Algorithmique et architectures parallèles
Partiel

E. Agullo C. Tedeschi Y. Robert

December 24, 2006

1 Topologie en m-star

On trouve dans la littérature sous l'appelation *m-star graph* une topologie initialement développée comme une alternative à l'hypercube et définie de la façon suivante. Une m-star est un graphe symétrique $G = (V, E)$ tel que $|V| = m!$, chaque sommet représentant une permutation distincte de m éléments, et E l'ensemble des arcs tels que deux permutations (nœuds) distinctes sont connectées ssi on peut passer de l'une à l'autre en interchangeant son premier élément avec n'importe quel autre. Par exemple, dans une 3-star, le nœud labellé par 123 est relié aux nœuds 213 et 321.

Remarquons qu'à l'instar des hypercubes, on peut construire cette structure de façon récursive. En effet, une m-star est composée de m $(m-1)$-stars disjointes. La figure 1 est une ébauche de 4-star.

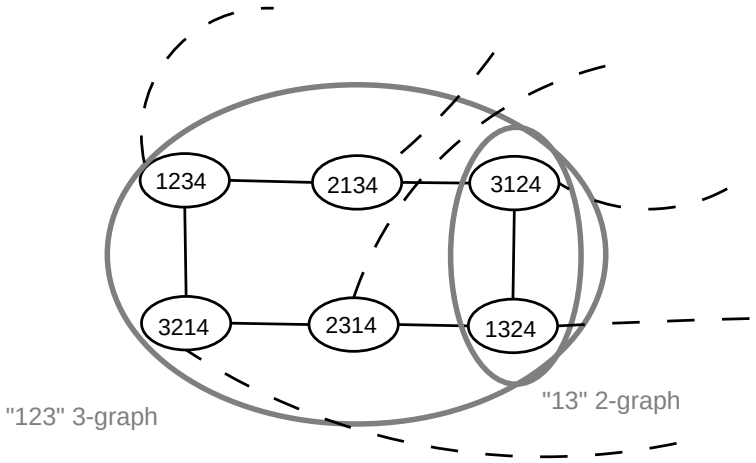

Figure 1: 4-star partielle

Question 1. *Dessinez une 4-star complète en expliquant votre méthode.*

Question 2. *Montrez qu'une m-star est un graphe $(m-1)$-régulier avec $|E| = m!(m-1)/2$.*

Question 3. *Proposez un algorithme de routage d'un point à un autre dans une m-star et donnez sa complexité.*

Question 4. *Proposez un algorithme de diffusion depuis un nœud quelconque dans une m-star et donnez sa complexité.*

2 Plongement d'un anneau

Soit $G = (V, E)$ un réseau d'interconnexion arbitraire (mais connexe) à n sommets. On veut le configurer en anneau, i.e. plonger un cycle à n sommets dans le réseau. On utilise l'algorithme suivant:

- Construire un arbre couvrant de G.
- Partitionner les nœuds de l'arbre en sommets pairs (de niveau pair dans l'arbre) et impairs (de niveau impair dans l'arbre).
- Parcourir l'arbre en profondeur, ajouter un sommet pair à l'anneau si c'est sa première visite dans le parcours, et ajouter un sommet impair si c'est sa dernière visite dans le parcours.

Question 5. *Traitez un petit exemple.*

Question 6. *Dilatation: quelle est la distance maximale dans le réseau de deux sommets consécutifs dans l'anneau?*

Question 7. *Partage: combien de fois un lien donné du réseau est-il partagé dans l'anneau?*

3 Ordonnancement sur m-star

Reprenons la topologie étudiée dans l'exercice 1 et faisons maintenant abstraction de la numérotation des nœuds. On considère une m-star comme un ensemble de processeurs numérotés de 0 à $m! - 1$. On peut découper une m-star en un nombre quelconque d'intervalles contigus, tant que les intervalles sont des i-stars valides, avec $1 \leq i \leq m$. On notera $[a, b]$ l'intervalle des processeurs contigus numérotés de a à b (avec $a < b$) formant une i-star valide. Pour un l quelconque, une m-star peut être divisée en l intervalles consécutifs $[a_1, b_1] \dots [a_l, b_l]$ où $a_1 = 0$ et $b_l = m! - 1$.

On souhaite maintenant ordonnancer n tâches indépendantes $J = (J_1, J_2, \dots, J_n)$ avec $J_i = (d_i, t_i)$ sur une m-star. Cela signifie que J_i doit s'exécuter sur une d_i-star (donc sur $d_i!$ processeurs) pendant t_i unités de temps. t_i est rationnel. On a toujours $1 \leq d_i \leq m$. On suppose de plus que ces tâches sont préemptives. Autrement dit, elles peuvent être arrêter au cours de leur exécution et reprendre plus tard là où elles s'étaient arrêtés, éventuellement sur un autre ensemble de processeurs, mais toujours de dimension d_i. On cherche à écrire un algorithme qui décide s'il est possible d'ordonnancer l'ensemble des tâches J de façon à ce que leur exécution soit finie avant une date butoir T.

Question 8. *Montrez d'abord que pour que l'ordonnancement soit valide, on doit avoir* $\forall i, 1 \leq i \leq n$: $t_i \leq T$ *et* $\frac{1}{m!} \sum_{i=1}^{n} t_i d_i! \leq T$.

On cherchera à ordonnancer les tâches (non ordonnées) une par une. On notera à chaque pas (après chaque tâche i ordonnancée) l'état de la plate-forme de la façon suivante:

$$S_{i-1} = ([a_1, b_1], r_1), ([a_2, b_2], r_2), \dots, ([a_k, b_k], r_k)$$

où $([a_j, b_j], r_j)$ signifie que pour la suite contiguë de processeurs $[a_j, b_j]$, le temps de calcul disponible restant est de r_j, autrement dit l'ordonnancement des $i - 1$ premières tâches necessite $T - r_j$ sur $[a_j, b_j]$. L'état initial est $S_0 = ([0, m! - 1], T)$. On pensera à ne garder que les ensembles de processeurs pour lesquels $r \neq 0$ et à trier les $([a_j, b_j], r_j)$ par r_j croissant.

Question 9. *En utilisant les indications précédentes, écrivez un algorithme qui décide s'il existe un ordonnancement optimal pour l'ensemble J, c'est-à-dire se termiant au plus tard à l'instant T. Donnez une idée de la preuve de correction de votre algorithme et sa complexité.*

Question 10. *Illustrez votre algorithme sur l'exemple suivant: 5-star,* $T = 4$, $J = (J_1, J_2, J_3, J_4, J_5, J_6)$ *avec* $J_1 = (4, 2)$, $J_2 = (4, 4)$, $J_3 = (4, 3)$, $J_4 = (3, 3)$, $J_5 = (3, 3.5)$ *et* $J_6 = (1, 4)$.