CentraleSupélec

Algorithmique avancée

Épreuve pour le bloc 2 : algorithmes *online*

Le concours consiste à concevoir et à implémenter des algorithmes *online* (un déterministe **et** un randomisé) pour un problème d'allocation de ressources et calculer expérimentalement son ratio de compétitivité.

Le problème à traiter est le ***clustering* dynamique** ayant de nombreuses applications pratiques, par exemple l'installation des VMs qui communiquent souvent entre elles sur un même serveur physique afin de limiter la latence due à la distribution de calculs. Il peut être décrit d'une manière générale comme suit :

les nœuds communiquent en s'envoyant des requêtes $\sigma_k(i,j)$, $i,j \in [\![0, n-1]\!]$, où n est le nombre des nœuds. Les requêtes sont traitées d'une manière séquentielle, une par une, $\sigma = (\sigma_\ell(i,j))$, $\ell = 1, 2, \ldots, m$, où m est la longueur de la série des envois que les nœuds réalisent.

L'ensemble des nœuds est partitionné en *clusters*. Le coût de transmission d'un message entre deux nœuds d'un même *cluster* est nul, tandis que le coût de transmission entre deux nœuds se trouvant dans des *clusters* différents est unitaire.

La configuration des *clusters* peut être recâblée au prix α, $\alpha > 1$, par lien, c.-à-d. pour un échange de deux nœuds des *clusters* il faut payer 2α.

L'objectif est d'assurer la transmission de toutes les requêtes de la séquence σ avec le coût le plus petit possible.

La précision de l'instance qui nous occupe :

Noeuds : Le nombre des noeuds $\{v_0, v_1, \ldots, v_{n-1}\}$ est pair,

Topologie : Les nœuds sont liés en anneau ; ils sont connectés suivant l'ordre de leurs indices, $(v_0, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_0)$,

***Clusters* :** Il y a deux *clusters*, $k = 2$; un *cluster* est constitué de $\frac{n}{2}$ nœuds consécutifs sur l'anneau ; la configuration initiale des deux *clusters* c_0 est quelconque,

Requêtes : Les seules requêtes autorisées sont des envois entre deux nœuds consécutifs sur l'anneau $\sigma_\ell(i, (i+1) \mod n)$ (les messages circulent sur l'anneau dans un seul sens),

Coût de modification des *clusters* : Il faut tenir compte du décalage entre une configuration de *clusters* courante c_i et une configuration de *clusters* cible c_j exprimé en nombre de liens sur l'anneau séparant ces deux configurations ; ce décalage peut être réalisé dans le sens d'orientation de l'anneau ou dans le sens inverse (la valeur plus petite des deux est prise comme la distance entre c_i et c_i) ; le coût total de recâblage est $2\alpha \cdot \mathrm{dist}(c_i, c_j)$.

L'exercice se compose de :

1. Implémenter l'algorithme Lazy qui ne change jamais de configuration initiale des deux *clusters*.
2. Implémenter un algorithme Greedy qui pour chaque requête cherche à trouver une configuration des *clusters* à coût de transmission nul (à condition qu'elle existe), qui a engendré le moins transmissions *inter-cluster* jusqu'alors.
3. Proposer et implémenter un algorithme *online* déterministe.
4. Proposer et implémenter un algorithme *online* randomisé.

La gauge de qualité des algorithmes *online* est le résultat de l'algorithme *offline* qui connaît toute la séquence σ au préalable. Son idée est d'établir la meilleure configuration de l'anneau avant de traiter les transitions :

1. Depuis l'histogramme, calculer la probabilité qu'une configuration c_i provoque des conflits, $p(c_i)$.
2. Trouver une coupe c_i qui assurera un bon compromis entre le nombre de conflits $mp(c_i)$ et le coût de recâblage $2\alpha \cdot \mathrm{dist}(c_0, c_i)$:
$$\operatorname*{argmin}_{c_i} mp(c_i) + 2\alpha \cdot \mathrm{dist}(c_0, c_i).$$

N.B. Cet algorithme, étant probabiliste, produit une bonne solution avec une très grande probabilité.