Algorithmes *output-sensitifs* pour les problèmes d'énumération dans les graphes

Sarah Blind*

Cette thèse se concentre sur des algorithmes efficaces pour certains problèmes d'énumération dans les graphes. Les enjeux sont doubles et d'intérêt indépendant: énumérer des *propriétés sur des ensembles de sommets définissables localement* et énumérer les orientations *k-arc-connexes*. Nous analyserons l'efficacité des algorithmes d'une manière *output-sensitive* et porterons une attention particulière au *délai* entre deux solutions consécutives.

0.1 Énumeration: algorithmes et complexité

L'algorithmique est l'un des domaines dont l'importance a considérablement augmentée au cours des dernières décennies. Le principal problème de ce domaine est de trouver des algorithmes efficaces pour résoudre un problème donné. La mesure de l'efficacité dépend alors de la tâche de calcul que nous abordons. On peut distinguer plusieurs types de problèmes, dans lesquels les objectifs de calcul diffèrent.

- *Les problèmes de décision:* décider si une instance du problème a une solution ou non;
- *Les problèmes de recherche:* trouver une solution (s'il y en a une) et la renvoyer;
- *Les problèmes de comptage:* compter le nombre de solutions qu'une instance du problème a;
- *Les problèmes d'énumération:* produire toutes les solutions d'une instance.

Dans de nombreux domaines, les problèmes d'énumération sont les problèmes les plus naturels: la génération d'objets avec des propriétés spécifiées a des applications importantes en informatique (exploration de données, apprentissage automatique, intelligence artificielle) ainsi que dans d'autres sciences, en particulier la biologie. Étant donné que le nombre de solutions d'un problème peut être exponentiel, dans un premier temps, les limitations informatiques ont mis de côté le domaine de la théorie de l'énumération. La question de l'énumération de toutes les solutions d'un problème de recherche donné ne s'est posée que pour des objets combinatoires relativement simples. Progressivement, avec le développement technologique, la nature des objets considérés est devenue de plus en plus complexe. L'énumération a connu un regain d'intérêt et est devenu un domaine complet de l'algorithmique et de la théorie de la complexité.

Les mesures habituelles de la complexité sont plutôt inadéquates lorsque l'on se concentre sur l'énumération. L'une des raisons est le nombre potentiellement exponentiel d'objets à énumérer. Elle rend inappropriée l'approche traditionnelle *input-sensitive*, qui estime l'efficacité en reliant le temps total nécessaire pour générer toutes les solutions à la taille de l'entrée. En revanche, l'approche *output-sensitive* mesure la complexité en temps d'un algorithme d'énumération en tenant compte à la fois de la taille de l'entrée et de celle de la sortie. Un algorithme d'énumération dont la

*Université de Lorraine, LGIPM, F-57000 Metz, France.

durée totale d'exécution est bornée par un polynôme en fonction à la fois de la taille de l'entrée et de la taille de la sortie s'appelle un algorithme *output-polynomial*. Le *délai* entre les sorties successives est une mesure plus fine et plus pertinente de l'efficacité de l'énumération, qui mesure la régularité de la production des solutions. Il est naturel de viser à maîtriser ce délai maximum ou au moins la moyenne de tous les délais, appelé temps amorti. Comme pour les problèmes de décision, un délai acceptable est un délai polynomialement borné en la taille de l'entrée. On peut espérer obtenir des résultats plus fins (délai linéaire ou constant), par exemple en utilisant certaines étapes de prétraitement. Dans la théorie des bases de données, le problème de réponse aux requêtes, qui vise à produire toutes les réponses à une requête sans doublons, est un exemple de problème d'énumération qui peut être résolu avec un délai constant après un prétraitement linéaire. Un autre algorithme à délai constant est le code de Gray pour générer tous les vecteurs de bits de longueur donnée. Une analyse amortie garantit la performance moyenne de chaque opération dans le pire des cas. L'analyse amortie est utilisée pour analyser certains algorithmes qui effectuent une séquence d'opérations similaires. Une analyse du pire des cas, dans laquelle nous additionnons les temps dans le pire des cas parmi les opérations individuelles, peut être indûment pessimiste, car elle ignore des effets corrélés des opérations sur la structure des données. Au lieu de borner le coût de la séquence d'opérations en bornant séparément le coût réel de chaque opération, une analyse amortie fournit une borne sur le coût réel de la séquence entière. Un avantage de cette approche est que, bien que certaines opérations puissent être coûteuses, de nombreuses autres peuvent ne pas l'être du tout. En d'autres termes, de nombreuses opérations peuvent s'exécuter beaucoup plus rapidement que le temps d'estimation le plus défavorable.

De nombreuses stratégies utilisées pour l'énumération reposent sur l'espace des solutions. L'idée est de mettre en évidence une propriété ou une structure de cet ensemble de solutions, puis de l'exploiter pour énumérer les solutions plus efficacement. Dans de nombreux cas, l'espace des solutions est structuré à l'aide d'un graphe dont les sommets sont des solutions. L'énumération se résume alors à une recherche intelligente du graphe où le défi est d'éviter deux problèmes: explorer des zones de l'espace dans lesquelles il n'y a pas de solution et sortir des solutions déjà considérées. Pour répondre à ces contraintes, il existe deux types de structures différentes qui peuvent être exploitées, correspondant respectivement aux méthodes dites *Backtrack Search* et *Reverse Search*.

Un algorithme de *Backtrack Search* repose sur un arbre dans lequel seules les feuilles représentent les solutions. Chaque nœud correspond à un embryon de solution qui s'étend progressivement jusqu'à l'obtention de solutions complètes. Le calcul revient alors à parcourir un arbre construit de la racine aux feuilles. Chaque feuille visitée correspond potentiellement à une solution. Plutôt que de recommencer de zéro pour calculer chaque nouvelle solution, *backtraquer* pour explorer une nouvelle branche permet de gagner du temps. Cela correspond à une recherche en profondeur. L'intérêt de cette méthode est que, par construction, elle évite les répétitions. Un *Backtrack Search* est par exemple utilisée avec succès, après un prétraitement linéaire, pour la génération de chemins d'un graphe acyclique dirigé. En revanche, pour certains problèmes, il existe un risque d'entrer dans des branches qui ne contiennent pas de solution. Dans ce cas, on descend de plus en plus profondément alors que la feuille cible ne convient pas. Par conséquent, une recherche intelligente de l'espace de solution est nécessaire. Cela peut être fait à condition que nous disposions d'un oracle qui décider si une branche mérite d'être explorée ou non. En d'autres termes, nous devons pouvoir vérifier avant chaque jonction que la piste que nous nous apprêtons à suivre sera fructueuse. Nous appellerons un tel *Backtrack Search* intelligent *Flashlight Backtrack Search*.

Dans la méthode *Reverse Search*, introduite par Avis et Fukuda, on exploite un graphe dont les nœuds (et pas seulement les feuilles) représentent les solutions. Ce graphe est construit à la volée en suivant une stratégie basée sur une certaine proximité des solutions avec un objectif prédéfini.

En d'autres termes, chaque solution est connectée à d'autres considérées comme réalisant plus efficacement un certain objectif. Ce lien de voisinage fournit toutes les solutions avec une structure de graphe orienté que nous pouvons parcourir pour énumérer les solutions. Pour construire ce graphe à la volée, nous devons définir un oracle d'adjacence qui donne les voisins d'un sommet, ainsi qu'une procédure de recherche locale. Cette procédure de recherche locale est utilisée pour parcourir efficacement le graphe et fournit un arbre couvrant. Le délai de l'algorithme d'énumération obtenu par cette technique de *Reverse Search* est borné en fonction de la hauteur de cet arbre couvrant. Il s'avère que pour de nombreux problèmes, cette hauteur est polynomiale en la taille de l'entrée, et donc l'algorithme correspondant a alors un délai polynomial. C'est par exemple le cas pour l'énumération des sous-graphes induits connexes ou pour l'énumération des ordres topologiques d'un graphe. Un autre exemple d'algorithme de *Reverse Search* est celui proposé par Makino et Uno pour énumérer les cliques maximales.

Une autre technique pour trouver des algorithmes d'énumération efficaces consiste à utiliser une notion appropriée de réduction. Il s'agit d'une méthode très classique pour les problèmes de décision. L'idée est de réduire un problème donné à un problème bien connu pour lequel l'existence d'algorithmes efficaces est bien connue. Une réduction est un moyen de transformer un problème en un autre afin que le deuxième problème puisse être utilisé pour résoudre le premier. Les réductions de Karp ainsi que les réductions de Turing servent cet objectif pour les problèmes de décision. Par exemple, on montre que le problème de trouver une orientation d'un graphe avec un degré sortant prédéfini en chaque sommet se réduit à un problème de flot, un problème qui est connu pour être résoluble en temps polynomial. Pour les problèmes de comptage, les réductions d'un problème à un autre doivent préserver non seulement l'existence d'une solution mais aussi le nombre de solutions. Valiant a défini une telle réduction, appelée *réduction parcimonieuse*. Une réduction appropriée entre des problèmes d'énumération doit porter encore plus d'informations car elle doit préserver les ensembles de toutes les solutions (et pas seulement leur cardinalité).

Dans cette thèse, nous illustrerons chacune de ces différentes techniques d'énumération - *Bactrack Search*, *Reverse Search* et la réduction - à travers des problèmes de graphes.

0.2 Contributions de la thèse

0.2.1 Énumération efficace de propriétés sur les sommets d'un graphe

Un exemple célèbre de problème d'énumération en théorie des graphes est celui de l'énumération des ensembles minimaux dominants d'un graphe. Ce problème a été résolu avec un délai linéaire pour les classes de graphes d'intervalles et de permutations, par M.M.Kanté *et al.*. Une telle réduction semble possible lorsque la propriété de graphe considérée présente un caractère local. Dans cette thèse nous nous appuyons sur cette observation afin de proposer une méthode générale de recherche d'algorithmes à délai linéaire pour l'énumération de propriétés sur les sommets dans les graphes.

Nous considérons les graphes pour lesquels il existe un ordre naturel sur les sommets, à savoir les graphes ordonnés linéairement et les graphes ordonnés cycliquement. On dit qu'une propriété est localement définissable par rapport à un ordre linéaire ou cyclique sur les sommets du graphe, quand on peut décider si un sous-ensemble de sommets respecte la propriété en parcourant la liste ordonnée de sommets à travers une fenêtre coulissante de taille fixe et en vérifiant que chaque tuple analysé satisfait certaines contraintes. Notre résultat principal est qu'une telle propriété localement définissable peut être énumérée avec un délai linéaire après un prétraitement polynomial. Ce résultat est obtenu en réduisant le problème d'énumération des ensembles de sommets satisfaisant une telle propriété localement définissable à l'énumération des chemins dans un graphe acyclique dirigé (DAG). Dans le cas des graphes ordonnés linéairement, la réduction établit une bijection entre

les ensembles de sommets à énumérer et les chemins dans un DAG. Fait intéressant, dans le cas des graphes ordonnés cycliquement, nous utilisons une réduction de Turing, qui nécessite un nombre polynomial d'appels à la procédure qui énumère les chemins dans un DAG.

Nous illustrons l'intérêt de cette méthode en l'appliquant à des variantes du problème de l'ensemble dominant minimal, à savoir, le problème de l'ensemble dominant connexe minimal et le problème de l'ensemble irredondant maximal. Ces propriétés d'ensemble de sommets sont bien étudiées en théorie des graphes, en particulier en tant que problèmes d'optimisation. Trouver un ensemble dominant connexe minimal et trouver un ensemble irredondant maximal sont deux problèmes NP-difficiles.

Des algorithmes *input-sensitive* ont été proposés pour énumérer les ensembles dominants connexes minimaux et les ensembles irredondant maximaux sur des classes spécifiques de graphes. Boros et Makino ont prouvé qu'il n'est pas possible d'énumérer les ensembles irredondants maximaux avec un délai polynomial (et même un délai incrémentiel) à moins que P = NP. Ici, nous limitons l'étude de l'énumération des ensembles dominants (connexes) minimaux et des ensembles irredondants maximaux aux classes de graphes suivantes: les graphes d'intervalle et de permutation pour les graphes ordonnés linéairement et les graphes d'arc-circulaire et de permutation-circulaire pour les graphes ordonnés cycliquement. Ces classes de graphes ont de nombreuses applications et sont des classes dans lesquelles de nombreux problèmes NP-complets peuvent être résolus efficacement.

En appliquant notre méthode, nous prouvons que les ensembles dominants connexes minimaux peuvent être énumérés avec un délai linéaire sur les graphes d'intervalle et les graphes de permutation. En corollaire, nous obtenons que l'on peut calculer un ensemble dominant minimum connexe sur de tels graphes en temps polynomial. Nous prouvons également que les ensembles irredondants maximaux peuvent être énumérés avec un délai linéaire sur les graphes d'intervalle, les graphes de permutation, les graphes d'arc-circulaire et les graphes de permutation-circulaire. De plus sur ces classes de graphes, le calcul d'un ensemble irredondant maximum peut se faire en temps polynomial.

Les problème de l'ensemble minimal dominant et de l'irredondant maximal étant au cœur de ce chapitre, nous prendrons également le temps d'étudier la relation entre ces deux problèmes. D'un côté la littérature contient des études approfondies sur les variantes du problème de domination, mais de l'autre les résultats sur la relation entre l'ensemble de tous les ensembles minimaux dominants et l'ensemble de tous les ensembles irredondants maximaux sont moins abondants. Nous proposons donc une caractérisation, en termes de sous-graphes interdits, des graphes qui respectent pour chacun de leurs sous-graphes induits la propriété que l'ensemble des ensembles minimaux dominants et l'ensemble des ensembles irredondants maximaux correspondent.

0.2.2 Énumération efficace des orientations k-arc-connexes dans les graphes

Étant donné un graphe non dirigé G, nous considérons le problème d'énumérer ses orientations avec des propriétés données. L'ensemble de toutes les orientations de $G = (V, E)$ peut être identifié avec l'ensemble des vecteurs $\{0, 1\}^m$ et l'exploration de ce dernier peut se faire en utilisant un code de Gray avec un délai constant. Cela devient plus intéressant lors de l'énumération des orientations acyclique, c'est-à-dire celles qui n'ont pas de cycles dirigés, par exemple, Figure 1.

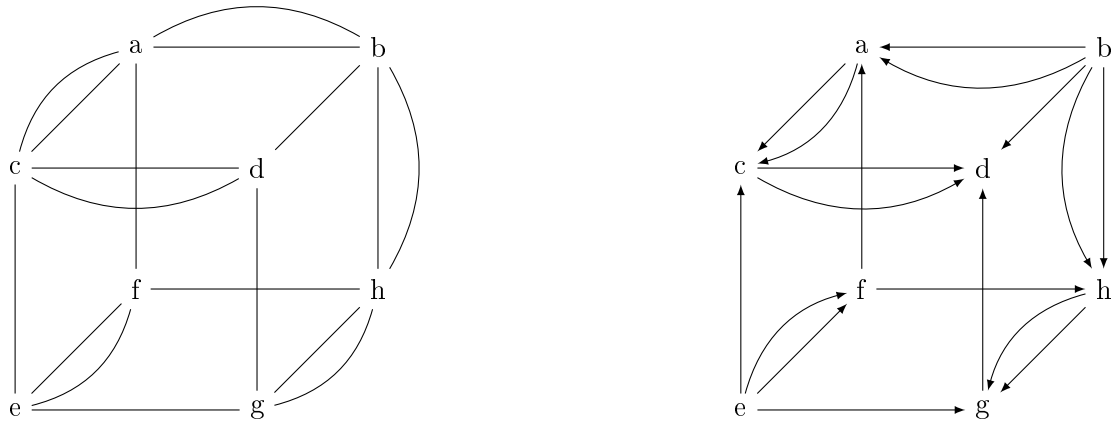

Figure 1: Un multigraph G et une orientation acyclique de G

Il existe un algorithme pour énumérer toutes les orientations acycliques avec un délai $O(n^3)$ et un temps amorti linéaire $O(n)$. Plus tard, le délai a pu être réduit à $O(mn)$ avec une augmentation du temps amorti à $O(m+n)$. Une autre amélioration a été obtenue avec un algorithme avec un délai de $O(m)$ et un temps amorti de $O(m)$. De nombreux autres types d'orientations ont été étudiés en fonction de leur complexité d'énumération. Un concept dual aux orientations acycliques est celui des orientations *fortement connexes* c'est-à-dire celles pour lesquelles il existe entre chaque paire de sommets $u, v \in V$ un chemin dirigé de u vers v.

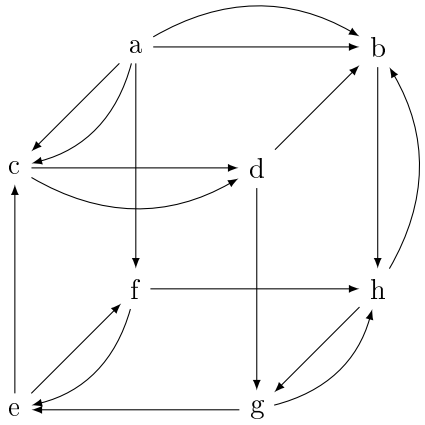

Figure 2: Une orientation qui n'est ni acylique ni fortement connexe

La Figure 2 est un exemple d'orientation qui contient un cycle (c, d, g, e) mais qui n'est pas fortement connexe: il n'y a aucun chemin pour aller du sommet b vers le sommet a. La Figure 3 est un exemple d'orientation fortement connexe. Il est possible d'énumérer les orientations fortement connexes avec un délai en $O(m)$.

Dans ce chapitre nous étudions des algorithmes efficaces pour l'énumération d'orientations qui sont une généralisation des orientations fortement connexes, il s'agit des *orientations k-arc-connexes*.

Definition 1. *Une orientation $D = (V, A)$ de $G = (V, E)$ est k-arc-connexe si pour tout $A' \subseteq A$ avec $|A'| < k$, le graphe non dirigé $D \setminus A'$ est fortement connexe.*

En d'autres termes, une orientation d'un graphe est k-arc-connexe si elle est fortement connexe et si la suppression d'au moins k arcs est nécessaire pour détruire la forte connexité.

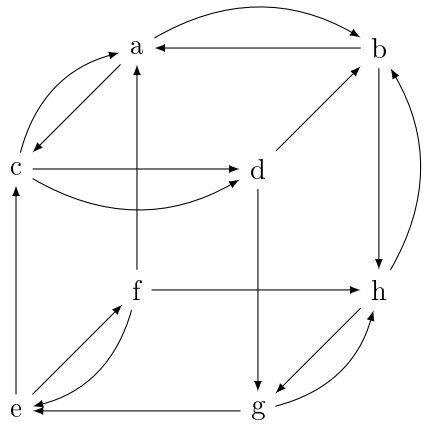

Figure 3: Orientation 1-arc-connexe de G

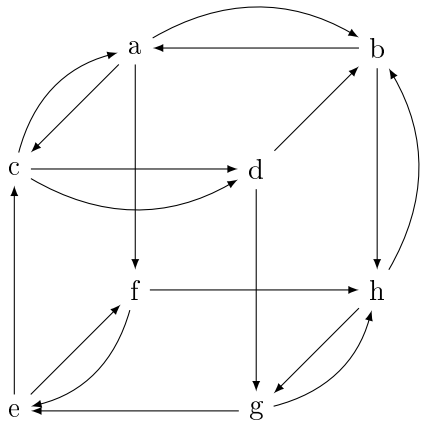

Figure 4: Orientation 2-arc-connexe de G

Les Figures 3 et 4 sont des orientations du graphe non dirigé G de la Figure 1. Les deux orientations sont fortement connexes mais, contraitement à la Figure 3, l'orientation de la Figure 4 est 2-arc-connected. En effet, malgré le fait que la seule différence entre les deux orientations soit la direction de l'arête (a, f), supprimer du graphe 3 l'arc (g, e) donne lieu à un graphe qui n'est plus fortement connexe (il n'est par exemple plus possible d'aller du sommet a au sommet e).

Dans ce chapitre nous nous intéresserons au problème d'énumération suivant:

ENUM_k-ARC-CONNECTED-ORIENTATIONS	
Input :	un graphe $G = (V, E)$
Output :	l'ensemble des orientations k-arc-connexes de G

Une contribution majeure dans le domaine des orientations de graphes sous contrainte de connexité est due à Robbins. Il a caractérisé les graphes qui admettent une orientation fortement connexe.

Definition 2. *Un graphe non dirigé $G = (V, E)$ est k-arête-connexe si pour tout $E' \subseteq E$ avec $|E'| < k$, le graphe $G \setminus E'$ est connexe.*

En d'autres termes, un graphe non dirigé est k-arête-connexe s'il est connexe et si la suppression d'au moins k arêtes est nécessaire pour détruire la connexité.

Theorem 3 (Robbins)**.** *Un graphe G admet une orientation fortement connexe si et seulement si G est 2-arête-connexe.*

Une generalisation de ce théorème est donnée par Nash-Williams pour les orientations k-arc-connexes. Son théorème est au cœur de la plupart des résultats utilisés dans ce chapitre.

Theorem 4 (Nash-William). *Un graphe $G = (V, E)$ admet une orientation k-arc-connexe D si et seulement si G est $2k$-arête-connexe.*

Frank apporte une contribution importante à la théorie des orientations k-arc-connexes. En particulier, il montre que deux orientations k-arc-connexes de G peuvent être transformées l'une en l'autre en inversant des chemins dirigés et des cycles dirigés. L'analyse de ce résultat est au cœur des algorithmes présentés dans ce chapitre. Une autre contribution importante de Frank est l'intégration d'orientations k-arc-connexes dans la théorie des flots sous-modulaires. En particulier, trouver un flot sous-modulaire (et donc une orientation k-arc-connexe) peut être fait en temps polynomial. Il existe une quantité considérable de littérature proposant différentes solutions algorithmiques pour cette tâche sur des graphes simples, multigraphes et graphes mixtes.

En utilisant un algorithme d'orientation k-arc-connexe de coût minimal, il est possible de construire un algorithme qui décide en temps $O(n^3k^3 + kn^2m)$ si un graphe mixte $G = (V, E \cup A)$ peut être étendu en une orientation qui est k-arc-connexe. Cela donne lieu à un algorithme pour énumérer les orientations k-arc-connexes avec un délai en $O(m(n^3k^3 + kn^2m))$. L'idée est tout simplement de commencer avec $G = (V, E)$ comme racine de l'arbre d'énumération et à chaque noeud de créer deux fils en choisissant une arête et en considérant ses deux orientations possible. Pour chacune des deux orientations possible de l'arête considérée, il s'agit de vérifier s'il existe une extension k-arc-connexe. Nous présenterons deux façons de résoudre ce problème d'extension.

La principale contribution de ce chapitre est une approche alternative pour résoudre le problème d'énumération des orientations k-arc-connexes. Nous considérons deux problèmes d'énumération d'intérêt indépendant, dont la combinaison donne lieu à un algorithme efficace pour énumérer les orientations k-arc-connexes. Le premier consiste à énumérer toutes les orientations de G avec une séquence de degrés sortants prescrite, de telles orientations sont connues sous le nom de *α-orientations*. Étant donné $\alpha : V \to \mathbb{N}$, une orientation D de G est une α-orientation si $\delta^+_D(v) = \alpha(v)$ pour tous $v \in V$, où $\delta^+_D(v)$ désigne le degré sortant de v. Le deuxième problème consiste en la génération de toutes les séquences de degrés sortants pour lesquelles il existe au moins une orientation k-arc-connexe. Pour chacun de ces problèmes, nous proposons et comparons deux méthodes de résolution. Les principaux avantages de cette approche sont la simplicité de l'algorithme final (alors que les algorithmes pour les flots sous-modulaires ont tendance à être très complexes) et que les deux parties de cet algorithme énumèrent des objets d'intérêt indépendant. L'idée derrière notre approche vient essentiellement du résultat déjà mentionné de Frank selon lequel les inversions de cycles dirigés et de chemins dirigés suffisent pour énumérer toutes les orientations k-arc-connexes.

Nous présenterons deux algorithmes pour l'énumération des α-orientations pour un certain α fixé. Le premier est dû à une réduction au problème d'énumération des sommets d'un polyèdre et le deuxième est un *Flashlight Backtrack Search* qui utilise des BFS. Le deuxième algorithme s'avère être le plus efficace avec un délai de $O(m^2)$.

Concernant le deuxième problème qui consiste à énumérer, pour un graphe donné, toutes les séquences de degrés sortants pour lesquelles il existe des orientations k-arc-connexes, nous proposons également deux manières différentes de le résoudre. La première est une illustration de la technique *Reverse Search* et la seconde plus efficace, est un *Flashlight Backtrack Search* avec un délai de $O(m^2kn)$.

Enfin, nous combinons les deux algorithmes les plus efficaces pour chaque problème, ce qui conduit à un algorithme d'énumération final pour les orientations k-arc-connexes de délai $O(m^2kn)$ et de temps amorti $O(m^2)$.