Clustering Dynamique par Rayon

Nicolas K. Blanchard, Nicolas Schabanel

▶ **To cite this version:**

Nicolas K. Blanchard, Nicolas Schabanel. Clustering Dynamique par Rayon. ALGOTEL 2016 - 18èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, May 2016, Bayonne, France. hal-01304598v2

HAL Id: hal-01304598
https://hal.science/hal-01304598v2

Submitted on 27 Apr 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Clustering Dynamique par Rayon †

Nicolas K. Blanchard[1] et Nicolas Schabanel[2]

[1]*ENS Paris, Université Paris Diderot (France) http://www.irif.univ-paris-diderot.fr/users/nkblanchard*
[2]*CNRS, Université Paris Diderot (France) http://www.irif.univ-paris-diderot.fr/users/nschaban*

Comprendre les dynamiques d'évolution de réseaux sociaux et d'infrastructures est un problème crucial dans les domaines comme l'épidémiologie, l'urbanisme ou la marketing viral. Une quantité croissante de données dynamiques sur des réseaux divers est disponible depuis plusieurs années mais les outils pour analyser ces données ne sont pas toujours adaptés. Nous proposons d'utiliser ces données dynamique pour faire des groupes d'individus de comportement similaire restant stables avec le temps. Pour cela nous introduisons une variante dynamique du problème Sum-Radii Clustering, en utilisant le formalisme du problème Dynamic Facility Location, avec la distinction que nous cherchons à minimiser le diamètre des groupes au lieu de la somme des distances au centre. Nous étudions deux adaptations naturelles d'algorithmes probabilistes utilisés pour Dynamic Facility Location (marchant respectivement dans le cas général et quand on se restreint à des espaces métriques). Dans le premier cas, l'algorithme atteint la même borne d'approximation et nous proposons une amélioration, aussi valable pour l'algorithme original (faisant passer le facteur d'approximation de $O(\log nT)$ à $O(\log n)$, où n est le nombre de clients et T la durée en nombre de pas de temps). Enfin, nous montrons que dans le cas métrique, les outils actuels ne permettent pas encore de donner un meilleur résultat, et exhibons un contre-exemple pour le deuxième algorithme, prouvant qu'il ne peut pas atteindre une approximation constante.

Mots-clefs : Facility location, algorithmes d'approximation, clustering, graphes dynamiques

1 Introduction

La multiplication des données dynamiques et la difficulté à les analyser et même à les représenter a montré l'intérêt des algorithmes de Facility Location, utilisés en recherche opérationnelle depuis les années 1960 pour optimiser la répartition d'entrepôts quand on cherche à livrer des clients. Il est en effet facile d'adapter ces algorithmes pour faire du clustering qui soit stable dans le temps comme montré par [3]. Jusqu'ici ces algorithmes essayaient généralement de minimiser la somme des distances à la facility (centre du cluster), mais cela fait apparaître des effets de moyenne : un individu pourrait être arbitrairement loin du centre de son cluster car cette distance est absorbée par le nombre de personnes très proches du centre, ce qui est injuste pour l'individu éloigné. Prendre le rayon – c'est-à-dire le maximum des distances au centre – donnerait donc des solutions plus équitables.

Le problème

On considère une variante du problème de Facility Location où on cherche à faire en sorte qu'un ensemble de clients aient tous accès à un service pendant un intervalle de temps (on a ici à l'avance toute l'évolution du réseau, contrairement au cas online). En pratique on a un ensemble d'endroits où construire des antennes (les facilities) et un ensemble de n clients se déplaçant sur le territoire, et on veut que chaque client soit couvert à chaque instant par au moins une antenne tout en minimisant les coûts. La fonction de coût correspond ici au nombre d'antennes (coût d'ouverture), à la puissance de celles-ci (coût des rayons, c'est-à-dire la distance de l'antenne à son client le plus éloigné), et aux coûts de changement (chaque client changeant d'antenne crée un petit coût). Ces coûts de changement ont été introduits [3] et permettent d'introduire une stabilité temporelle dans les solutions, donnant une meilleure solution que des analyses statiques à chaque instant, correspondant mieux aux dynamiques perçues et vérifiées expérimentalements sur ces réseaux.

†Ces travaux ont été financés par les bourses ANR-12-BS02-005 RDAM et IXXI-Molecal

Ce type de problème a jusqu'ici principalement été étudié dans le cas où le coût correspond à la somme des distances entre client et facility associée. Cela a été étendu et résolu de manière asymptotiquement optimale dans le cas dynamique [1] ainsi que dans le cas statique pour la somme des rayons [2], mais jamais en combinant ces deux extensions, ce que nous proposons ici.

Formalisation

On utilise les notations de Dynamic Facility Location et on considère donc un ensemble $\mathcal{C}$ de n clients, un ensemble $\mathcal{F}$ de m facilities et un graphe des distances $d^t(i,j)$ pour chaque paire (client, facility) et chacun des T pas de temps $t \in \mathcal{T}$. On a aussi un ensemble de coûts d'ouverture $f_i, i \in \mathcal{F}$, et un coût de changement g. Le but est de produire un assignement de chaque client à une facility ouverte pour chaque t, en minimisant la fonction de coût suivante

$$\sum_{i\in\mathcal{F},t\in\mathcal{T}} f_i \times y_i^t + \sum_{i\in\mathcal{F},t\in\mathcal{T}} y_i^t \times r_i^t + g \sum_{j\in\mathcal{C},t\in\mathcal{T}\setminus\{0\}} z_j^t$$

où y_i^t vaut 1 si la facility i est ouverte au temps t (0 sinon), r_i^t est le rayon de i (distance au client le plus éloigné qui lui est assigné) et z_j^t vaut 1 si j a changé de facility entre $t-1$ et t (0 sinon).

Nos Résultats

Théorème 1. *Dynamic Sum-Radii Clustering (DSRC) n'a de* $(1-o(1))\ln n$*-approximation dans le cas général (non-métrique) que si* $\mathbf{P} = \mathbf{NP}$.

Ce résultat se démontre de manière standard mais il faut noter que l'instance n'utilise pas le côté dynamique, et que cette borne d'inapproximation est donc toujours valable dans le cas statique. La preuve ne marche que dans le cas où l'inégalité triangulaire n'est pas obligatoirement vérifiée et nous n'avons pas d'équivalent dans le cas métrique.

Théorème 2. *Il existe un algorithme probabiliste fonctionnant en temps polynomial donnant pour toute entrée une solution dont le coût a une probabilité au moins* $\frac{1}{2}$ *d'être inférieur à* $4\ln(n)$ *fois le coût optimal pour le problème DSRC.*

On peut ici améliorer la probabilité de succès et la constante d'approximation (sans changer le temps de calcul asymptotique), et notamment atteindre une 1.93-approximation dans le cas statique, et ces résultats sont asymptotiquement optimaux. De plus, la méthode développée permet d'améliorer les résultats de [3] qui avaient une $\ln(nT)$ approximation pour Dynamic Facility Location (avec la somme des distances), donnant un algorithme de $\ln(n)$-approximation asymptotiquement optimal.

Nous n'avons pas d'algorithme donnant une approximaiton asymptotiquement meilleure que $O(\ln(n))$ dans le cas métrique mais nous avons montré que les techniques utilisées pour résoudre les variantes métriques dans le cas somme des distances ne marchent pas. En effet, nous avons étudié l'adaptation naturelle de l'algorithme ANS[1] qui donne une approximation constante dans le cas métrique des somme des distances et obtenu :

Théorème 3. *L'adaption naturelle de l'algorithme ANS donne au mieux une* $\Omega(\ln\ln n)$*-approximation pour DSRC dans le cas métrique.*

Pour prouver cela nous introduisons un contre-exemple basé sur une structure arborescente et utilisons un lemme combinatoire digne d'intérêt par lui-même :

Lemme. *Soit T un arbre binaire parfait enraciné en r avec $N-1 = 2^n - 1$ noeuds et une permutation uniforme donnant à chaque noeud une clé unique entre 1 et $N-1$. Alors avec probabilité au moins $1/3$, il existe une feuille f telle que la clé de chaque ancêtre de f est inférieure à la clé de r.*

2 Algorithmes d'approximation dans le cas général

Relaxation linéaire

Ce problème se prête très naturellement à la programmation entière et toute solution est aussi solution du programme linéaire (PL) de la page suivante, où la variable secondaire x_{ij}^t vaut 1 si j est connecté à i au temps t.

$$\begin{cases} \text{Minimiser} & \sum_{i\in\mathcal{F},t\in\mathcal{T}} f_i \times y_i^t + \sum_{i\in\mathcal{F},t\in\mathcal{T}} y_i^t \times r_i^t + g \sum_{i\in\mathcal{F},j\in\mathcal{C},t\in\mathcal{T}\setminus\{0\}} z_{ij}^t \\ \text{Avec les contraintes} & \begin{array}{ll} \forall j,t & \sum_i x_{ij}^t \geq 1 \\ \forall i,j,t & x_{ij}^t \leq \sum_{r\geq d(i,j)} y_{ir}^t \\ \forall i,j,t & z_{ij}^t \geq x_{ij}^t - x_{ij}^{t-1} \\ \forall i,j,t & x_{ij}^t, z_{ij}^t, y_{ir}^t \geq 0 \end{array} \end{cases}$$

Naturellement trouver une solution entière à ce programme linéaire n'est pas faisable en temps polynomial si $\mathbf{P} \neq \mathbf{NP}$ mais on peut relaxer la contrainte d'avoir des variables entières pour pouvoir utiliser un solveur linéaire. On obtient alors une solution pour laquelle les variables x_{ij}^t, z_{ij}^t et y_{ir}^t peuvent être fractionnaires et dont le coût est inférieur ou égal au coût optimal d'une solution entière.

Une $O(\log nT)$-approximation

On commence par s'inspirer de [3] en utilisant un preprocessing qui garantit que pour chaque client j, on a un ensemble d'intervalles de temps tels que sur chaque intervalle I les x_{ij}^t sont constants pour tout $t \in I$, et vérifient les autres contraintes (ce preprocessing multiplie au plus par 2 le coût de la solution). Une fois cela fait on peut lancer l'algorithme suivant :

Entrée: La solution (x,y,y) au programme linéaire passée par le preprocessing
Répéter
Pour chaque facility $i \in \mathcal{F}$ **faire**
Tirer a_i uniformément dans $[0,1]$
Ouvrir la facility i à chaque temps t avec le plus grand r tel que $a_i \leq \sum_{p\geq r} y_{ip}^t$
Pour chaque client $c \in \mathcal{C}$ et chacun de ses intervalles **faire**
Si c est couvert par une facility i sur tout l'intervalle **alors**
Assigner j à n'importe quelle facility le couvrant
Jusqu'à ce que Tous les clients soient assignés à des facilities sur chaque intervalle (sans fermer les facilities entre les répétitions)

La probabilité qu'une facility i ne couvre pas un client j est donc $1 - \sum_{r'\geq d(i,j)} y_{ir}$. Au cours d'une répétition, la probabilité qu'un client soit couvert sur un intervalle est égale à la probabilité qu'au moins une des facility soit ouverte avec un rayon suffisant, soit

$$1 - \prod_{i\in\mathcal{F}} \Pr\{i \text{ ne couvre pas } j \text{ sur } I\}$$

On sait que $x_{i,j}^t$ est constant sur l'intervalle I et que pour tout $t \in I$, $x_{i,j}^t \leq \sum_{r\geq d(i,j)} y_{ir}^t$, donc j est couvert sur I avec probabilité $1 - \prod_{i\in\mathcal{F}} (1 - x_{ij}^t)$. Comme $\sum_{i\in\mathcal{F}} x_{ij}^t \geq 1$, par convexité on obtient qu'un client est couvert sur un intervalle donné avec probabilité au moins $\frac{1}{2}$. En répétant ce processus de manière indépendante $O(\log(\#\text{intervalles}))$ fois, on peut montrer que la probabilité qu'un intervalle soit couvert dans au moins une répétition est supérieure à $\frac{1}{2\#\text{intervalles}}$, et par union bound ils sont tous couverts avec probabilité $\frac{1}{2}$.

Comme chaque répétition coûte le coût de la solution du PL, l'espérance est au plus égale au nombre de répétitions fois le coût optimal. On a donc une $O(\log(\#\text{intervalles}))$-approximation pour DSRC.

$O(\log(n))$-approximation

Pour améliorer l'algorithme précédent on découpe notre instance de manière gloutonne en périodes telles que le nombre d'intervalles sur chaque période est compris entre n et $2n$, et on lance l'algorithme sur chacune de ces sous-instances. On a à chaque fois une $\log(3n)$-approximation au pire, et le coût de recollage

(au plus n changements par période) est absorbé par les changements présents dans chaque période. On obtient donc une $O(\log n)$-approximation pour DSRC. De plus cette amélioration peut aussi s'appliquer à l'algorithme de [3] pour donner une $O(\log n)$-approximation dans le cas classique (somme des distances).

3 Cas métrique

Dans le cas métrique la preuve de $\ln(n)$-inapproximation ne marche plus, et il est donc possible qu'il y ait un algorithme donnant une meilleure approximation (idéalement constante). Deux pistes naturelles sont les adaptations des algorithmes présents dans [2] et [1]. Le problème est que le premier peut se tromper d'un facteur exponentiel même sur des contre-exemples simples (et le modifier pour éviter cela semble difficile). Le deuxième algorithme, basé sur des « horloges exponentielles » et faisant des redirections locales basées sur la solution du PL s'adapte très naturellement au cas des rayons. Nous avons construit une instance (une structure arborescente fractale en n dimensions) qui force cet algorithme à « hesiter » entre plusieurs solutions optimales mutuellement exclusives. Le lemme permet de montrer qu'on a une forte chance d'être redirigé vers une facility lointaine, et vu que les clients ne coopèrent pas on peut prouver que le coût de cette confusion est $\Omega(\log\log n) \times \mathbf{OPT}$.

Conclusion

Le tableau suivant représente l'état de l'art pour Dynamic Facility Location et DSRC (nos contributions sont indiquées par une étoile).

	Cas métrique (algorithme/difficulté)		Non-Métrique (algorithme/difficulté)	
	Somme des distances	Somme des Rayons	Somme des distances	Somme des Rayons
Statique	1.52 [7]/ 1.46[4]	3/NP-Hard [2]	$O(\log n)$[3]/$\Omega(\log n)$	$2\log n/(1-\varepsilon)\log n$ *
Dynamique	14 [1]/ 1.46[4]	?/NP-Hard [2]	$O(\log n)$*/$\Omega(\log n)$[5]	$4\log n/(1-\varepsilon)\log n$ *

Comme on peut voir la vraie question restante est celle du cas métrique dynamique. Nos recherches dans ce domaine ont pour le moment été infructueuses mais plusieurs pistes devraient permettre de montrer une borne sur la qualité des solutions de ANS[1] modifié. Nous essayons aussi une autre méthode inspirée par [2] pour laquelle nous avons quelques résultats partiels.

Nous comptons aussi tester notre algorithme sur plusieurs ensembles de données pour voir s'il fournit de nouvelles intuitions et fonctionne bien en pratique. On peut noter que dans nos deux algorithmes le facteur déterminant d'un point de vue calcul est la résolution du programme linéaire, nos algorithmes peuvent donc fonctionner en temps $O(nm(n+m)^{1/2})$ [6].

Références

[1] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. In *SODA*, pages 708–721, 2015.

[2] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci.*, 68(2) :417–441, 2004.

[3] David Eisenstat, Claire Mathieu, and Nicolas Schabanel. Facility location in evolving metrics. In *ICALP*, pages 459–470, 2014.

[4] Sudipto Guha and Samir Khuller. Greedy strikes back : Improved facility location algorithms. *J. Algorithms*, 31(1) :228–248, 1999.

[5] Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Math. Program.*, 22(1) :148–162, 1982.

[6] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming : Solving linear programs in $\tilde{O}$(vrank) iterations and faster algorithms for maximum flow. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 424–433, Washington, DC, USA, 2014. IEEE Computer Society.

[7] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2) :411–432, 2006.