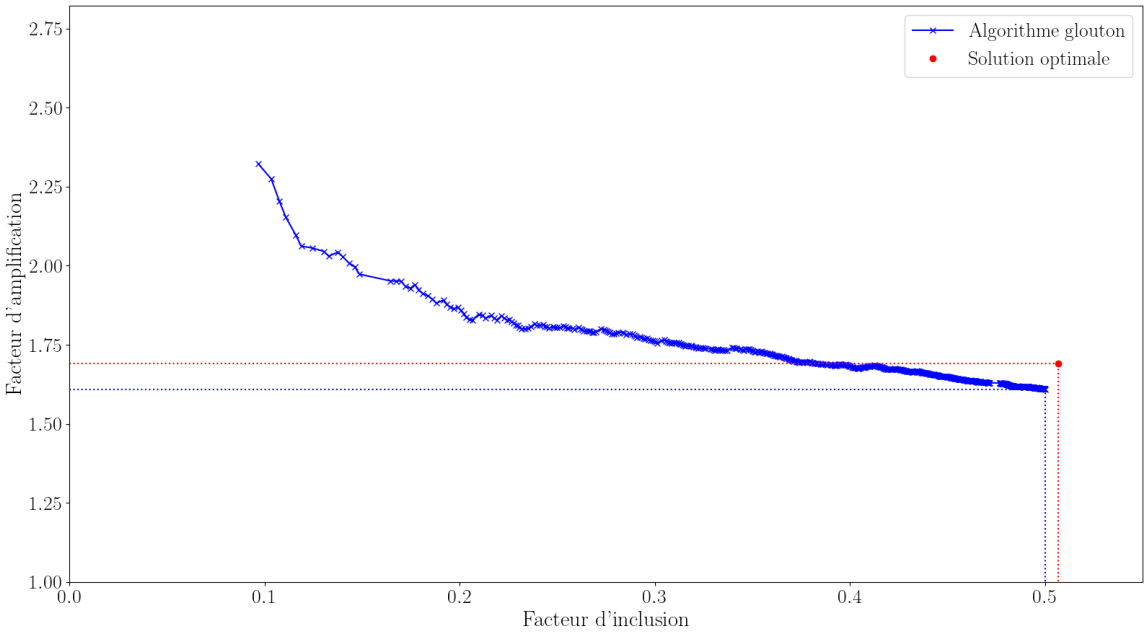


FIGURE I – Résultat de l'algorithme glouton pour un graphe de 868 sommets et 3 325 arêtes avec 50 arêtes testées

On remarque sur les premiers graphes, où la solution optimale a pu être calculée, que pour un facteur d'inclusion d'au moins 50%, l'algorithme glouton n'atteint pas la solution optimale, mais il est tout de même assez proche. Là où cet algorithme montre réellement ses limites est sur toute la partie précédant le résultat final, et en particulier la bande $[0; 0,3]$ de facteur d'inclusion, où l'algorithme présenté comme naïf dans l'article [2] de Massimo Amato, Nazim Fatès et Lucio Gobbi a de bien meilleurs résultats sur des facteurs d'inclusions plus petits.

Pour corriger ce problème, plusieurs solutions ont été imaginées :
— Considérer les chemins arrières : on peut en effet remarquer qu'en ajoutant à un chemin de l'algorithme 1 existant des arêtes en amont dont le poids est plus faible que l'arête suivante, alors le financement n'est plus injecté que sur un nœud mais potentiellement sur plusieurs, et que celui-ci n'augmente pas. Cette solution donne soit les mêmes résultats soit des résultats légèrement meilleurs que précédemment en réduisant le nombre d'étapes mais en allongeant légèrement la durée de chaque itération.
— Forcer les cycles : en imposant aux chemins créés avec l'algorithme 1 de repasser le plus souvent possible sur le sommet initial pour diminuer le financement et ainsi augmenter le facteur d'inclusion. Après reconduite des tests, l'impact de ce changement est négligeable par rapport aux résultats précédents.

Cependant, comme expliqué en annexe I, à l'heure actuelle, il n'existe aucune méthode permettant de résoudre exactement et rapidement ce problème pour des graphes de plusieurs dizaines de milliers de nœuds. Il faut donc trouver un moyen de comparaison de cet algorithme glouton : l'algorithme génétique.

4 Algorithme génétique

Cette méthode, qui a déjà donné de bon résultats pour d'autres problèmes NP-complets, comme pour le problème du sac à dos [7], a été développée afin de pouvoir comparer l'algorithme glouton à une méthode ayant déjà fait ses preuves. En effet, les algorithmes génétiques sont une approche classique dans la recherche d'heuristiques pour résoudre un problème NP-complet. Une telle implémentation permettra d'observer le comportement du problème en pratique, c'est-à-dire de voir s'il sera envisageable d'utiliser pour ce problème des heuristiques efficaces pour d'autres problèmes NP-complet.

4.1 Principe des algorithmes génétiques

Les algorithmes génétiques sont une famille d'algorithmes dont le principe s'inspire de la théorie de l'évolution pour obtenir une solution approchée à un problème d'optimisation lorsqu'il n'existe pas de méthode exacte connue pour résoudre ce problème en un temps raisonnable.

Ces algorithmes manipulent des populations qui évolueront au fil d'une itération de plusieurs processus spécifiques après avoir débuté d'une population initiale. On retrouve généralement les étapes suivantes :
— **sélection** d'une partie de la population selon leur évaluation (voir ci-dessous) ;
— **croisement** des individus sélectionnés pour créer de nouveaux individus : les individus créés possèdent en général deux parents avec lesquels ils partageront leur caractéristiques ;

— **mutation** d'une partie des individus sélectionnés (généralement probabiliste) ;
— **évaluation** de chaque individu par une fonction d'évaluation ;
— **remplacement** de la population actuelle par celle générée lors des précédents processus.

Toutes ces étapes peuvent être probabilistes, ce qui permet d'explorer un éventail de solutions plus important que pour un algorithme déterministe.

Plusieurs conditions d'arrêt peuvent être mises en place, les plus communes étant : un nombre fixe d'itérations, un temps d'exécution fixe, répéter tant qu'un critère n'a pas été atteint.

4.2 Implémentation pour ce problème

Dans le cadre du problème de la réduction de dettes mutuelles, chaque individu est un ensemble d'arêtes du graphe à réduire, et la fonction de score est définie par :

$$\textbf{score}(S) = \begin{cases} \alpha(S) & \text{si } \iota(S) \in [0.50, 0.55] \\ \frac{\alpha(S)}{2} \times (1 - 2\,|\iota(S) - 0.5|) & \text{sinon} \end{cases}$$

Cette fonction permet de favoriser les individus proches de 50% d'inclusion tout en maximisant l'amplification (représentation graphique en annexe J). D'autres fonctions polynomiales en l'amplification et l'inclusion ont été testées, mais nous avons retenu celle donnant les meilleurs résultats.

La représentation de chaque individu en mémoire était initialement un graphe, mais cela n'était pas efficace temporellement. La deuxième implémentation était une liste d'arêtes stockées par indice unique, mais était également trop peu efficace, surtout en ce qui concerne la recherche de la présence ou non d'une arête spécifique. La dernière implémentation, et également la plus efficace, est le codage binaire, à savoir représenter chaque individu par une liste de booléens où chaque booléen représente la présence ou non d'une arête.

L'algorithme qui a été implémenté est le suivant :
— Génération initiale d'une population de taille N fixé.
— Calcul du score de chaque individu de la population initiale.
— Itération des étapes suivantes pour un nombre d'étapes fixé :
  — Une partie des individus avec les meilleurs scores est gardée pour la nouvelle population.
  — Une partie des individus restants avec les meilleurs scores est utilisée pour les croisements ; les individus parents et enfants sont ajoutés à la nouvelle population.
  — Une partie des individus restants avec les meilleurs scores est mutée : pour chaque individu, on effectue un nombre fixe de modifications de gènes.
  — Les individus restants, donc ceux avec les scores les plus faibles, sont remplacés par de nouveaux individus générés aléatoirement.

Génération d'individus Pour générer un individu, la méthode utilisée consiste à partir d'un ensemble d'arêtes vide, et d'ajouter une arête choisie aléatoirement parmi celles du graphe initial qui ne l'ont pas déjà été, tant qu'une condition n'est pas vérifiée. Pour se rapprocher le plus de l'objectif de l'algorithme, et ainsi limiter au plus le temps de calcul, cette condition était : $\iota(S) \geqslant 0.45\%$. Cependant, cette dernière avait également un impact négatif sur la résolution du problème : la distribution de la population initiale est très inégalement répartie.

Croisements Le croisement se fait à partir de deux individus parents qui vont en générer un nouveau. Contrairement aux autres étapes de l'algorithme génétique, l'implémentation de celle-ci dépend beaucoup de la représentation mémoire des individus. Avec le codage binaire, le croisement se fait facilement en « coupant » les deux parents à un même indice, puis en constituant un individu en prenant la première partie du premier parent, et la seconde partie du second (voir figure II).

Mutations Pour muter un individu, l'implémentation choisie est de tirer une arête du graphe initial au hasard : si elle est déjà présente dans l'individu on la retire, sinon on l'ajoute, et on répète cela un nombre fixé de fois. Pour le codage binaire, cela revient à appliquer le « non » logique sur des bits aléatoires d'un individu (voir figure III). Avec ce principe, une même arête peut être ajoutée et retirée plusieurs fois durant une seule phase de mutation, mais ce phénomène est négligeable compte tenu de sa faible probabilité.

Sélection L'objectif de cette partie est de maintenir une population de taille constante. Or, lors des croisements, plusieurs individus ont été générés, il faut donc supprimer autant d'individus de la nouvelle population. La méthode utilisée pour la sélection est celle de la roulette russe : on calcule pour chaque individu i son score s_i, et on choisit aléatoirement un individu, de sorte que chaque individu i ait une probabilité $\dfrac{s_i}{\sum s_i}$ d'être choisi. On répète alors cette étape autant de fois qu'il n'y a d'individus dans une population.

Exemple 4 Illustration d'un croisement et d'une mutation

Chaque ligne correspond à un individu, et chaque case à une arête : elle est présente si la case est un 1, et ne l'est pas si la case est un 0.

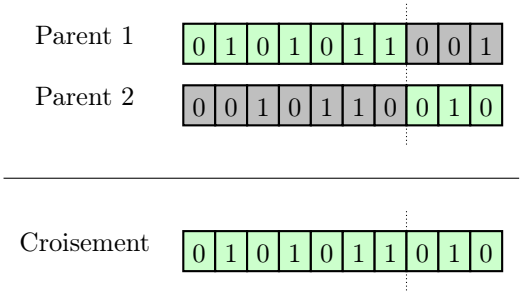


FIGURE II – Schéma d'un croisement d'individus représentés sous forme binaire

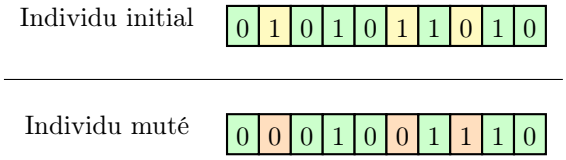


FIGURE III – Schéma d'une mutation d'un individu représenté sous forme binaire

4.3 Résultats des tests

4.3.1 Modification de l'algorithme

Les premiers tests avec l'algorithme génétique mettent en évidence un problème : l'algorithme évolue bien trop lentement. Une solution a donc été mise en place : étendre les individus, c'est-à-dire pour chaque individu de la population, ajouter le maximum d'arêtes qui peuvent être prises sans augmenter le financement. On utilise pour cela l'algorithme POT décrit dans la partie 3.1.2. Cela a augmenté le facteur d'inclusion plus tôt dans le nombre d'itérations, mais a ralenti considérablement le temps d'exécution de chaque itération. Cette opération est tout de même nécessaire pour obtenir les meilleurs résultats potentiels pour chaque individu, et a donc été conservée.

La seconde modification est l'inversement du processus de sélection : étant donné qu'il y a plus d'individus qui sont gardés d'une itération à la suivante que ceux qui ne le sont pas, on garde le même principe en remplaçant les probabilités de chaque individu i par $1 - \frac{s_i}{\sum s_i}$, et celle-ci ne désigne plus la probabilité d'être pris mais la probabilité d'être retiré. Une conséquence directe est qu'un individu ne peut plus être ajouté plusieurs fois à la population suivante.

4.3.2 Résultats sur des graphes générés aléatoirement

Les tests ont été effectués sur les mêmes graphes que pour l'algorithme glouton. La figure IV correspond au test sur le même graphe que la figure I. Les résultats complets sont disponibles en annexe K.

Tous les tests ont été effectué avec une population de 100 individus, les proportions utilisées étant :
- 10% des individus avec les meilleurs scores ont été gardés ;
- 35% des individus restants avec les meilleurs scores ont été utilisés pour faire des croisements, et chaque croisement était réalisé en scindant les parents en leur milieu ;
- 35% des individus restatns avec les meilleurs scores ont été mutés à hauteur de 20% de leurs gènes ;
- tous les individus restants, donc 20% de la population, ont été remplacé par des individus nouvellement générés.

De plus, la condition d'arrêt était temporelle : l'algorithme itérait jusqu'à ce que le temps total d'exécution dépasse une durée fixée.

Le nuage de points ci-dessous représente le facteur d'amplification en fonction du facteur d'inclusion : chaque point bleu correspond à la meilleure solution d'une population à chaque itération de l'algorithme génétique. Le point rouge est le même que celui visible sur le tests de l'algorithme glouton. Son obtention est décrite en annexe I.

On peut donc voir que l'algorithme génétique est très peu efficace pour des durées équivalentes à l'algorithme glouton avec ces paramètres. Par ailleurs, après de nouveaux tests, il est apparu que la taille de la population ne modifie pas significativement ces résultats. Quelques changements des paramètres (proportions d'individus gardés, croisés, mutés) ont été testés, mais cela n'a pas significativement modifié les résultats.

Il faudrait confirmer ces résultats en essayant de passer plus de temps à calibrer les paramètres et les étapes de chaque itération pour obtenir un algorithme plus performant. Cependant, il semble que cela n'a que peut de chances de modifier radicalement le comportement de l'algorithme génétique. En effet, le problème de cet algorithme est son manque d'évolution : trouver une nouvelle configuration meilleure que la précédente à partir de mutations est très peu probable pour de grands graphes, et le croisement de deux individus n'a que peu de chances de donner un individu meilleur que ses parents à cause de l'influence qu'à chaque arête sur les autres dans le calcul de l'amplification.

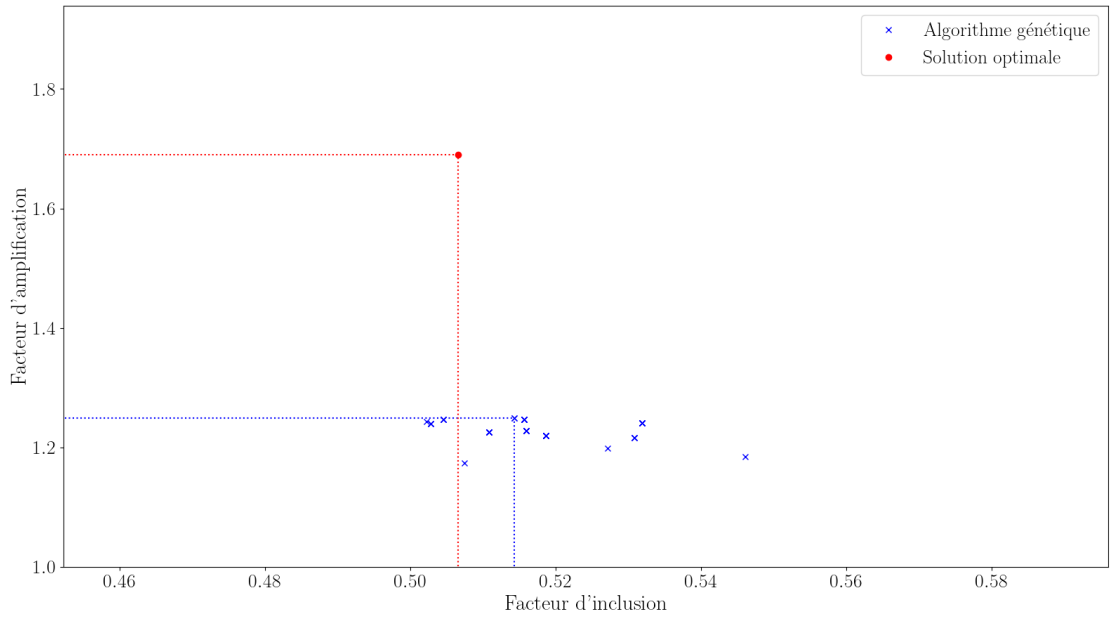


FIGURE IV – Résultat de l'algorithme génétique pour un graphe de 868 sommets et 3 325 arêtes en 300 secondes

5 Comparaison des algorithmes glouton et génétique

La comparaison entre les deux algorithmes sur des graphes générés aléatoirement montre que l'algorithme glouton est globalement meilleur que le génétique, mais il est nécessaire de vérifier ce résultat sur ce qui nous intéresse le plus ici : un graphe de paiement tiré de données réelles anonymisées. Celui-ci comporte 38 334 nœuds et 145 540 arêtes, ce qui en fait le plus gros graphe sur lequel les deux algorithmes ont été testé.

L'algorithme glouton a été exécuté en testant à chaque itération les 100 arêtes de poids maximal (voir 3.2.1). Le test a duré au total 3 480 secondes, soit environ une heure, et 3 735 étapes ont été nécessaires pour arriver à la solution finale.

L'algorithme glouton a été exécuté avec une limite de temps de 3 600 secondes, donc une heure également, avec les mêmes paramètres que lors de la réalisation des tests sur les graphes aléatoires (voir partie 4.3.2). Au total, il y a eu 24 générations d'individus.

Les résultats sont sur la figure V ci-dessous. Chaque point rouge correspond au meilleur individu de chaque génération de l'algorithme génétique, et la courbe bleue correspond au lissage de l'ensemble des points générés par l'algorithme glouton : le nombre de points est trop important rapporté à la taille du graphe pour qu'il soit pertinent de tous les afficher.
Ici encore, on peut voir que pour une même durée d'exécution, l'algorithme glouton donne de meilleurs résultats que l'algorithme génétique. Il est certain, par la nature probabiliste de l'algorithme génétique, que ce dernier finira par donner de meilleurs résultats que l'algorithme glouton avec un temps beaucoup plus long d'exécution, mais cela n'est pas souhaitable compte tenu de l'objectif fixé qui est d'obtenir une bonne approximation en un temps raisonnable.

6 Conclusion

Lors de ce stage, deux heuristiques ont été implémentées pour trouver des solutions approchées au problème de réduction intégrale de dettes mutuelles. Plusieurs améliorations et tentatives d'améliorations ont également été ajoutées afin d'améliorer les performances des algorithmes proposés, autant sur le plan temporel que sur l'approximation avec la solution optimale.

On a tout d'abord étudié un algorithme glouton essayant de prendre les arêtes avec les potentiels de réduction les plus élevés, puis une implémentation d'un algorithme génétique pour pouvoir comparer la performance de l'algorithme glouton, ainsi que des potentiels futurs heuristiques développées pour ce problème, avec un algorithme ayant déjà donné d'excellents résultats pour d'autres problèmes NP-complets. Ces deux heuristiques ont pu être comparées sur différents graphes : certains ont été générés aléatoirement, tandis que le dernier test s'est effectué sur des données réelles. Il apparaît alors que l'algorithme glouton est plus performant que l'algorithme à durées d'exécution égales :

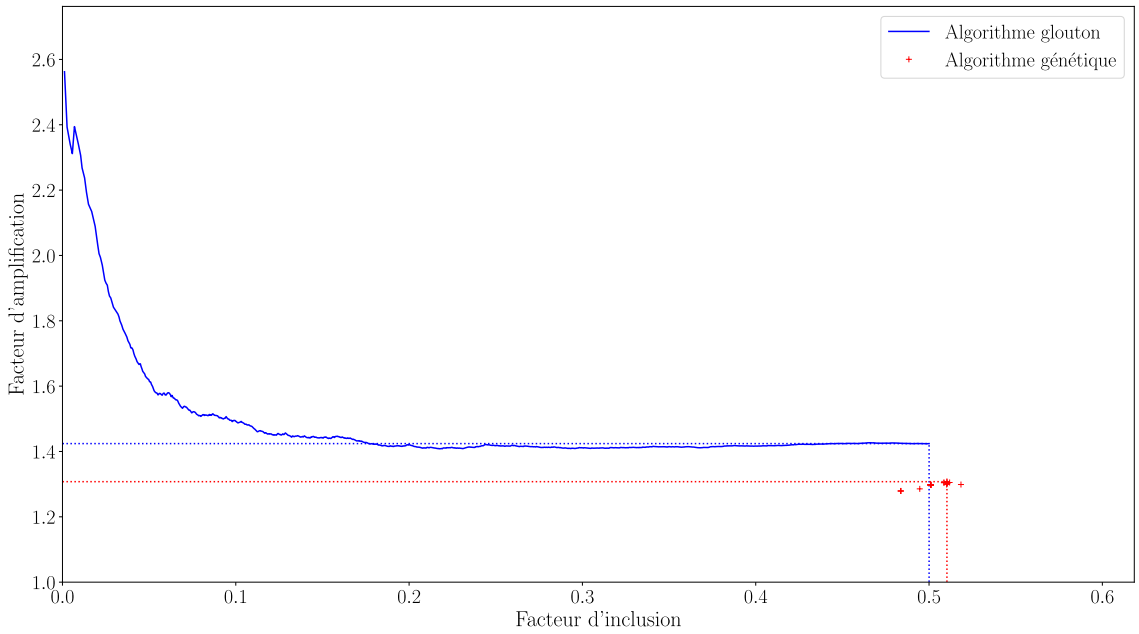


FIGURE V – Comparaison des algorithmes glouton et génétique pour un graphe de 38 334 nœuds et 145 540 arêtes

les solutions obtenues par l'algorithme glouton ont dans tous les tests réalisés un facteur d'amplification meilleur que pour l'algorithme génétique.

Cependant, les résultats obtenus ici sont moins bons que ceux présentés dans l'article [2], il faut donc améliorer les deux heuristiques proposées, ce qui semble peu prometteur, ou tenter une nouvelle approche à ce problème. Plusieurs pistes ont été discutées pendant ce stage sans pour autant pouvoir les réaliser par manque de temps :

— réaliser une coupe du graphe pour résoudre rapidement chaque sous-partie indépendamment des autres très rapidement (par exemple avec CPLEX, présenté en annexe I), puis construire une solution générale à partir des solutions partielles (un premier résultat a été brièvement étudié et est disponible en annexe) ;
— résoudre la version partielle du problème sur le même graphe en un temps polynomial (voir propriété 1), puis convertir la solution du problème partiel en une solution du problème intégral par un moyen qui reste à définir.

Une dernière problématique qu'il serait intéressant d'étudier est la structure des graphes, travail commencé par Arthur Rousseau durant un stage dans le même laboratoire ([4] et [5]), qui pourrait permettre de trouver des algorithmes, ou du moins des paramètres plus adaptés à ces graphes de paiements.

Finalement, l'implémentation de ces deux algorithmes a permis d'avoir des résultats de référence qui pourront être utilisés pour comparer les solutions proposées par de nouvelles heuristiques ou des versions modifiées de celles déjà existantes.

Remerciements

Je remercie toute l'équipe SIMBIOT du LORIA de m'avoir accueilli chaleureusement pendant ces deux mois.

Je remercie Massimo Amato, Lucio Gobbi et Daniele Cittero pour nous avoir fourni les données réelles sur lesquelles les tests finaux ont été réalisés.

Je remercie Ammar Oulamara pour son aide ainsi que ses explications sur l'optimisation sur les méthodes de résolution exacte des problèmes de réduction mutuelle de dettes.

Je remercie énormément Sylvain Contassot-Viver et Nazim Fatès pour m'avoir accompagné durant ces deux mois de stage en me donnant des conseils théoriques et pratiques en informatique, mais également sur le monde de la recherche en général. L'existence de ce stage n'a été possible que grâce à eux.

Références

[1] G. Iosifidis, Y. Charette, E. Airoldi, G. Littera, L. Tassiulas et N. Christakis, « Cyclic motifs in the Sardex monetary network, » *Nature Human Behaviour*, 2018. adresse : `https://doi.org/10.1038/s41562-018-0450-0`.

[2] M. Amato, N. Fatès et L. Gobbi, « The economics and algorithmics of an integral settlement procedure on B2B networks, » 2021. adresse : `https://dx.doi.org/10.2139/ssrn.3915380`.

[3] P. C. György, « The debts' clearing problem : a new approach, » *Acta Univ. Sapiente*, 2011. adresse : `https://doi.org/10.48550/arXiv.1111.3663`.

[4] A. Rousseau, « Génération de graphes pour la compensation de dettes mutuelles entre entreprises, » 2021, Rapport de stage de L3. adresse : `https://members.loria.fr/nazim.fates/doc/lecture/rapportfinal-ArthurRousseau.pdf`.

[5] A. Rousseau, « Mémoire bibliographique L3 maths-info, Étude de graphes pour la compensation mutuelle de dettes, » 2021. adresse : `https://members.loria.fr/nazim.fates/doc/rapports/memoire-biblio-Rousseau-2021.pdf`.

[6] M. M. Güntzer, D. Jungnickel et M. Leclerc, « Efficient algorithms for the clearing of interbank payments, » *Elsevier*, 1998. adresse : `https://doi.org/10.1016/S0377-2217(97)00265-8`.

[7] M. Hristakeva, « Solving the 0-1 Knapsack Problem with Genetic Algorithms, » 2004. adresse : `https://api.semanticscholar.org/CorpusID:12659882`.

[8] D. Beauquier, J. Berstel et P. Chrétienne, *Éléments d'algorithmique*. 2005, p. 241-269. adresse : `https://www-igm.univ-mlv.fr/~berstel/Elements/Elements.pdf`.

Annexes

A – Résolution du problème de réduction partielle de dettes mutuelles en temps polynomial

Montrons que ce problème peut être résolu en temps polynomial.

Soient $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe, et $Q \in \mathbb{N}^*$.

Tout d'abord, pour tout $(u, v) \in \mathcal{V}^2$, on fusionne toutes les arêtes de u vers v pour n'avoir qu'au plus une arête. On peut se convaincre rapidement que cette étape ne modifie pas les solutions car il s'agit du problème partiel.

Alors, on peut utiliser l'algorithme de Ford-Fulkerson sur ce graphe sur chaque nœud successivement (considéré ici comme source et puits), ce qui s'effectue en un temps polynomial [8]. On répète alors l'exécution de cet algorithme jusqu'à obtenir un graphe acyclique.

Ensuite, on crée deux sommets fictifs, source et puits, et l'on ajoute les arêtes suivantes : on ajoute une arête partant de source jusqu'à tous les nœuds n'ayant aucune arête entrante, et une arête partant de chaque nœud n'ayant aucune arête sortante jusqu'à puits. Toutes les arêtes créées ont un poids égal au financement maximal restant. On applique ensuite l'algorithme de Ford-Fulkerson sur ce graphe, ce qui s'effectue en temps polynomial.

On répète l'étape précédente tant que le financement ne dépasse pas Q. Cela s'effectue en $\mathcal{O}(Q)$, car à chaque étape où il reste des arêtes, on finance d'au-moins 1 la dette.

Or, comme pour la version intégrale du problème, avec un financement de $\phi(\mathcal{E})$, toute la dette est réduite et il ne reste plus aucune arête à financer. De plus on a facilement $\phi(\mathcal{E}) \leqslant \Sigma(\mathcal{E})$, et $\Sigma(\mathcal{E})$ est linéaire en m, donc cette répétition s'effectue en au plus $\mathcal{O}(m)$.

Cet algorithme résout donc le problème de réduction partielle en temps polynomial. □

B – NP-complétude du problème de réduction intégrale de dettes mutuelles

NP Montrons que ce problème est dans NP.

Soient $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe et $Q \in \mathbb{N}^*$.

Considérons le problème de décision suivant : étant donné $k \in \mathbb{N}^*$, existe-t-il $S \subseteq \mathcal{E}$ tel que $\phi(S) \leqslant Q$ et $\Sigma(S) \geqslant k$?

Ce problème décisionnel est clairement dans NP : il suffit de choisir de manière non-déterministe $S \subseteq \mathcal{E}$, de calculer $\phi(S)$ et $\Sigma(S)$ en temps polynomial, puis de les comparer respectivement à Q et k.

Ensuite, comme les poids sur les arêtes sont indépendants de $|\mathcal{V}|$ et $|\mathcal{E}|$, on sait que $\Sigma(\mathcal{E})$ ne dépend que linéairement de m. On teste donc cette solution pour tout $k \in [\![0, \Sigma(\mathcal{E})]\!]$, et on note k_0 le plus grand entier tel qu'il existe $S \subseteq \mathcal{E}$ vérifiant $\phi(S) \leqslant Q$ et $\Sigma(S) \geqslant k_0$. De plus, k_0 est maximal, on sait donc qu'il existe $S \subseteq \mathcal{E}$ vérifiant $\phi(S) \leqslant Q$ et $\Sigma(S) = k_0$.

On cherche donc un sous-ensemble d'arêtes de poids total k_0 : on choisit un $S \subseteq \mathcal{E}$ de manière non déterministe, et on le renvoie si $\Sigma(S) = k_0$, ce qui se calcule en temps polynomial.

Donc ce problème est dans NP. □

NP-difficile Montrons que ce problème est NP-difficile en faisant une réduction depuis le problème du sac à dos.

Soient I un ensemble d'objets et $K \in \mathbb{N}^*$. On note m_i la masse de l'objet i, avec $\forall i \in I$, $m_i \in \mathbb{N}^*$. Tous les objets ont ici une masse distincte des autres.

On pose par ailleurs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ avec : $\mathcal{V} = \{s, t\}$ et $\mathcal{E} = \{(s, t, m_i)_{i \in I}\}$, et $Q = K$.

Résoudre le problème du sac à dos revient à trouver $J \subseteq I$ tel que $\sum_{j \in J} m_j \leqslant K$ et $\sum_{j \in J} m_j$ soit maximale parmi les sous-ensembles J de I vérifiant $\sum_{j \in J} m_j \leqslant K$, ce qui est équivalent à prendre un sous-ensemble S de $\mathcal{E}$ tel que $\sum_{(s,t,m_j) \in S} m_j = \sum_{j \in J} m_j \leqslant K = Q$ et $\sum_{j \in J} m_j$ maximal.

Donc le problème du sac à dos se réduit à celui-ci, il est donc NP-difficile. □

Donc le problème de la réduction intégrale de dettes mutuelles est NP-complet. □

C – NP-complétude du calcul de potentiel

NP Montrons que ce problème est dans NP.

Comme v est le seul sommet avec une position nette négative, on a $\phi(S) = -\pi_S(v)$.

On raisonne alors de la même manière que dans l'annexe B : on considère le problème décisionnel associé qui est clairement dans NP, et on le résout pour tout $k \in [\![0, \Sigma(\mathcal{E})]\!]$, ce qui se fait en temps polynomial puisque $\Sigma(\mathcal{E})$ ne dépend que linéairement de m. On obtient donc k_0 maximal tel qu'il existe $S \subseteq \mathcal{E}$ vérifiant les deux conditions, et tel que $\Sigma(S) = k_0$. On est ici encore ramené au problème SUBSET-SUM qui est NP-complet.

Donc ce problème est dans NP. □

NP-difficile La preuve est la même que dans l'annexe B.

Donc ce problème est bien NP-complet. □

D – Terminaison et correction de l'algorithme 1

Terminaison À chaque appel récursif, on peut prendre 0 ou 1 arête :

— si aucune arête n'est prise, l'algorithme s'arête ;

— si une arête est prise, alors elle ne l'avait pas été avant.

Donc le nombre d'arêtes qui n'ont pas été prises est à valeur dans $\mathbb{N}$ et décroît strictement à chaque appel récursif.

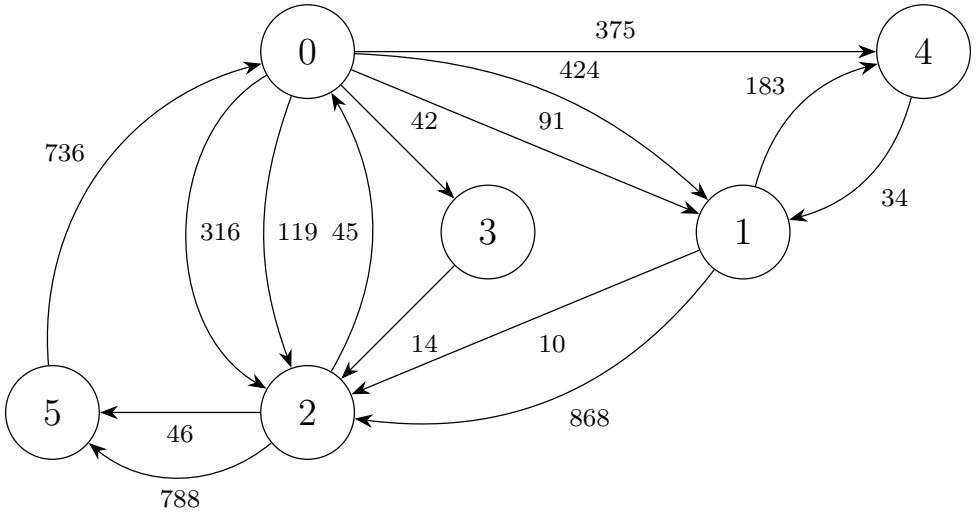
Donc l'algorithme 1 termine bien. □

Correction Initialement, une arête de poids au plus Q est prise, le financement actuel est donc d'au plus Q. Par la suite, à chaque appel récursif, les arêtes pouvant être choisies sont celles dont le poids est inférieur ou égal à celui de l'arête précédente, la position nette de chaque nœud sera donc positive, sauf pour le premier dont la position nette est au moins $-Q$. Donc, en notant S l'ensemble des arêtes renvoyées par l'algorithme, on a : $\phi(S) = -\pi_S(\text{sommet initial}) \leqslant Q$.

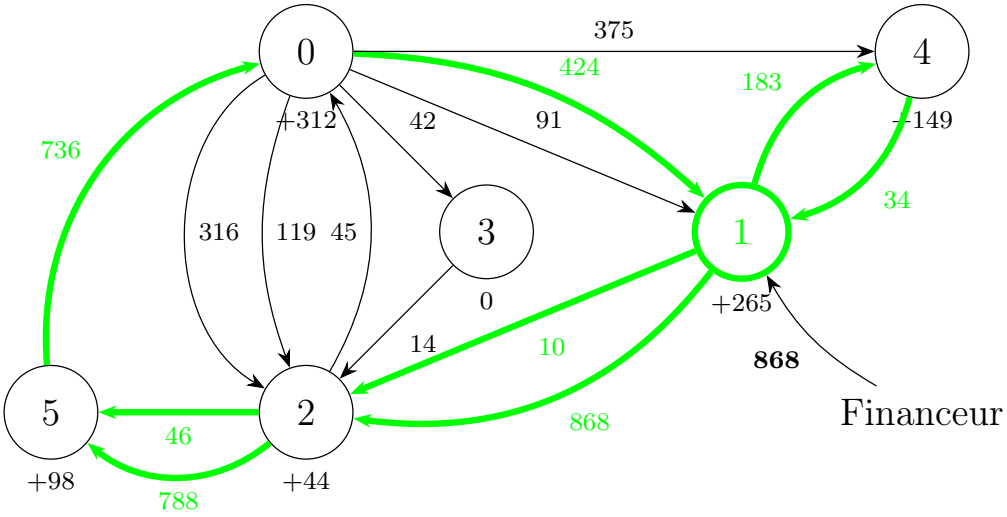
L'algorithme est donc correct. □

E – Exemple d'exécution pas à pas de l'algorithme POT

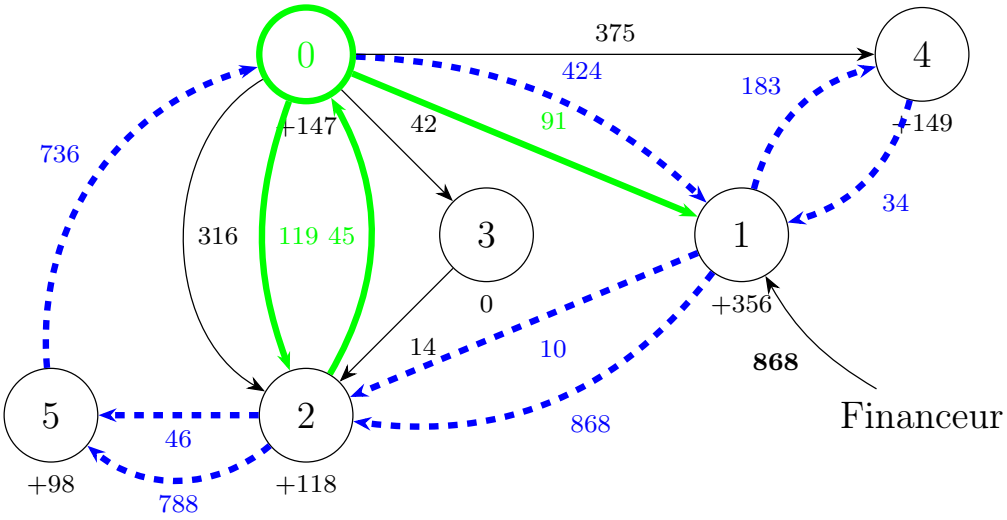
Chaque schéma correspond à une itération, donc à nouvelle exécution de l'algorithme 1. Les nouvelles arêtes sont marquées en vert et en gras, et celles déjà prises en bleu avec des lignes en pointillés.



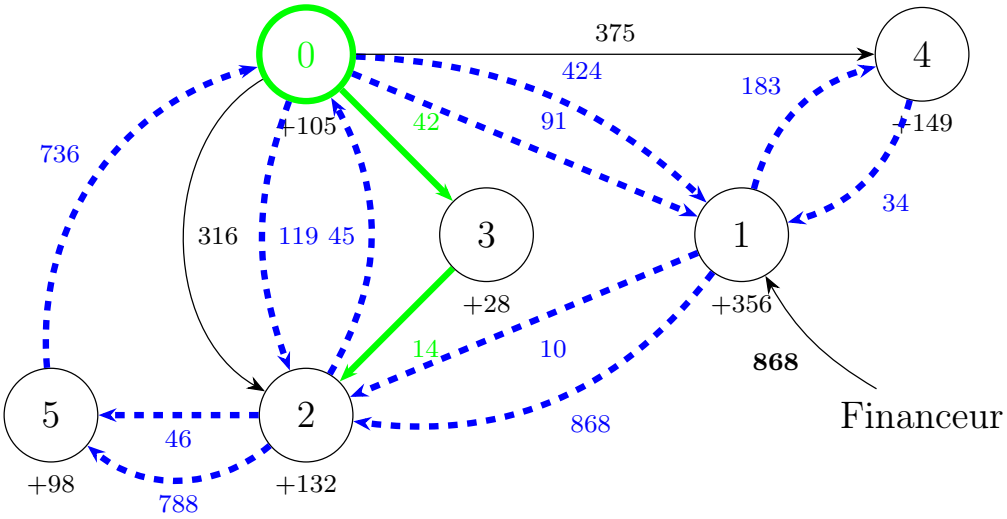
On finance le nœud 1 à hauteur de son arête avec le plus gros poids, à savoir 868, puis on applique l'algorithme du chemin glouton 1.



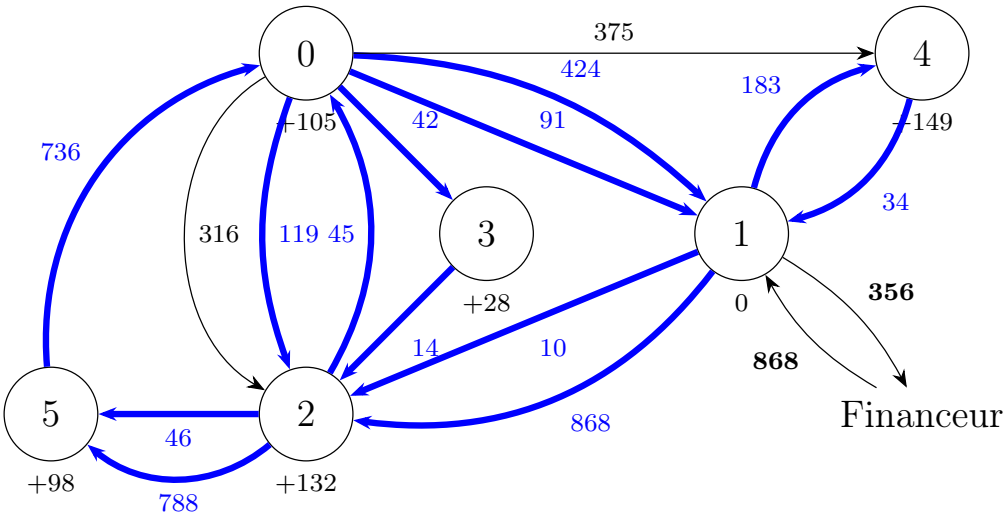
Maintenant, on continue d'appliquer l'algorithme 1 sur le sommet 0 qui a la plus grande position nette dans la solution.



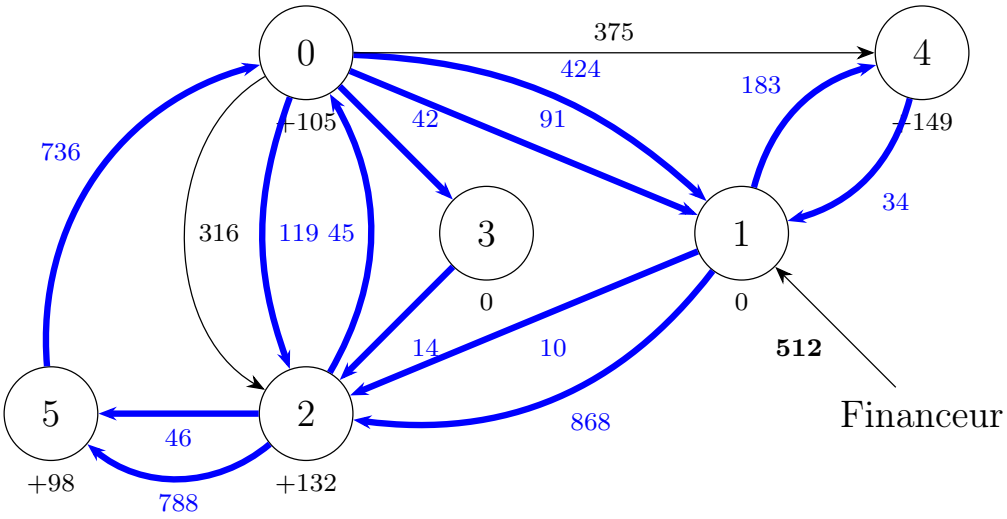
Les sommets 1 et 4 n'ont plus d'arête sortante, on réitère donc l'algorithme 1 en partant encore une fois du sommet 0.



Il ne reste plus aucune arête pouvant être financée sans apport supplémentaire, l'algorithme est donc terminé. Par ailleurs, on considère que le sommet financé, ici 1, rembourse le financeur autant qu'il le peut à la fin de l'algorithme.

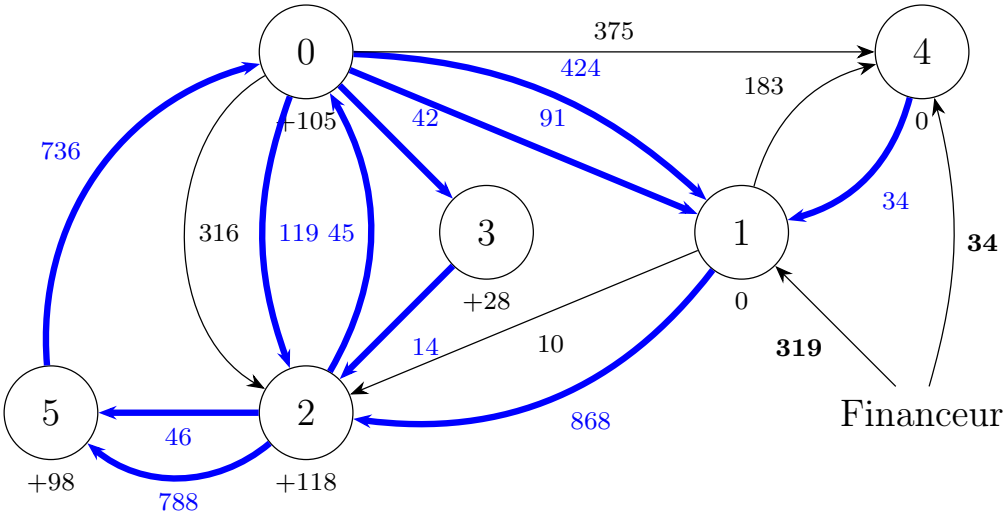


On peut alors simplifier le financement en la différence entre le financement initial et la somme remboursée à posteriori.

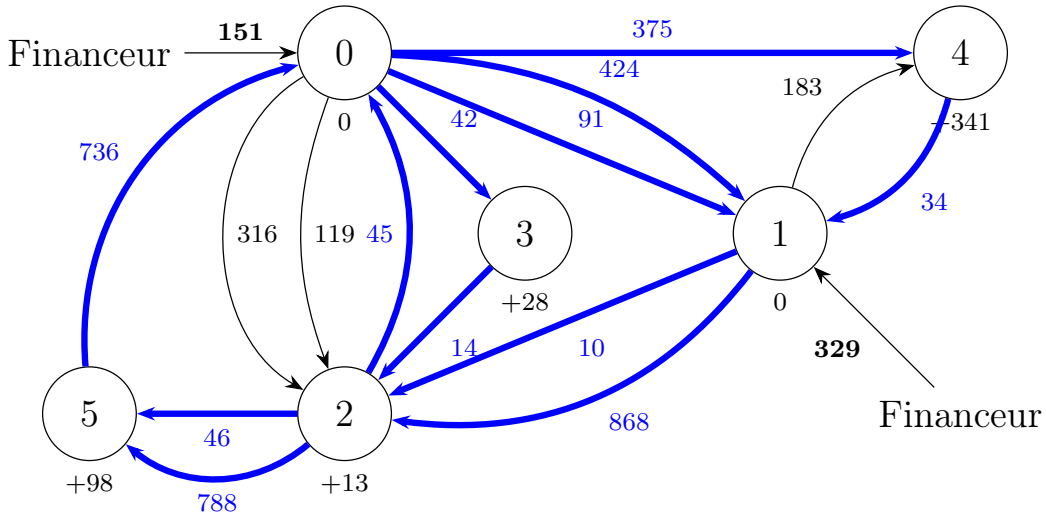


F – Solutions optimales de l'exemple 3 pour un financement maximal inférieur ou égal à 512

Solution optimale pour l'amplification avec un financement inférieur ou égal à 512

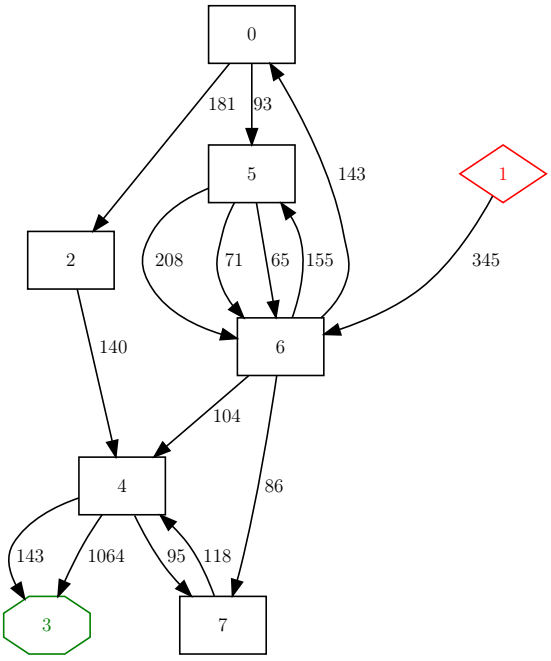


Solution optimale pour l'inclusion avec un financement inférieur ou égal à 512

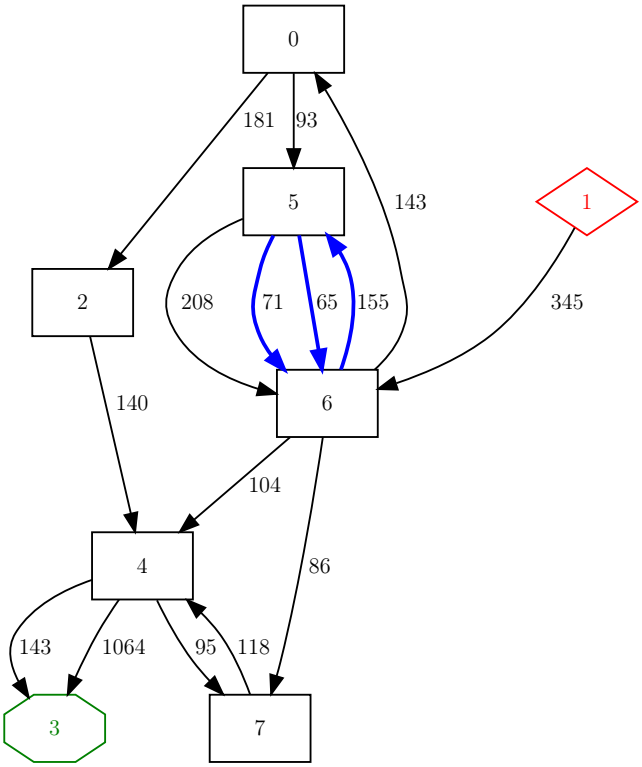


G – Exemple d'exécution pas à pas de l'algorithme 2

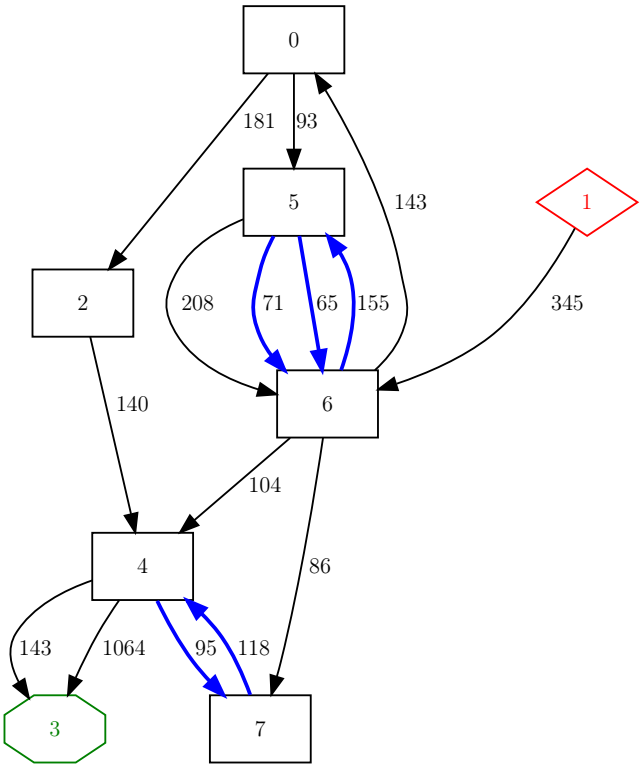
Graphe initial



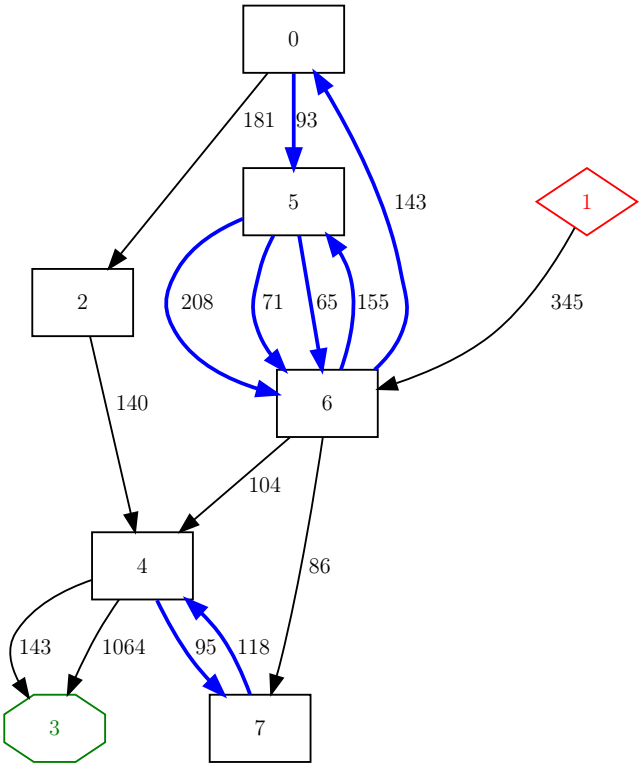
Financement du nœud 6 à hauteur de 155



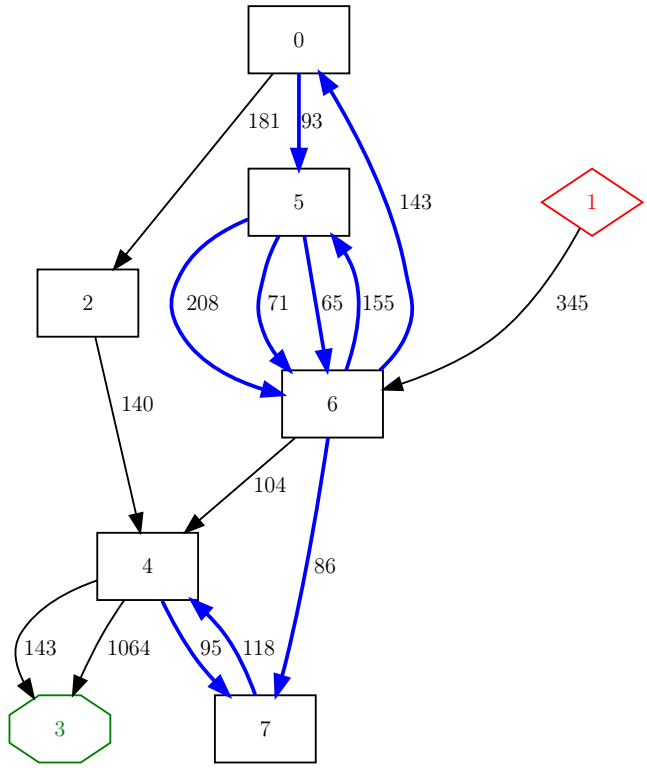
Financement du nœud 7 à hauteur de 118



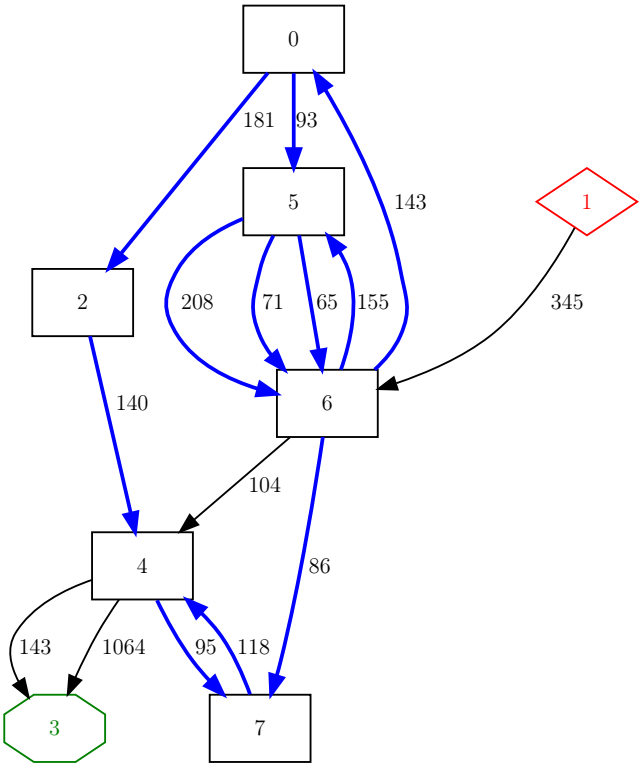
Financement du nœud 5 à hauteur de 208



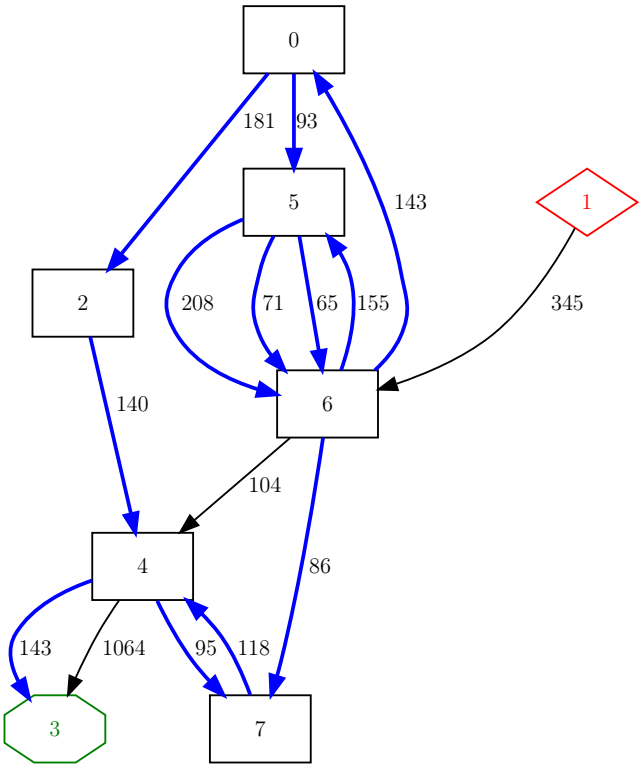
Financement du nœud 6 à hauteur de 86



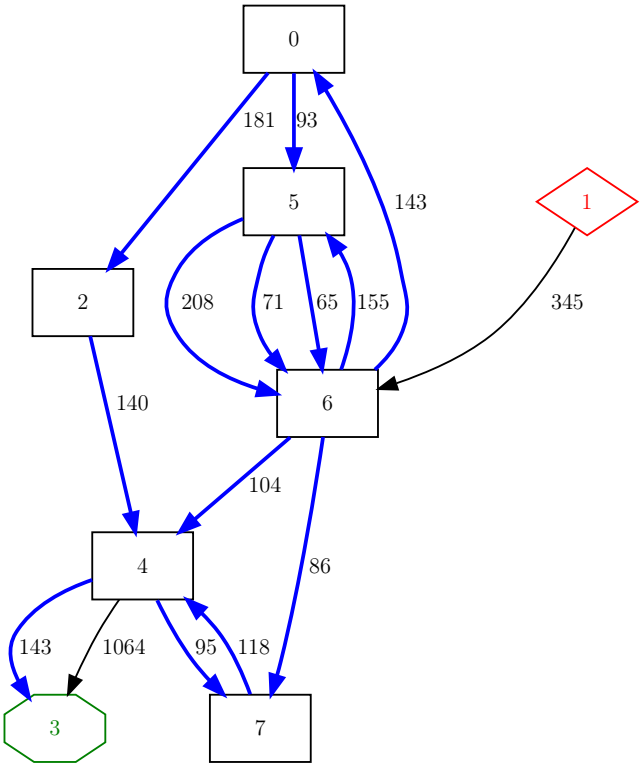
Financement du nœud 0 à hauteur de 181



Financement du nœud 4 à hauteur de 143



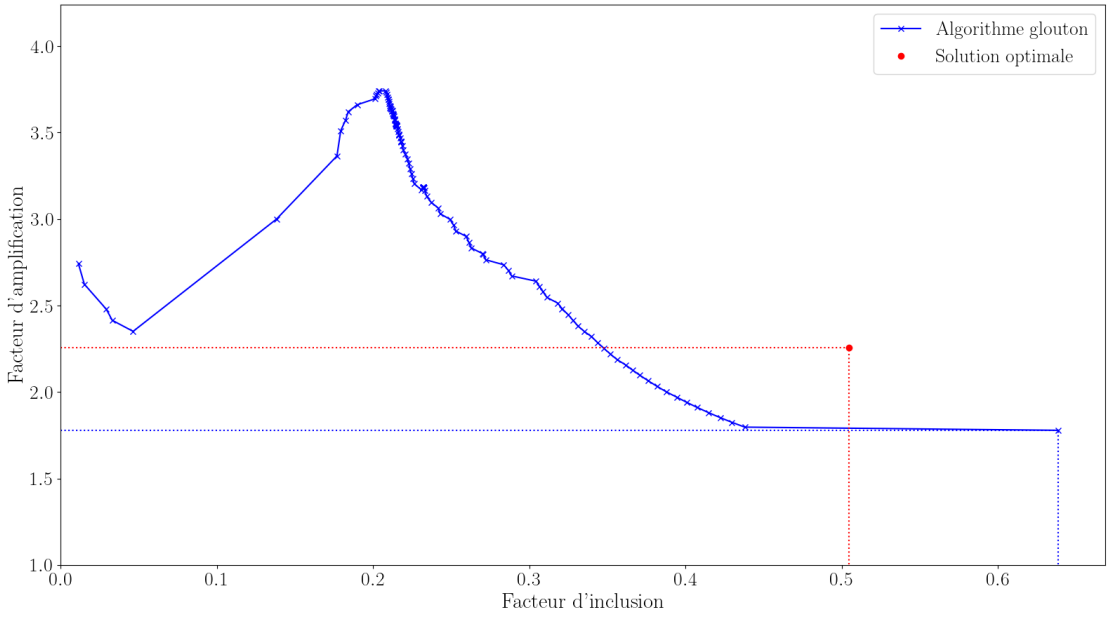
Financement du nœud 6 à hauteur de 104



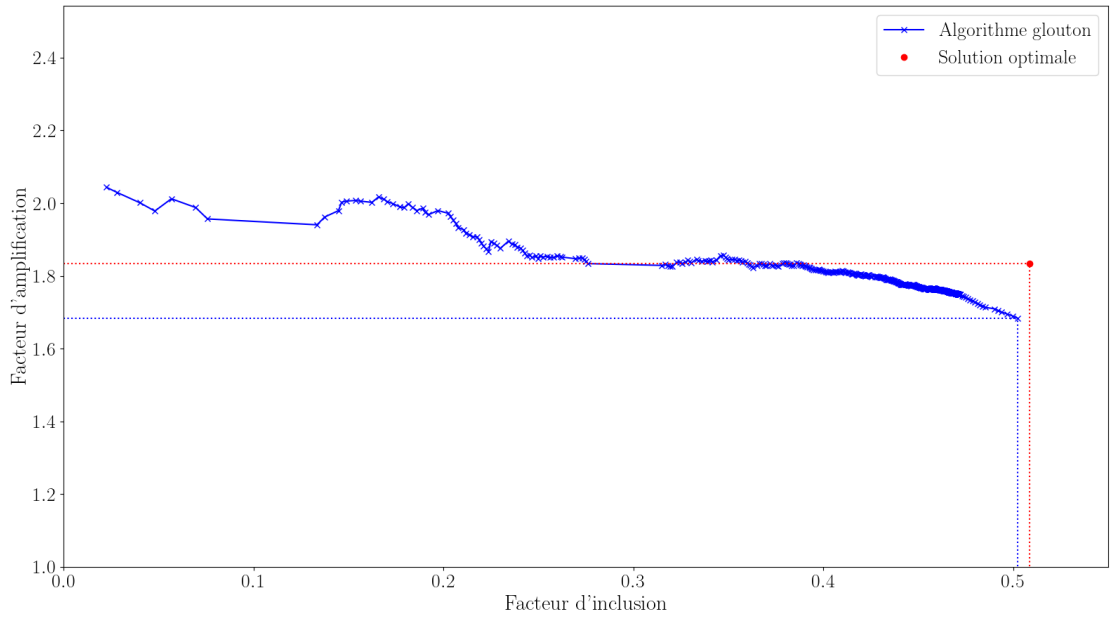
H – Résultats complets des tests de l'algorithme glouton sur des graphes générés aléatoirement

Pour chaque graphe, on indique les nombres de sommets et d'arêtes du graphe, ainsi que le nombre d'arêtes testées à chaque itération lors de l'exécution de l'algorithme 2 (voir 3.2.1).

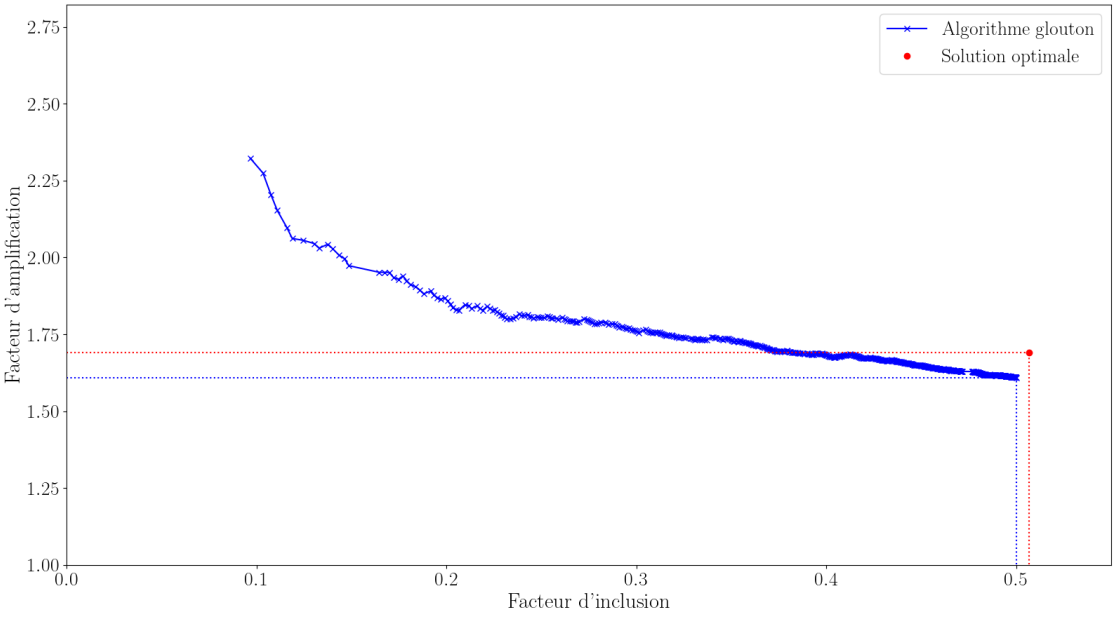
Résultat pour un graphe de 96 sommets et 333 arêtes avec un test de 100 arêtes



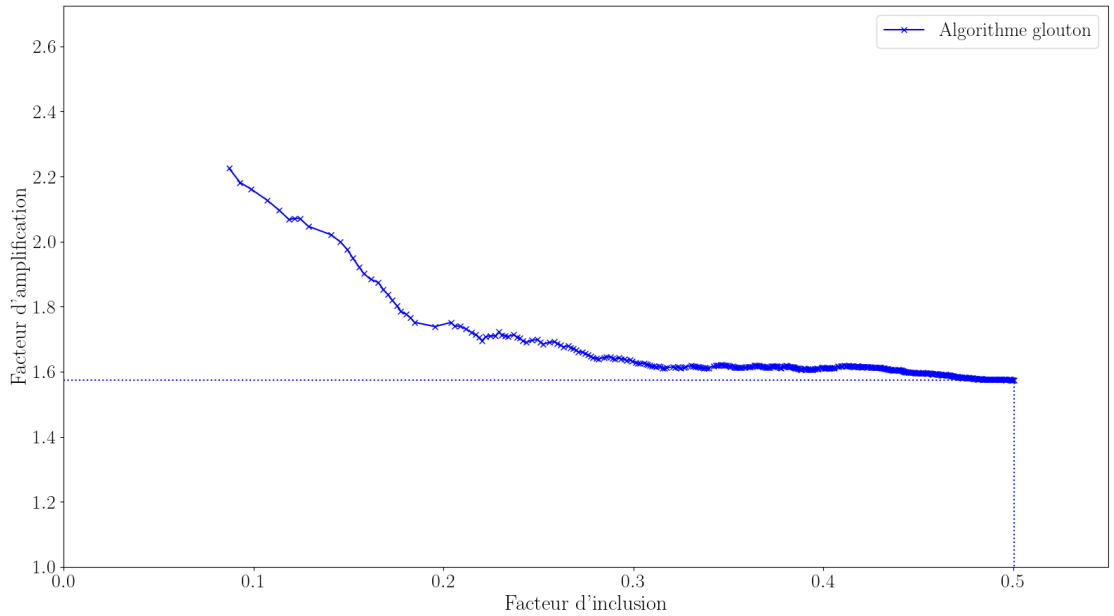
Résultat pour un graphe de 447 sommets et 1 666 arêtes avec un test de 50 arêtes



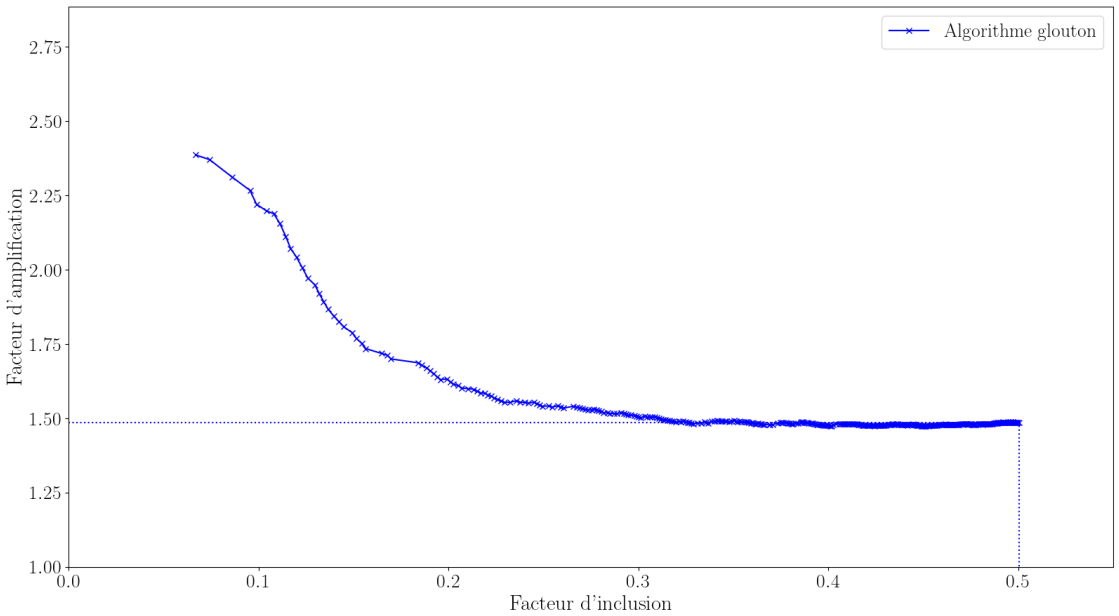
Résultat pour un graphe de 868 sommets et 3 325 arêtes avec un test de 50 arêtes



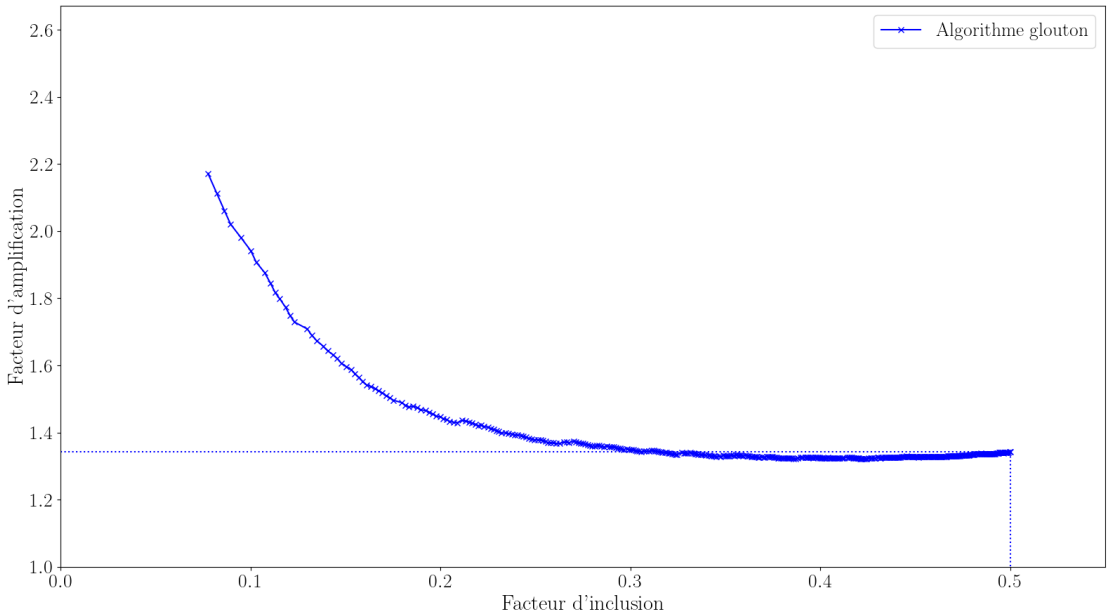
Résultat pour un graphe de 1 689 sommets et 6 666 arêtes avec un test de 50 arêtes



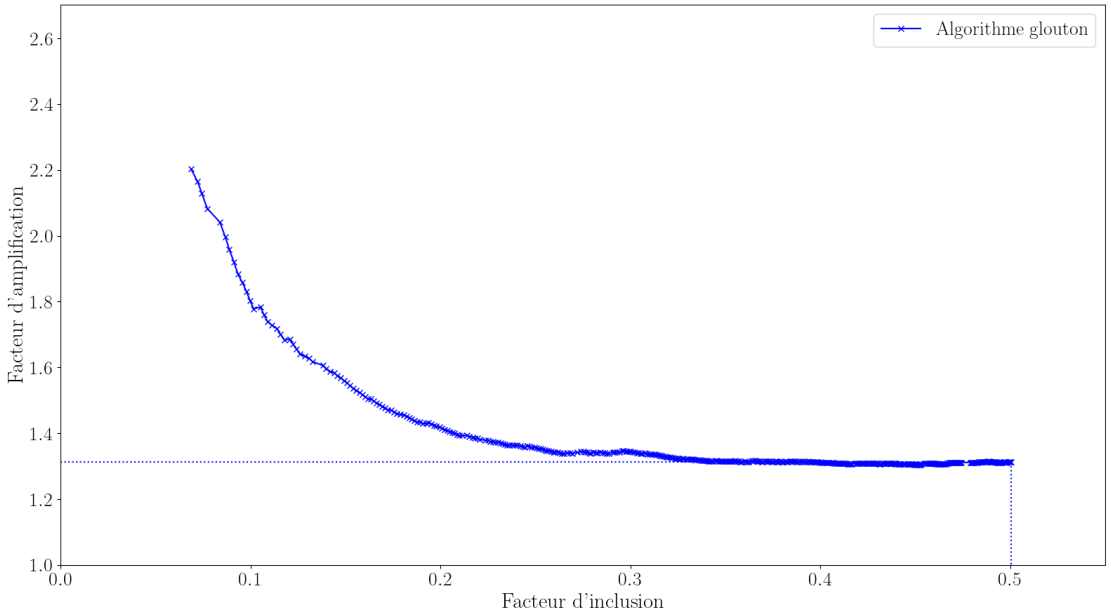
Résultat pour un graphe de 2 505 sommets et 10 032 arêtes avec un test de 50 arêtes



Résultat pour un graphe de 4 137 sommets et 16 680 arêtes avec un test de 50 arêtes



Résultat pour un graphe de 8 272 sommets et 33 239 arêtes avec un test de 25 arêtes



I – Méthode d'obtention des solutions optimales

Pour obtenir la solution exacte au problème de réduction intégrale de dettes mutuelles, la première méthode utilisée était l'énumération de tous les sous-ensembles d'arêtes du graphe initial. Cette méthode était naïve et permettait d'obtenir des solutions optimales en un temps raisonnable pour des graphes allant jusqu'à 30 arêtes.

La seconde méthode utilisée a été développée par un autre chercheur du laboratoire pendant mon stage : il s'agit de linéariser le problème puis de le résoudre avec un solveur optimisé, en l'occurrence

CPLEX, optimiseur développé par IBM. [1]

Plus précisément, soient $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe, $n = |\mathcal{V}|$ et $m = |\mathcal{E}|$. Pour toute arête $e \in \mathcal{E}$, on note w_e son poids. On crée $n + m$ variables de telle manière :

— pour chaque nœud $v \in \mathcal{V}$, on crée une variable x_v correspondant au financement apporté à v ;
— pour chaque arête $e \in \mathcal{E}$, on crée une variable y_e valant 1 si l'arête est prise dans la solution, et 0 sinon.

On cherche alors à trouver une solution maximisant $\sum_{e \in \mathcal{E}} w_e y_e - \sum_{v \in \mathcal{V}} x_v$ et telle que $\sum_{e \in \mathcal{E}} w_e y_e \geqslant \frac{1}{2} \sum_{e \in \mathcal{E}} w_e$.

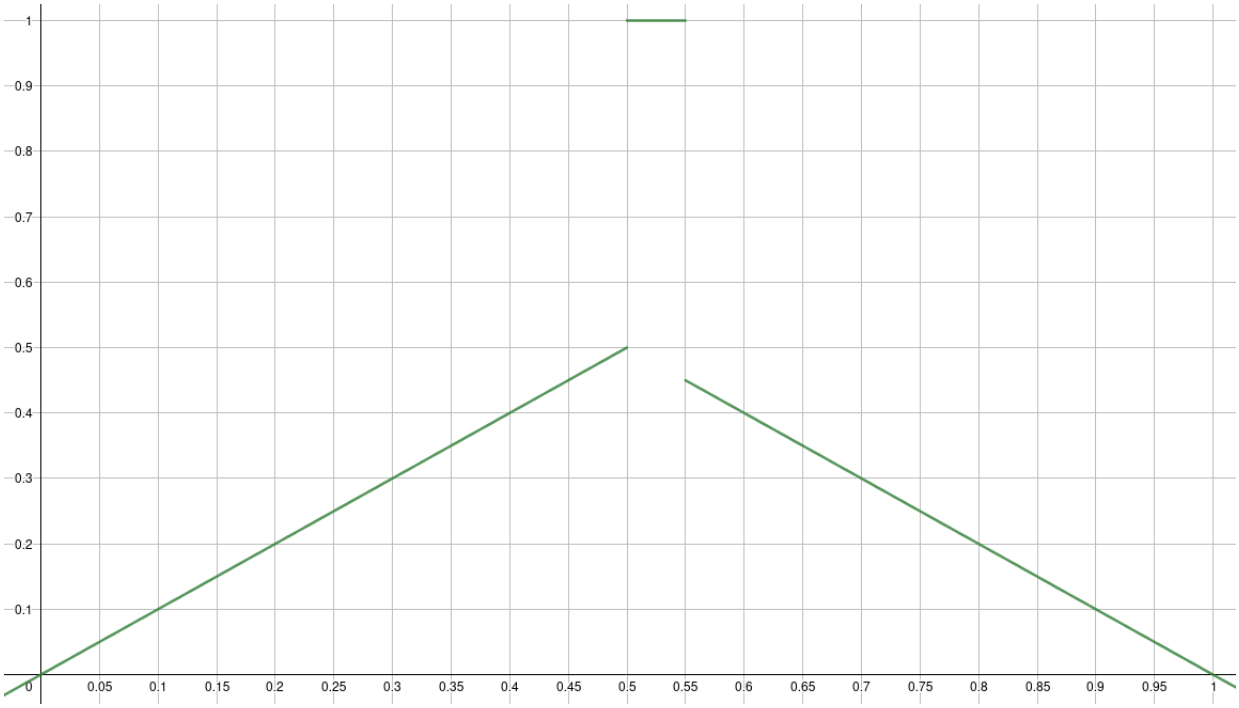
J – Représentation graphique de la fonction de score de l'algorithme génétique

On définit : $f : [0;1] \longrightarrow [0;1]$

$$\iota(S) \longmapsto \begin{cases} 1 & \text{si } \iota(S) \in [0.50, 0.55] \\ \frac{1}{2} \times (1 - 2\,|\iota(S) - 0.5|) & \text{sinon} \end{cases}$$

.

Voici la représentation graphique de cette fonction :



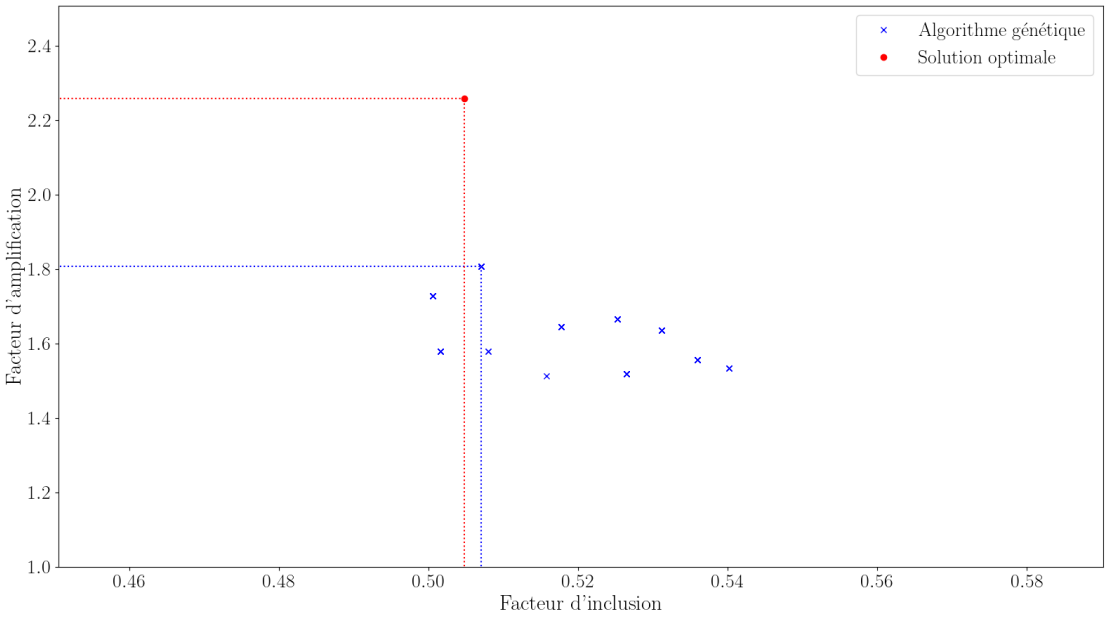
On multiplie le résultat par la suite par $\alpha(S)$ pour obtenir la fonction de score complète de l'algorithme génétique.

K – Résultats complets des tests de l'algorithme génétique sur des graphes générés aléatoirement

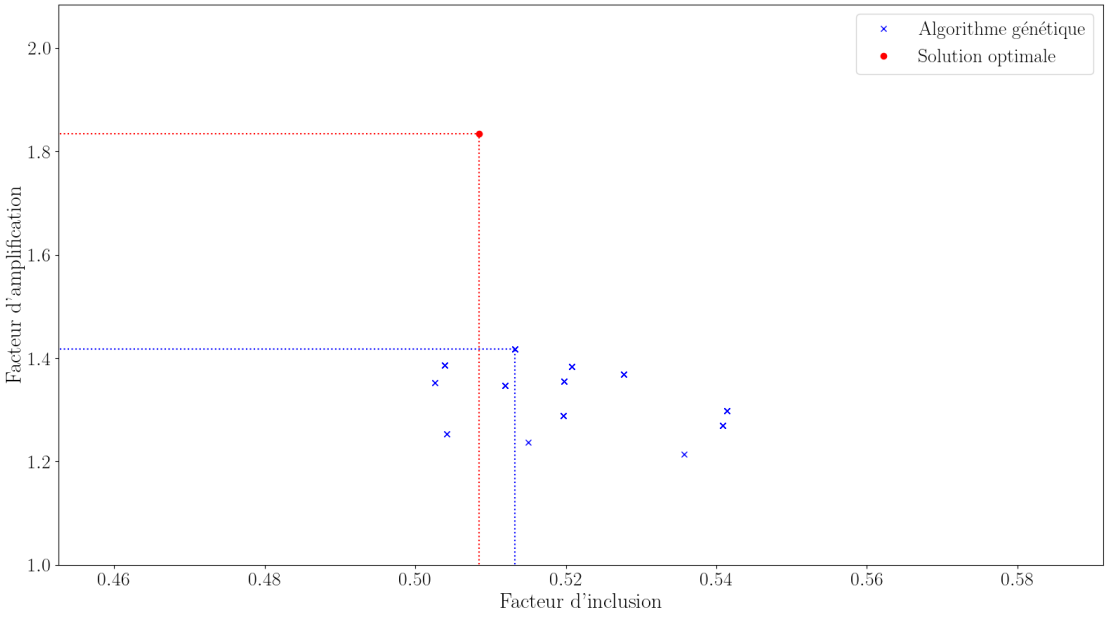
Pour chaque graphe, on indique les nombres de sommets et d'arêtes du graphe, ainsi que le temps d'exécution fixé à l'avance.

1. URL vers la page officielle d'IBM : `https://www.ibm.com/analytics/cplex-optimizer`

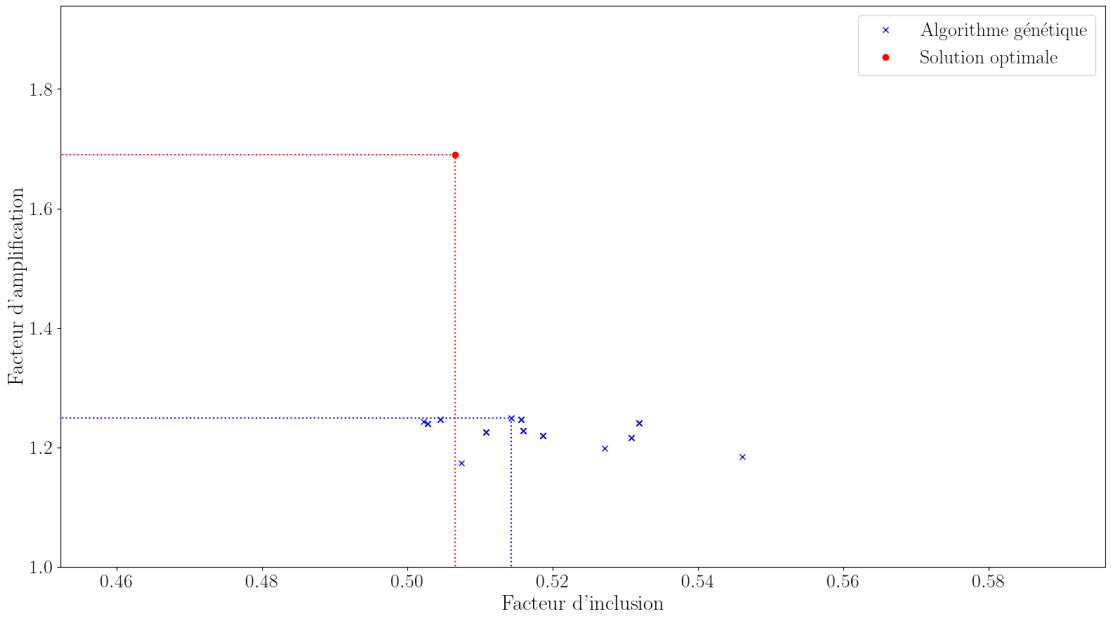
Résultat pour un graphe de 96 sommets et 333 arêtes en 300 secondes



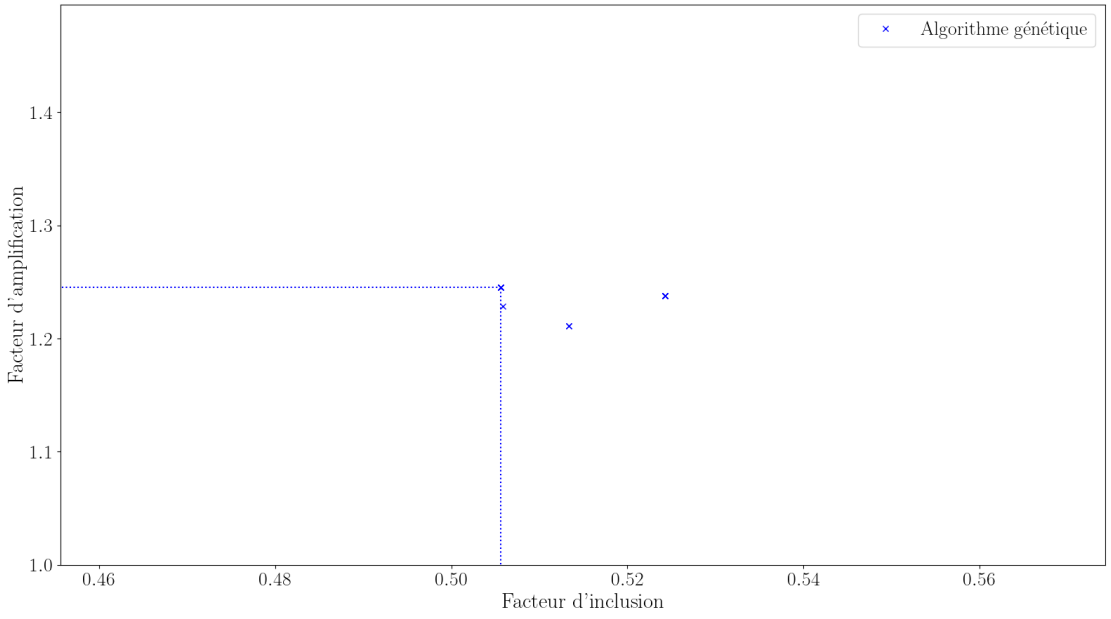
Résultat pour un graphe de 447 sommets et 1 666 arêtes en 300 secondes



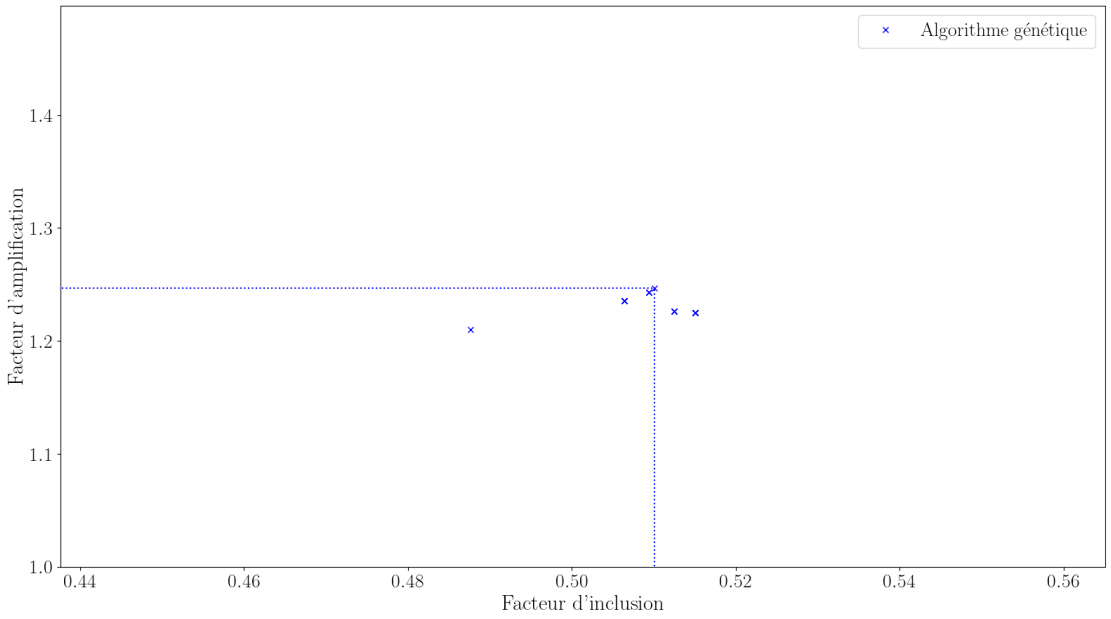
Résultat pour un graphe de 868 sommets et 3 325 arêtes en 300 secondes



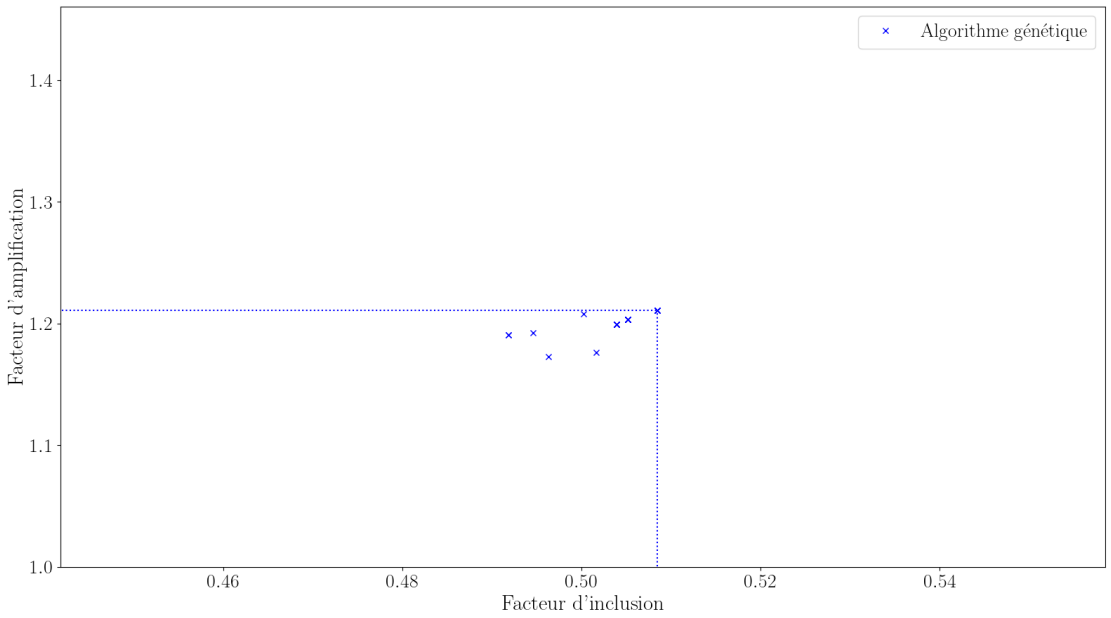
Résultat pour un graphe de 1 689 sommets et 6 666 arêtes en 300 secondes



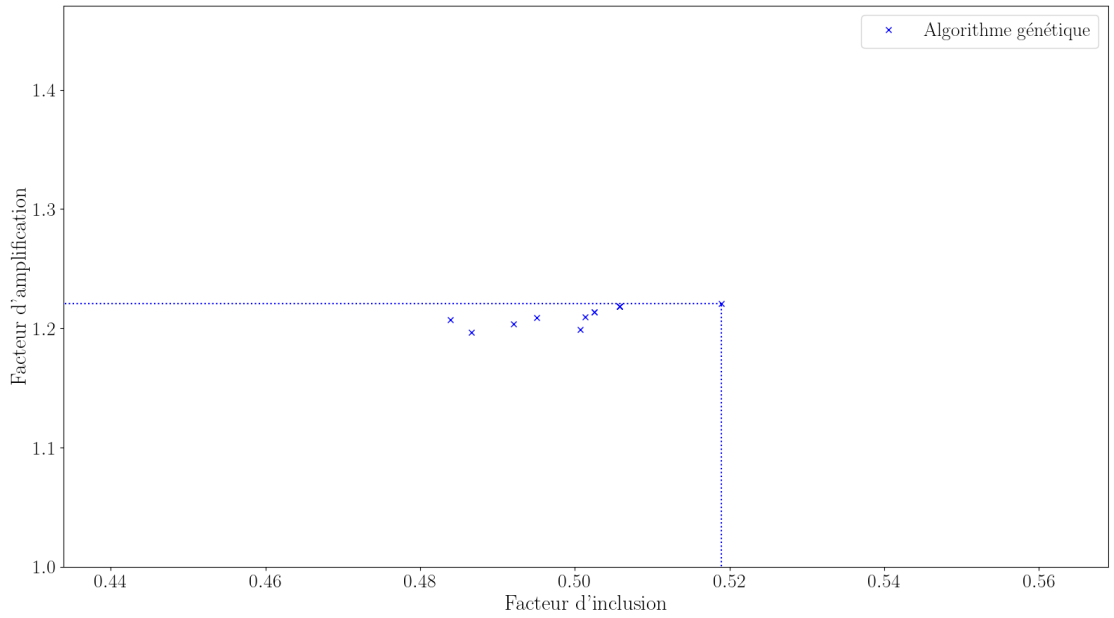
Résultat pour un graphe de 2 505 sommets et 10 032 arêtes en 600 secondes



Résultat pour un graphe de 4 137 sommets et 16 680 arêtes en 1 200 secondes



Résultat pour un graphe de 8 272 sommets et 33 239 arêtes en 1 200 secondes

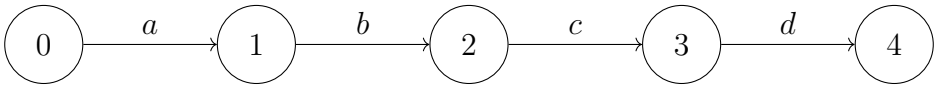


L – Équivalence entre certaines solutions partielles et totales

Définition 10: Graphe chemin

Soit $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe orienté pondéré.
On dit que $\mathcal{G}$ est un *graphe chemin* si tous les nœuds de $\mathcal{G}$ ont un degré sortant d'au-plus 1.

Exemple 5 Exemple de graphe chemin

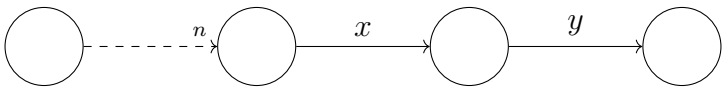


Propriété 9: Équivalence du remboursement complet entre les solutions partielles et totales dans les graphes chemins

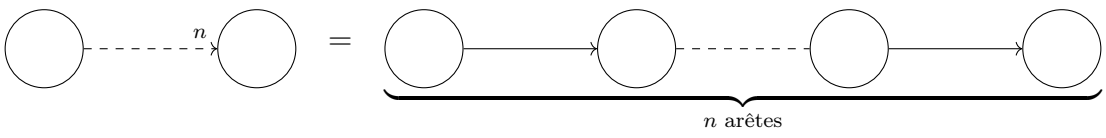
Soit $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un multigraphe orienté pondéré.
Si $\mathcal{G}$ est un graphe chemin, alors le financement minimal permettant d'éliminer la totalité des dettes de $\mathcal{G}$ est le même pour les problèmes de réduction intégrale et partielle de dettes mutuelles sur ce graphe.

Preuve On fait ici la supposition qu'aucun nœud est isolé, c'est-à-dire qu'il ne possède ni arête entrante ni arête sortante, car cela ne change en rien le raisonnement mais simplifie la preuve.

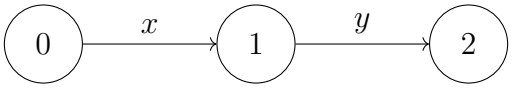
On raisonne par récurrence sur le nombre de sommets du graphe chemin de la forme :



Remarque : les arêtes en pointillés avec un exposant, ici n, signifient qu'il y a n arêtes successives. On a donc :



Initialisation Soient $x, y \in \mathbb{N}^*$. On considère le graphe :



Résolvons d'abord la version intégrale :

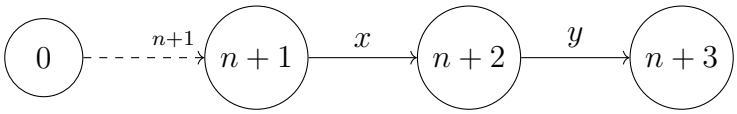
— si $x \geqslant y$, alors il faut financer le nœud 0 de x, et la dette y sera financée par le remboursement de la dette x ;

— si $y > x$, alors il faut financer le nœud 0 de x, et le nœud 1 de $y - x$ pour compléter le financement de la dette y.

Or, ce raisonnement est exactement le même pour la version partielle du problème, donc la propriété est vraie pour $n = 0$.

Hérédité Soit $n \in \mathbb{N}$ tel que la propriété est vérifiée pour n.

Soient $x, y \in \mathbb{N}^*$. On considère le graphe suivant :



On applique l'hypothèse de récurrence au sous-graphe comprenant les nœuds $0, 1, \ldots, n + 2$. On peut donc financer l'intégralité des dettes de ce sous-graphe avec le même financement minimal que ce soit pour le problème de réduction intégrale ou de réduction partielle de dettes mutuelles.

On retourne dans le graphe complet :

— si $x \geqslant y$, alors sans ajouter de financement supplémentaire, on peut directement financer la dette y grâce à la dette x qui est déjà éliminée ;

— si $x < y$, alors que ce soit dans la version intégrale ou dans la version partielle du problème, il faut financer le nœud $n + 2$ à hauteur de $y - x$ pour régler la dette y.

Dans tous les cas, le financement minimal nécessaire pour annuler l'ensemble des dettes de $\mathcal{G}$ est le même pour les versions partielles et intégrales du problème de réduction de dettes mutuelles. La propriété est donc vraie pour $n + 1$.

Donc d'après le principe de récurrence, la propriété est vraie pour tout n.

Remarque Il a été conjecturé, mais non démontré formellement, que ce résultat s'étend à tout multigraphe acyclique.