Un algorithme facteur 16 pour l'approximation d'une distance par une dissimilarité de Robinson

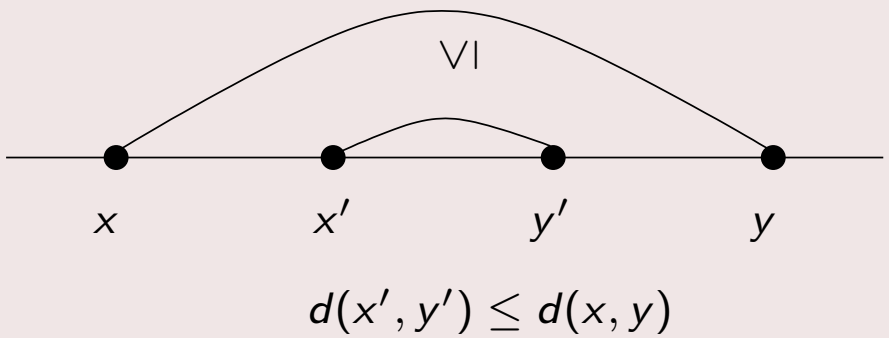
Victor Chepoi et Morgan Seston

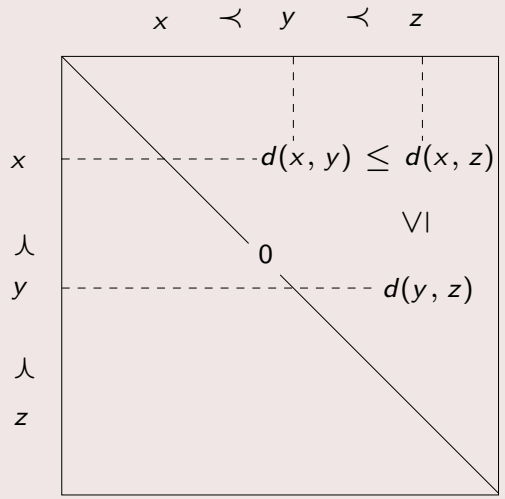
Laboratoire d'Informatique Fondamentale de Marseille

- Introduction
- Etat de l'art
- L'algorithme

Dissimilarités

- Une dissimilarité d définie sur X est une application de X^2 dans $\mathbb{R}^+$, symétrique et nulle sur la diagonale principale.
- Un ordre total $\prec$ est compatible avec d si $\forall\ x \prec y \prec z$, $d(x,z) \geq \max\{d(x,y), d(y,z)\}$.
- Une dissimilarité est de *Robinson*, ssi il existe un ordre total compatible avec d.





Problème de sériation

- Les dissimilarités de Robinson interviennent dans différentes disciplines comme la classification, la sériation ou la visualisation de matrices avec des champs d'application variés comme l'archéologie, l'écologie numérique, la biologie avec le séquençage d'ADN, etc.
- Approcher une distance par une distance d'un type donné est devenu un problème majeur de la géométrie des distances touchant différents domaines comme l'informatique théorique, les mathématiques discrètes, et l'analyse des données.

Problème d'approximation

Problème l_∞-FITTING-BY-ROBINSON : *Etant donnée une dissimilarité d, trouver une dissimilarité de Robinson d_R minimisant l'erreur l_∞ : $||d - d_R||_\infty = max_{x,y \in X}\{|d(x,y) - d_R(x,y)|\}$.*

- Un ordre total $\prec$ sur X est ϵ-*compatible* si $u \prec x \prec y \prec v$ implique $d(u,v) + 2\epsilon \geq d(x,y)$.
- Une dissimilarité d sur X est ϵ-*Robinsonienne* si elle admet un ordre ϵ-compatible.

Exemple

d	x	y	z	w
x	0	1	3	5
y		0	5	3
z			0	1
w				0

d_R	x	y	z	w
x	0	1	4	5
y		0	4	4
z			0	1
w				0

Problème d'approximation

Résultat principal

l_∞-FITTING-BY-ROBINSON est approximable avec un facteur 16.

Définition

Pour un problème de minimisation Π, *un algorithme d'approximation avec un facteur* $c \geq 1$ est un algorithme qui pour toute instance I de Π s'exécute en temps polynômial dans la taille de I et garantit que le coût de la solution obtenue est inférieur ou égal à c fois le coût de la solution optimale.

Travaux similaires

- Farach, Kannan, Warnow (1995) ont montré qu'approcher en norme l_∞ une dissimilarité par une ultramétrique est polynômial.
- Agarwala, Bafna, Farach, Paterson et Thorup (1999) donnent un algorithme facteur 3 pour l'approximation en norme l_∞ d'une distance par une distance d'arbre.
- Håstad, Ivansson, Lagergren (2003) donnent un algorithme facteur 2 pour l'approximation en norme l_∞ d'une distance par une distance unidimentionnelle.
- Si la norme d'erreur l_∞ est remplacé par l_1, les meilleurs algorithmes existants ont un facteur d'approximation $O(\log n)$ (Ailon et Charikar (2006), Dhamdhere (2004)).

Faux départ

Une heuristique similaire à ceux utilisés par (Håstad et al., 2003) et (Agarwala et al., 1999) pour leur algorithme facteur 3 (qui teste n ordres au lieu des $n!$ ordres possibles) ne fournit pas un algorithme à facteur constant dans notre cas.

Heuristique

- On teste tous les $x \in X$ comme point le plus à gauche.
- On construit l'ordre $\prec_x$ t.q. $y \prec_x z$ ssi $d(x, y) <_x d(x, z)$.
- On construit la dissimilarité optimale pour chaque $\prec_x$, et on retourne la meilleure.

Problème restreint

- *Etant donnée une dissimilarité d et un ordre total $\prec$, trouver une dissimilarité de Robinson d' compatible avec $\prec$ et minimisant l'erreur $||d - d'||_\infty$.*
- Ce problème peut être résolu en temps polynomial :

- Pour tout $x, y \in X$ avec $x \prec y, x \neq y$, soit

$$\check{d}_\prec(x, y) = \max\{d(u, v) : x \prec u \prec v \prec y\}.$$

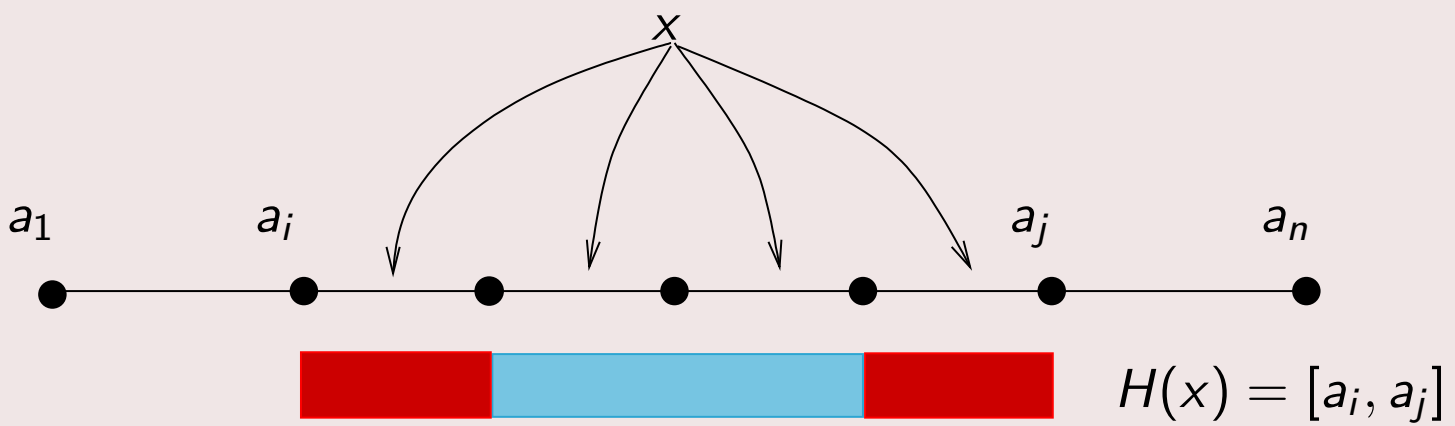
- Soit $2\tilde{\epsilon}_\prec = ||d - \check{d}_\prec||_\infty$.
- Alors $\tilde{d}_\prec = \check{d}_\prec - \tilde{\epsilon}_\prec d_0$, ou d_0 est la distance du graphe complèt.

Idée générale

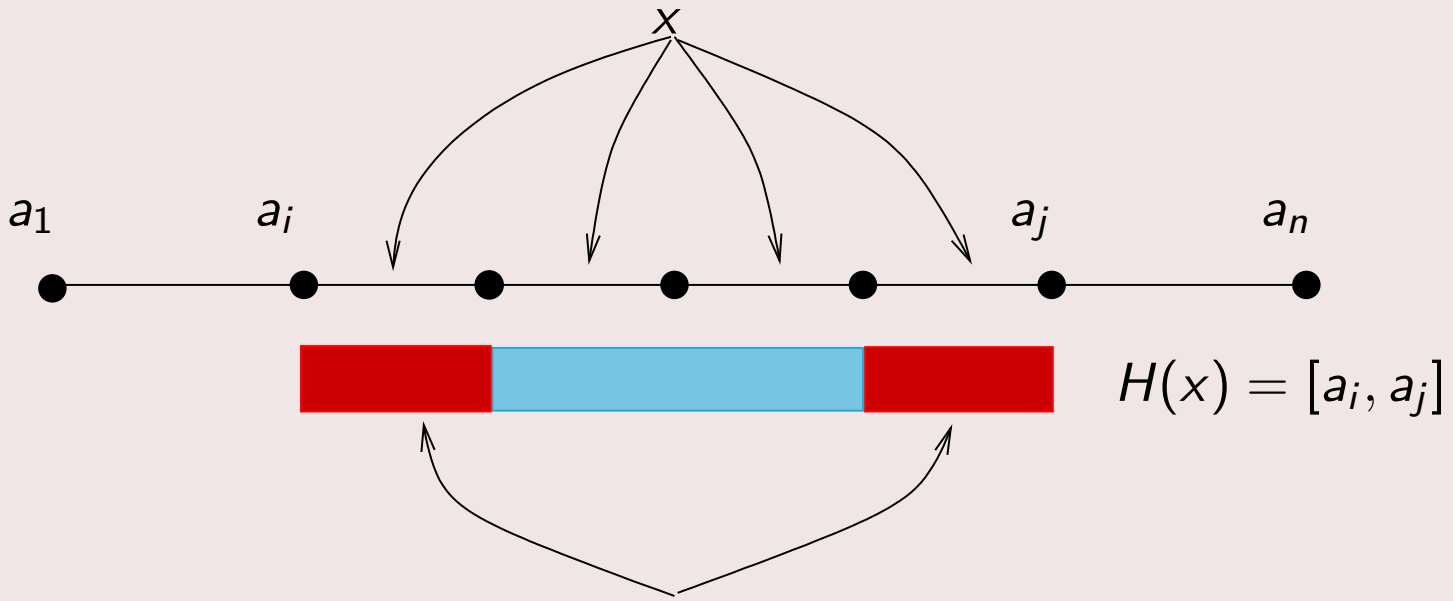
- L'erreur optimale ϵ^* appartient à l'ensemble de taille $O(n^4)$ $\Delta = \{\frac{1}{2}|d(x,y) - d(x',y')| : x, y, x', y' \in X\}$.
- Pour tout $\epsilon \in \Delta$:
  - On construit un ordre partiel $\preccurlyeq$ t.q. tout ordre ϵ-compatible raffine $\preccurlyeq$ ou son dual.
  - Soit $P = (a_1, a_2, \ldots, a_p)$ une chaîne maximale de $\preccurlyeq$.
  - Chaque x de $X^\circ = X \setminus P$ est placé dans un *trou* $H_i = [a_i, a_{i+1}]$.
  - L'algorithme est appelé récursivement sur des sous-ensembles de chaque ensemble d'éléments assignés à un même trou.
  - Si l'algorithme retourne la réponse "non" alors $\epsilon^* > \epsilon$ sinon l'algorithme retourne un ordre total 16ϵ-compatible.

Trous admissibles

- Soit $P = (a_1, a_2, \ldots, a_p)$ une chaine maximale de $\preccurlyeq$. Deux elements a_i, a_{i+1} of P forment un *trou* H_i.
- Le trou H_i est x-*admissible*, si l'ordre total sur $P \cup \{x\}$ obtenu en ajoutant la rélation $a_i \preccurlyeq x \preccurlyeq a_{i+1}$ est ϵ-compatible avec d.
- Une paire de trous (H_i, H_j) est (x, y, c)-*admissible* si l'ordre $\prec$ sur $P \cup \{x, y\}$, obtenu en ajoutant les relations $a_i \prec x \prec a_{i+1}$ et $a_j \prec y \prec a_{j+1}$ à l'ordre $\preccurlyeq$ est $c\epsilon$-compatible.
- Soit $H(x)$ l'ensemble des trous H_i tel que pour tout y il existe H_j tel que (H_i, H_j) est une paire $(x, y, 1)$-admissible.



Trous admissibles

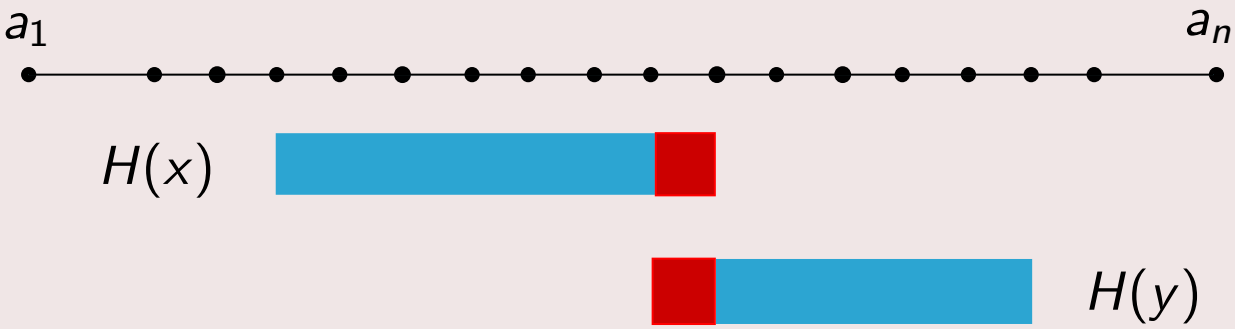


les trous frontières de $H(x)$

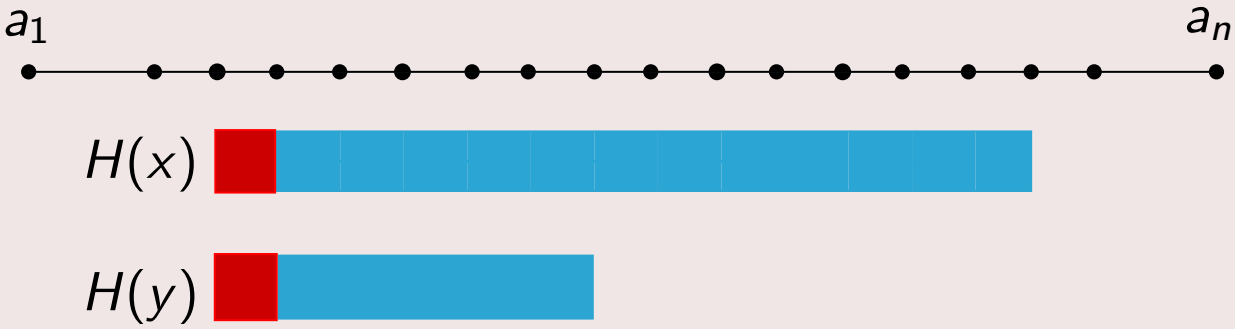
Proposition

Soit H_i un trou frontière de $H(x)$ et H_j un trou frontière de $H(y)$. Si $H(x) \neq H(y)$ alors (H_i, H_j) forme une paire $(x, y, 12)$-admissible.

Trous admissibles



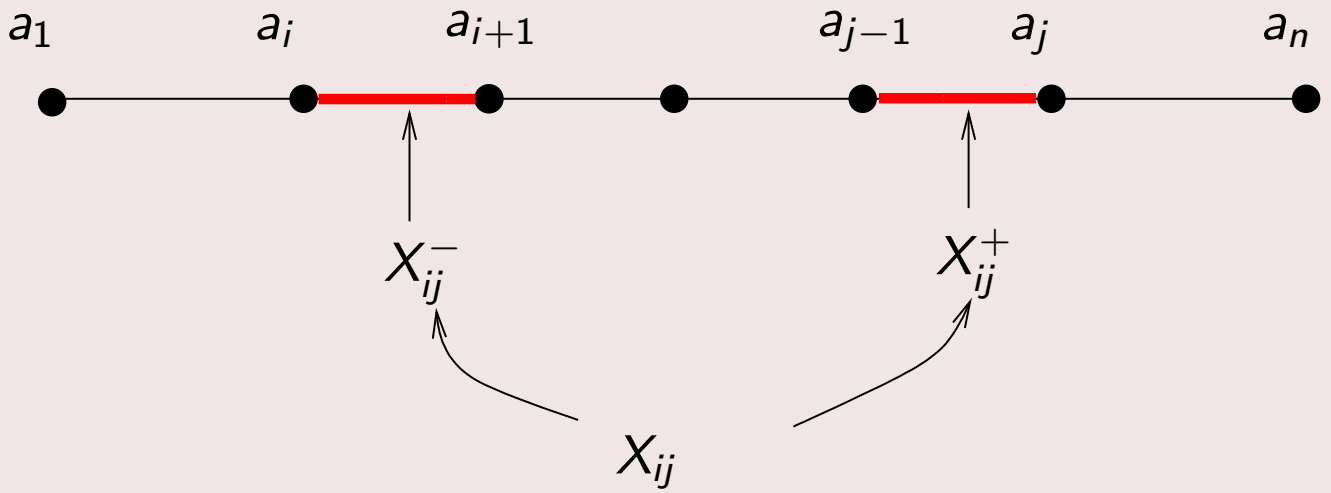
Placer x à gauche de y



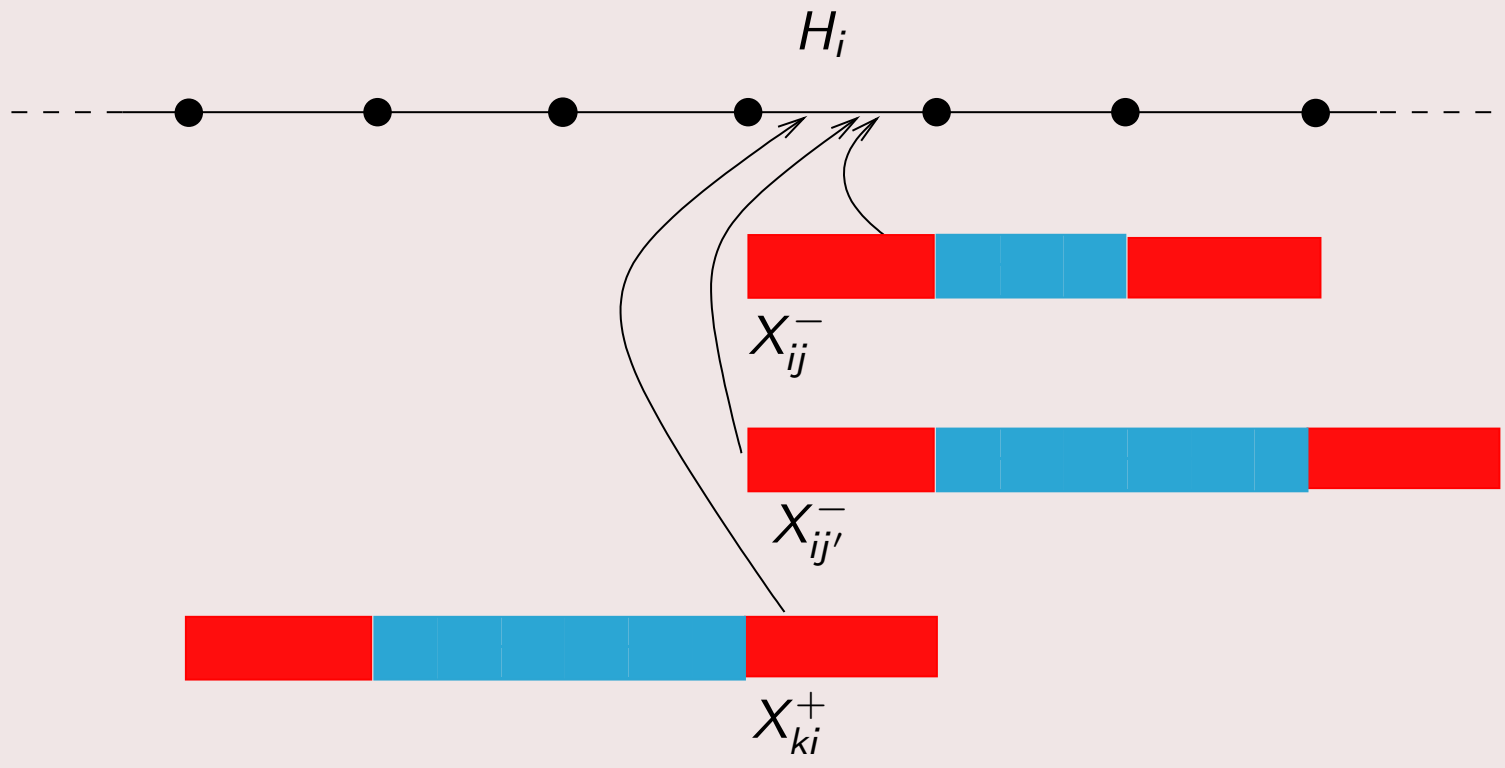
Placer x à gauche de y

Répartition des éléments

- Soit X_{ij} l'ensemble des $x \in X^\circ$ t.q. $H(x) = [a_i, a_j]$.
- On veut partitionner (si possible) l'ensemble X_{ij} en deux sous-ensembles X_{ij}^- et X_{ij}^+, tels que :
  - Les éléments de X_{ij}^- seront placés dans le trou H_i.
  - Les éléments de X_{ij}^+ seront placés dans le trou H_{j-1}.



Répartition des éléments



$$X^+_{1(i+1)} \prec X^+_{2(i+1)} \prec \cdots \prec X^+_{(i-1)(i+1)} \prec X^-_{ip} \prec \cdots \prec X^-_{i(i+2)}$$

Comment partitionner X_{ij} ?

Si $x, y \in X_{ij}$, alors :

- Si $d(x,y) \gg_3 d_x, d_y$ (i.e., $d(x,y)$ est "trop grand"), alors x, y doivent être placés dans des trous frontières différents.
- Si $d(x,y) \ll_3 d_x, d_y$ (i.e., $d(x,y)$ est "trop petit"), alors x, y doivent être placés dans le même trou.
- La transitivité force la position relative de certains éléments de X_{ij}, mais elle ne conduit pas encore à une bipartition.

Ceci est **insuffisant** pour définir une partition de X_{ij} en X_{ij}^+ et X_{ij}^-.

Comment partitionner X_{ij} ?

Pour obtenir une partition de X_{ij} en X_{ij}^+ et X_{ij}^-, on introduit deux graphes orientés $\overrightarrow{\mathcal{L}_{ij}}$ sur X_{ij} et $\overrightarrow{\mathcal{G}_{ij}}$ de sorte que :

- Les composantes fortement connexes (les *cellules*) de $\overrightarrow{\mathcal{L}_{ij}}$ forment des intervalles de tout ordre ϵ-compatible.
- $\overrightarrow{\mathcal{G}_{ij}}$ a pour sommets les cellules de $\overrightarrow{\mathcal{L}_{ij}}$ et une partition de $\overrightarrow{\mathcal{G}_{ij}}$ en deux sous-graphes acycliques donne une partition souhaitée de X_{ij}.
- L'ordre topologique entre les cellules d'une même partie, fixe un ordre 16ϵ-compatible entre les éléments inter-cellules.
- L'algorithme est appelé récursivement sur les cellules et pas sur les ensembles X_{ij}^- et X_{ij}^+.

Comment partitionner X_{ij} ?

- Décider si il existe une partition d'un graphe en deux sous-graphes acycliques est NP-complet dans le cas général.
- A partir des propriétés du graphe $\overrightarrow{\mathcal{G}_{ij}}$, on peut construire une formule 2-SAT Φ_{ij} telle que :
  - Si Φ_{ij} est satisfaisable, alors une affectation "vraie" de Φ_{ij} donne une bipartition requise acylique de $\overrightarrow{\mathcal{G}_{ij}}$ et donc de X_{ij}.
  - Si Φ_{ij} n'est pas satisfaisable, alors $\overrightarrow{\mathcal{G}_{ij}}$ n'a pas de bipartition acyclique et l'algorithme retourne la réponse "non".

Résultat principal

Théorème

Si pour $\epsilon \in \Delta$, l'algorithme retourne la réponse "non", alors d n'est pas ϵ-Robinson, sinon l'ordre retourné est 16ϵ-compatible. En conclusion, cet algorithme est un algorithme d'approximation avec un facteur 16 pour le problème l_∞-FITTING-BY-ROBINSON avec une complexité $O(n^6 \log n)$.