Senkaya Mikrail

Waldner Céline

Gestion distribué des ressources

Exercice 1 : Allocation distribuée des ressources –cas de la 3-Coloration d’un anneau

Le but de cet exercice est de réalisé une 3-Coloration. Cela signifie que l’on souhaite attribuer des couleurs parmi 3 possibles de telle sorte que deux nœuds voisins aient des couleurs différentes. Nous avons généralisé à K couleurs mais tous nos exemples seront donnés pour K=3.

Pour cet algorithme nous aurons besoin d’utiliser une synchronisation en étoile ouverte, nous avons besoin de vérifier le label de nos voisins mais nous ne changeons que notre propre label.

Au départ tous les labels sont à l’état N, nous n’avons pas besoin que l’une des couleurs souhaitées à la fin soit présente au démarrage. Nous avons choisi le graphe suivant pour tester notre algorithme :

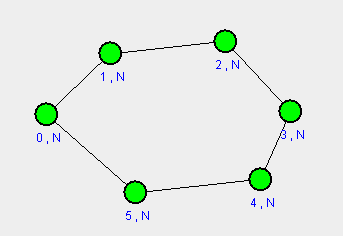


Figure 1 : Graphe en anneau

Pour appliquer la 3-Coloration nous devons vérifier plusieurs cas. Le premier est lorsque le nœud où l’on se place est de la même couleur que ces voisins, il doit choisir une autre couleur. Pour cela, nous avons créé un tableau comportant les différentes couleurs possibles et qui ne contient plus celle de ses voisins. Nous attribuons aléatoirement une couleur parmi celles restantes. Par exemple dans un cas de 3-Coloration (rouge, vert, bleu), mes voisins et moi sommes de couleur rouge, les possibilités seront alors : vert ou bleu.

Ensuite le cas où le nœud est de la même couleur qu’un seul de ses voisins. Ici nous excluons également de nos possibilité les couleurs des différents voisins. De même que précédemment nous choisissons une couleur parmi celle restante. En reprenant notre exemple précédent, mon voisin et moi sommes rouge mais mon autre voisin est bleu, la seule possibilité est alors vert.

Dans le dernier cas, si le nœud et ses voisins sont de couleurs différentes, une couleur est attribuée aléatoirement parmi les couleurs possibles ne contenant pas celles de ses voisins. Ce cas nous permet de gérer le cas où le label du nœud est encore à N.

Lorsqu’une nouvelle couleur est attribuée au nœud, son label va alors changer en fonction de la couleur assignée. Par exemple dans notre exemple précédent, le label sera changé par R, V ou B.

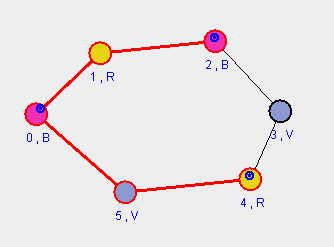


Figure 2 : résultat de notre algorithme de 3-Coloration

Exercice 2 : Portée de la terminaison ou de la stabilité

Le but de cet exercice est d’étudier la portée de la terminaison dans un algorithme SSP (Szymansky, Shi, Prywes). Il s’agit de calculer jusqu’à quelle distance la propriété de mise à jour est vérifiée.

Nous aurons besoin d’un *booléen p* initialisé à **false** en variable locale et d’un *compteur a* à -1 en propriété locale. Chaque nœud à un *compteur interne n* initialisé aléatoirement qui va définir la fin de la récupération de la mise à jour. Nous afficherons continuellement la valeur du compteur *a*.

Pour chaque nœud, à chaque synchronisation, nous décrémentons *n* tant qu’il est supérieur à 0. Lorsque le compteur arrive à 0, la valeur de *p* passe à **true**, et *a* passe à 0 et nous décrémentons une dernière fois *n*.   
Ensuite si *n* est inférieur à 0, nous prenons la valeur minimale entre le *a* du nœud et les *a* des voisins. Nous ajoutons 1 à cette valeur et remplaçons le *a* local par cette nouvelle valeur.

Si nous gardons l’algorithme tel qu’il est, les compteurs s’incrémenteront à l’infinie. Il faut donc déterminer quand appliquer la terminaison.

Nous commencerons par la terminaison locale. Nous avons choisi d’arrêter chaque nœud lorsque son compteur *a* vaut 5.

Comme les compteurs *n* de chaque nœud est différent, la synchronisation se fera différemment à chaque simulation, nous n’aurons donc jamais le même résultat.   
Si nous prenons l’exemple suivant, nous pouvons voir que le compteur *a* de certains nœuds n’atteindront jamais 5.

Nous pouvons également observer cela sur la simulation suivante :

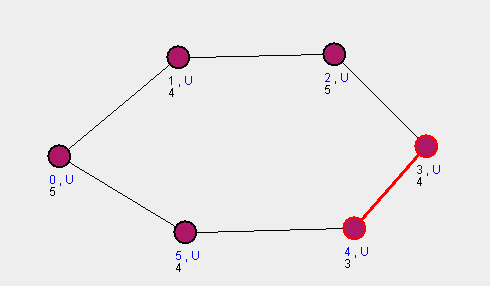


Figure 3 : Terminaison local avec un anneau

Maintenant, si au lieu de faire une terminaison locale, nous faisons une terminaison globale, le premier nœud dont le compteur *a* arrive à 5 sera le seul et la synchronisation s’arrêtera.

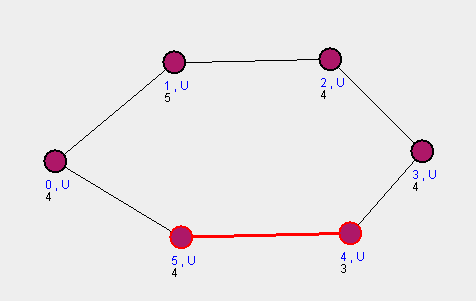


Figure 4 : Terminaison Globale avec un anneau

TD/TP : Algorithmes auto-stabilisants & pallier la perte d’une information locale

Exercice 1 : Algorithmes auto-stabilisants

Le but de cet exercice est de stabiliser les communications entre les différents nœuds d’un graphe afin que l’information se synchronise dans un ordre spécifique. Nous avons choisi d’utiliser une synchronisation locale en étoile ouverte. Nous utilisons le graphe suivant pour nos résultats :

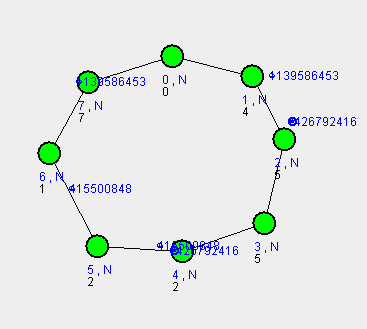


Figure : Graphe en anneau de 8 nœuds

Nous avons tout d’abord défini une propriété locale