Deux méthodes incrémentales pour le maintien d'un arbre de connexion

Nicolas Thibault, Christian Laforest

Tour Evry 2, LaMI/CNRS, Université d'Evry, 523 place des Terrasses, 91000 Evry, France
Email:{nthibaul,laforest}@lami.univ-evry.fr

Cet article est consacré au problème suivant : étant donné un ensemble de machines dans un réseau, construire une structure permettant de les connecter tout en minimisant le poids de la structure, ainsi que la distance moyenne et maximale entre machines. Nous traitons ici plus spécifiquement de l'aspect incrémental. Nous prenons en compte le fait que les machines à connecter ne sont pas toutes connues dès le début, mais sont dévoilées au fur et à mesure.
Après avoir modélisé cette situation sous forme de problème de graphe, nous proposons d'évaluer et de comparer deux méthodes (de type *online*) réalisant ces ajouts de manière incrémentale.

Mots clés : Arbre de connexion, algorithme online, optimisation.

Nous nous intéressons dans cet article aux communications entre *membres* d'un *groupe* de machines. Nous souhaitons pour cela les interconnecter par une structure, bâtie sur le réseau sous-jacent. Il est alors naturel d'exiger de la part de cette structure une certaine performance. Nous nous sommes spécifiquement intéressés ici au *temps de communication moyen* et au *temps de communication maximum* entre membres. De plus, dans un souci d'économie d'utilisation des ressources physiques du réseau, nous nous sommes également intéressés au *poids* de la structure bâtie. D'autre part, une des principales caractéristiques de la formation d'un groupe au sein d'un réseau est sa mouvance. En effet, connaître à l'avance la totalité des membres du groupe à couvrir n'est pas toujours possible en pratique. Nous avons donc choisi d'étudier le cas *dynamique* qui consiste à incrémenter une structure de connexion existante pour incorporer au mieux chaque nouveau membre.

Les applications potentielles. Ce problème a de nombreuses applications concrètes. On peut évoquer toute forme de forum ou de réunion via un réseau. Dans ce type de rassemblements virtuels, des participants peuvent arriver, dans un ordre et à des moments inconnus à l'avance. Les identités de ces futurs participants ne sont même pas toujours connues à l'avance. Par exemple, dans les systèmes pair à pair (*peer-to-peer*) il est inconcevable de prédire qui va échanger des données avec qui. Les échanges se font "au fil de l'eau", en fonction de paramètres qu'il est difficile de quantifier à l'avance. Pour prendre en compte cette difficulté de prévision, nous nous imposons un schéma d'interconnexion en *arbre* (qui permet donc une grande facilité de routage) dans lequel les critères à minimiser sont la *distance moyenne* et la *distance maximale* entre participants (pour la performance des communications), ainsi que le poids de l'arbre (coût de réservation des ressources ou mesure de l'emcombrement de la structure).

Deux méthodes étudiées. Nous proposons d'étudier deux schémas simples d'interconnexion et d'ajouts des machines. Ces schémas peuvent être facilement implémentés sur un réseau. Ce qui va nous préoccuper principalement ici c'est la *qualité* de connexion offerte par chacune des deux propositions. Cette qualité est mesurée en termes de *rapports de compétitivité*, notion venant de l'algorithmique online et présentée en section 1, avec la modélisation du problème et ses contraintes (voir [BEY98] pour des références sur les algorithmes onlines). En section 2, nous donnons un certain nombre de bornes inférieures pour chacun des critères étudiés, provenant d'analyses fines de l'impact des contraintes imposées. Puis, en section 3, nous étudions la première méthode, l'algorithme *sommet-ajout*, qui provient de [TL03]. La seconde méthode, l'algorithme *plus-proche-ajout*, provient de [IW91] et est étudiée en section 4. Les résultats obtenus sont synthétisés et discutés en section 5.

1 Modélisation du problème et des contraintes.

Soit $G = (V,E,w)$ un graphe pondéré quelconque, avec V l'ensemble de ses sommets (représentant l'ensemble des machines), E l'ensemble de ses arêtes (représentant l'ensemble des liens) et w une fonction de poids positive sur les arêtes ($\forall e \in E$, $w(e) > 0$ représente la longueur du lien e). Nous considérons ici que plus un lien est long, plus il nécessite de resources physiques et donc plus le poids qui lui est attribué est important. On notera $d_G(u,v)$ la *distance entre u et v dans G*, c'est-à-dire la somme des poids des arêtes d'un chemin de poids minimum entre u et v dans G.

Notations. Dans la suite de l'article, nous adopterons les notations suivantes. Soient $G = (V,E,w)$ un graphe, $M \subseteq V$ un groupe quelconque. On notera la *somme des distances* de M dans G par $C_G(M) = \sum_{u,v \in M} d_G(u,v)$ (cette quantité permet de déduire de manière évidente la distance moyenne entre sommets), le *diamètre* de M dans G par $D_G(M) = max\{d_G(u,v) : u,v \in M\}$ et le *poids* d'un arbre $T(M) = (V_T,E_T,w_T)$ couvrant M par $W(T(M)) = \sum_{e \in E_T} w_T(e)$. On notera $T_S(M)$ l'arbre de poids minimum couvrant M (ou *arbre de Steiner*).

On définit le premier groupe M_0 comme étant un sommet de départ quelconque $u_0 \in V$ ($M_0 = \{u_0\}$). On note $S = u_1,u_2,\dots,u_i$ la suite des i sommets de G à ajouter au groupe courant, et on note $\forall\ j \geq 1$, $M_j = M_{j-1} \bigcup \{u_j\}$ le $j^{ème}$ groupe.

Contraintes de construction. Notre objectif est *d'inclure* les nouveaux membres $S = u_1,u_2,\dots,u_i$ qui arrivent au fur et à mesure. Pour cela, nous devons construire des sous structures, notées T_i, couvrant chaque groupe intermédiaire M_i en respectant, pour tout $i \geq 0$, les impératifs suivants :

- T_i doit être un *arbre*, pour simplifier les mécanismes de routage et de duplication de l'information.
- T_i doit être construit sans modifier aucune des liaisons présentes dans T_{i-1} (ce qui implique $V(T_{i-1}) \subseteq V(T_i)$ et $E(T_{i-1}) \subseteq E(T_i)$). En effet, on ne peut pas systématiquement reconstruire un nouvel arbre à chaque nouvelle arrivée, ce qui serait trop coûteux en temps et perturberait gravement les communications déjà en cours. La méthode d'insertion d'un nouveau membre doit donc consister à le connecter à l'arbre existant en ajoutant au plus un chemin du nouveau membre vers T_{i-1}.
- T_i doit induire une somme des distances $C_{T_i}(M_i)$ entre membres de M_i et un diamètre $D_{T_i}(M_i)$ les plus petits possibles, pour des questions déjà évoquées de performances des communications.
- T_i doit avoir un poids $W(T_i(M))$ le plus faible possible, pour minimiser l'encombrement du réseau.

Mesure de qualité d'une méthode : le rapport de compétitivité. Dans ce type de problèmes *onlines*, les données (ici les nouveaux membres u_i) sont *révélées* les unes après les autres. La mesure de performance de tels algorithmes est connue sous le nom de *rapport de compétitivité*. Celui-ci évalue, pour un critère donné, la qualité (au pire cas) d'un algorithme en fonction de l'optimum à chaque étape (ici à chaque nouveau membre u_i ajouté). Il mesure l'impact qu'a la non connaissance du futur (imposée par le modèle de dévoilement des données online) sur les paramètres étudiés. On définit un rapport de compétitivité par critère à minimiser. Soit $\mathscr{A}$ un algorithme online construisant à la $k^{ème}$ étape un arbre T_k. $\mathscr{A}$ a un rapport de compétitivité c_s (ou est dit c_s-compétitif) pour la somme des distances ssi $\forall\ G,\forall\ k,\ 0 \leq k \leq i$, on a $C_{T_k}(M_k) \leq c_s.C_G(M_k)$. $\mathscr{A}$ a un rapport de compétitivité c_d pour le diamètre ssi $\forall\ G,\forall\ k,\ 0 \leq k \leq i$, on a $D_{T_k}(M_k) \leq c_d.D_G(M_k)$. $\mathscr{A}$ a un rapport de compétitivité c_w pour le poids ssi $\forall\ G,\forall\ k,\ 0 \leq k \leq i$, $W(T_k) \leq c_w.W(T_S(M_k))$.

2 Bornes inférieures pour les divers rapports de compétitivité

Borne inférieure pour la somme des distances. Le théorème 1 montre que *tout algorithme* traitant notre problème induit un rapport de compétitivité au moins linéaire en i (le nombre de nouveaux sommets) pour la somme des distances (la preuve figure dans [TL03, TL04]). Puisqu'il s'agit d'une borne inférieure, nous utilisons la notation asymptotique usuelle Ω.

Théorème 1 *Il existe $G = (V,E,w)$, et $M_0 \subset \dots \subset M_i \subseteq V$, tels que pour tout algorithme incrémental online construisant un arbre T_i couvrant M_i, respectant les contraintes de construction, on a :*

$$C_{T_i}(M_i) \geq \Omega(i) C_G(M_i)$$

Borne inférieure pour le diamètre. Le théorème 2 montre que *tout algorithme* traitant notre problème est au moins 2-compétitif pour le diamètre (la preuve figure dans [TL04]).

Théorème 2 *Il existe $G = (V,E,w)$, et $M_0 \subset \ldots \subset M_i \subseteq V$, tels que pour tout algorithme incrémental online construisant un arbre T_i couvrant M_i, respectant les contraintes de construction, on a :*

$$D_{T_i}(M_i) \geq 2D_G(M_i)$$

Borne inférieure pour le poids. Le théorème 3, du à Makoto Imase et Bernard M. Waxman [IW91], montre que *tout algorithme* traitant notre problème est au moins $\Omega(\log_2(i))$-compétitif pour le poids.

Théorème 3 *Il existe $G = (V,E,w)$, et $M_0 \subset \ldots \subset M_i \subseteq V$, tel que pour tout algorithme incrémental online construisant un arbre T_i couvrant M_i, respectant les contraintes de construction, $\forall j$, $1 \leq j \leq i$, on a :*

$$W(T_j(M_j)) \geq (1 + \frac{1}{2}\lfloor \log_2(j) \rfloor) W(T_S(M_j))$$

3 L'algorithme *sommet-ajout*

Description de l'algorithme *sommet-ajout*. À l'étape 0, $M_0 = \{u_0\}$, donc T_0 est réduit au sommet u_0. Pour tout k, $1 \leq k \leq i$, *sommet-ajout* incrémente l'arbre $T_{k-1} = (V_{T_{k-1}}, E_{T_{k-1}})$ en connectant le nouveau sommet u_k à T_{k-1} par un plus court chemin de G entre u_k et u_0, de telle façon que la nouvelle structure T_k ainsi créée reste un arbre.

Bornes supérieures. Les preuves des théorèmes 4, 5 et 6 figurent dans [TL04]. D'après les théorèmes 1 et 2, les résultats des théorèmes 4 et 5 sont optimaux en ordre de grandeur.

Théorème 4 Sommet-ajout *est $(i+1)$-compétitif pour la somme des distances, c'est à dire :*

$$\forall i \geq 0, \quad C_{T_i}(M_i) \ \leq \ (i+1)C_G(M_i)$$

Théorème 5 Sommet-ajout *est 2-compétitif pour le diamètre, c'est à dire :*

$$\forall i \geq 0, \quad D_{T_i}(M_i) \leq 2D_G(M_i)$$

Théorème 6 Sommet-ajout *est i-compétitif pour le poids, c'est à dire :*

$$\forall i \geq 0, \quad W(T_i) \ \leq \ i.W(T_S(M_i))$$

Cette borne supérieure est atteinte asymptotiquement.

4 L'algorithme *plus-proche-ajout*

Description de l'algorithme *plus-proche-ajout*. À l'étape 0, $M_0 = \{u_0\}$, donc T_0 est réduit au sommet u_0. Pour tout k, $1 \leq k \leq i$, *plus-proche-ajout* incrémente l'arbre $T_{k-1} = (V_{T_{k-1}}, E_{T_{k-1}})$ en connectant le nouveau sommet u_k à T_{k-1} par un plus court chemin dans G entre u_k et t_{k-1}, où $t_{k-1} \in V_{T_{k-1}}$ est tel que $d_G(u_k, t_{k-1}) = min\left\{d_G(u_k, t) : t \in V_{T_{k-1}}\right\}$.

4.1 Borne supérieure pour la somme des distances.

Le théorème 7 montre que l'algorithme *plus-proche-ajout* est $2i$-compétitif pour la somme des distances. D'après le théorème 1, ce résultat est optimal en ordre de grandeur. Les lemmes 1 et 2 ne sont que deux résultats techniques préliminaires (la preuve du lemme 1 figure dans [Laf02]).

Lemme 1 *Soient $G = (V,E,w)$ un graphe et $M \subseteq V$. Soit r un médian de M, c'est à dire un sommet $r \in M$ vérifiant $\sum_{u \in M} d_G(u,r) = min\{\sum_{u \in M} d_G(u,v) : v \in M\}$. On a alors :*

$$C_G(M) \geq |M| \sum_{u \in M} d_G(u,r)$$

Lemme 2 *Soit $G = (V,E,w)$ un graphe. Pour tout $M_i \subseteq V$,* plus-proche-ajout *construit un arbre de plus courts chemins T_i couvrant M_i tel que pour tout $r \in M_i \subseteq V$ on a :*

$$D_{T_i}(M_i) \leq 2 \sum_{u \in M_i} d_G(u,r)$$

Preuve. Par définition du diamètre $D_T(M)$, et du poids $W(T(M))$ d'un arbre T couvrant M, on a :

$$\begin{aligned} D_{T_i}(M_i) &\leq W(T_i(M_i)) = \sum_{k=1}^{i} d_G(t_{k-1},u_k) \leq \sum_{k=1}^{i} d_G(u_{k-1},u_k) \quad \text{(par définition de \textit{plus-proche-ajout})} \\ &\leq \sum_{k=1}^{i} (d_G(u_{k-1},r) + d_G(r,u_k)) = \sum_{k=0}^{i-1} d_G(u_k,r) + \sum_{k=1}^{i} d_G(u_k,r) \leq 2 \sum_{u \in M_i} d_G(u,r) \\ &\quad \text{(d'après l'inégalité triangulaire)} \end{aligned}$$

□

Théorème 7 Plus-proche-ajout *est 2i-compétitif pour la somme des distances, c'est à dire :*

$$\forall i \geq 0, \quad C_{T_i}(M_i) \leq 2iC_G(M_i)$$

Preuve. Comme $|M_i| = i + 1$ et $\forall u,v \in M_i$, $d_{T_i}(u,v) \leq D_{T_i}(M_i)$, on a :

$$\begin{aligned} C_{T_i}(M_i) &\leq i(i+1)D_{T_i}(M_i) \leq 2i(i+1) \sum_{u \in M_i} d_G(u,r_i) \leq 2iC_G(M_i) \\ &\quad \text{(d'après les lemmes 1 et 2, avec } r_i \in M_i \text{ un médian de } M_i\text{)} \end{aligned}$$

□

4.2 Borne supérieure pour le diamètre.

Le théorème 8 montre que l'algorithme *plus-proche-ajout* est i-compétitif pour le poids, et le lemme 3 montre que la borne supérieure est atteinte asymptotiquement.

Théorème 8 Plus-proche-ajout *est i-compétitif pour le diamètre, c'est à dire :*

$$\forall i \geq 0, \quad D_{T_i}(M_i) \leq i.D_G(M_i)$$

Preuve. Pour tout k, $1 \leq k \leq i$, et pour tout j, $1 \leq j \leq k$, par définition du diamètre et comme $u_k,u_{j-1} \in M_k \subseteq M_i$, on a $d_G(u_k,t_{k-1}) \leq d_G(u_k,u_{j-1}) \leq max\{d_G(u,v) : u,v \in M_k\} \leq D_G(M_i)$. On obtient alors $\sum_{k=1}^{i} d_G(u_k,t_{k-1}) \leq \sum_{k=1}^{i} D_G(M_i) = i.D_G(M_i)$. Ainsi, étant donné que $D_{T_i}(M_i) \leq W(T_i(M_i)) = \sum_{k=1}^{i} d_G(u_k,t_{k-1})$, on a $D_{T_i}(M_i) \leq i.D_G(M_i)$. □

Lemme 3 *La borne supérieure du théorème 8 est atteinte asymptotiquement.*

Preuve. Soit $0 < \varepsilon$ et soit $K_i = (V_i,E_i,w_i)$ le graphe complet tel que $V_i = \{s_0,\ldots,s_i\}$ et $w_i(e) = 1 - \varepsilon$ si $e \in \{s_0s_1,s_1s_2,\ldots,s_{i-1}s_i\} \subset E_i$, $w_i(e) = 1$ sinon. Soit $M_0 = \{s_0\}$. Soit la séquence d'ajouts $S = s_1,\ldots,s_i$. À l'étape i, *plus-proche-ajout* a nécessairement construit un arbre $T_i = (V_{T_i},E_{T_i},w_i)$ qui est le chemin $s_0s_1,s_1s_2,\ldots,s_{i-1}s_i$. On a donc $\lim_{\varepsilon\to 0} \frac{D_{T_i}(M_i)}{D_{K_i}(M_i)} = \lim_{\varepsilon\to 0} \frac{i(1-\varepsilon)}{1} = i$. □

4.3 Borne supérieure pour le poids.

D'après le théorème 3, le résultat suivant (voir preuve dans [IW91]) est optimal en ordre de grandeur.

Théorème 9 Plus-proche-ajout *est $\lceil \log_2(i+1) \rceil$-compétitif pour le poids, c'est à dire :*

$$\forall i \geq 0, \quad W(T_i) \leq \lceil \log_2(i+1) \rceil W(T_S(M_i))$$

5 Synthèse et comparaison des deux algorithmes

Nous résumons les performances des deux algorithmes dans un tableau à deux dimensions (voir tableau 1), où nous avons respectivement en colonne, les bornes inférieures universelles (basées sur les contraintes de construction et indépendantes de la méthode utilisée), les rapports de compétitivité de *sommet-ajout*, les rapports de compétitivité de *plus-proche-ajout*, et ce pour chaque critère (représentés en ligne).

Les bornes inférieures pour la somme des distances, le diamètre et le poids viennent respectivement des théorèmes 1, 2 et 3. Les rapports de compétitivité de *sommet-ajout* pour la somme des distances, le diamètre et le poids viennent respectivement des théorèmes 4, 5 et 6. Les rapports compétitivité de *plus-proche-ajout* pour la somme des distances, le diamètre et le poids viennent respectivement des théorèmes 7, 8 et 9.

Critères	$C_{T_i}(M_i)$	$D_{T_i}(M_i)$	$W(T_i(M_i))$
Bornes inférieures	$\Omega(i)$	2	$\Omega(\log_2(i))$
sommet-ajout	$i+1$	2	i
plus-proche-ajout	$2i$	i	$\lceil \log_2(i+1) \rceil$

TAB. 1 – *Comparaison tri-critères de* sommet-ajout *et* plus-proche-ajout.

Si les deux algorithmes sont équivalents en ordre de grandeur pour la somme des distances, *plus-proche-ajout* est bien meilleur pour le poids ($\lceil \log_2(i+1) \rceil$ v.s. i), alors que *sommet-ajout* est bien meilleur pour le diamètre (2 v.s. i).

Nous soulignons le fait qu'aucun des deux algorithmes n'est meilleur que l'autre de manière absolue. Le choix d'un algorithme au détriment de l'autre dépendra fortement de l'application et donc du critère à privilégier. Par exemple, si l'on considère la création online d'un système pair à pair dédié à l'échange de données, pour lesquelles les délais de réception ne sont pas soumis à des contraintes temporelles strictes, il est plus important de ne pas trop emcombrer le réseau, et donc de minimiser la quantité de ressources physiques allouées (d'autant plus que de nombreuses sessions de ce type peuvent être créées simultanément). Il sera alors préférable d'utiliser *plus-proche-ajout* afin de mieux contrôler le poids de la structure, quitte à perdre en qualité au niveau du délais de transmission. À l'inverse, si l'on considère une application avec de fortes contraintes temporelles, par exemple une réunion sur réseau où les participants doivent pouvoir discuter entre eux dans des conditions les plus proches possibles du temps réel, il sera alors préférable de maîtriser au mieux le delais de transmission maximum, et donc d'utiliser *sommet-ajout* afin de minimiser le diamètre de la structure. Une des perspectives de notre travail est d'avoir une réelle approche multi-critère, dans le but d'obtenir, pour le diamètre et le poids, un algorithme aux performances intermédiaires, comme ce qui a déjà été fait, dans le cas purement statique, dans [Laf02].

Nous soulignons le fait que *sommet-ajout* est une restriction de l'algorithme *median-ajout* que nous avons introduit dans [TL03]. Ce dernier prend en compte la possibilité d'avoir un groupe de départ M_0 avec $|M_0| > 1$, ce qui permet, dans de nombreux cas, d'améliorer de manière très significative le rapport de compétitivité pour la somme des distances (constant sous certaines hypothèses).

Références

[BEY98] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University press, 1998.

[IW91] M. Imase and B.M. Waxman. Dynamic steiner tree problem. *SIAM J. Discr. Math.*, 4(3):369–384, 1991.

[Laf02] C. Laforest. A good balance between weight and distances for multipoint trees. In *OPODIS*, pages 195–204, 2002.

[TL03] N. Thibault and C. Laforest. Algorithme incrémental pour le maintien d'un arbre de connexion. In *RenPar'15*, pages 27–34, 2003.

[TL04] N. Thibault and C. Laforest. Dynamic maintenance of a tree with guarantees of performance. In *Rapport de recherche n° 93, Laboratoire de Méthodes Informatiques, Évry*, 2004.