Kyliann Fontaine
Evan Bouvier
Victor Boutros
Fatih-Hussein Kaya

Groupe 1B2

S2.02 -
Analyse des algorithmes d’affectation

Table des matière

Vannes
Université Bretagne Sud

I - Description des fonctionnement

A - Fonctionnement de l’algorithme exhaustif

Auteur de l’algorithme : Victor Boutros

1 - Principe

Le principe de l'algorithme est simple, tout testé jusqu’à trouver la solution la plus optimisée.

2 - Fonctionnement

1. Pour chaque combinaison, teste toutes les affectations possibles
2. Calcule coût et compétences couvertes, sélectionne selon critères prioritaires

3 - Exemple

Soit un Secouriste R1 avec la compétence S1 et un autre secouriste R2 avec la compétence S2. Et soit un DPS ayant les besoins suivants : 1 S1 et 1 S2.
Alors l’algorithme vas testé toutes les combinaisons possibles soit 2 combinaisons.
Combinaison 1 :
R1 -> S1
R2 -> S2
Combinaison 2 :
R2 -> S1
R1 -> S2
L’algorithme choisira la meilleure combinaison soit la combinaison 1.

B - Fonctionnement de l’algorithme glouton

Auteur de l’algorithme : Evan Bouvier

1 - Principe

L'algorithme glouton fonctionne selon le principe de choix localement optimal :
- À chaque étape, il fait le meilleur choix possible selon un critère de sélection
- Bien qu'il ne garantisse pas une solution optimale globale, il fournit une solution sous-optimale en temps polynomial

2 - Algorithme de tri utilisé

L'algorithme utilise un tri rapide hybride avec deux approches :
a) Tri rapide pour les grandes parties (> 10 éléments)
- Utilise la méthode "pivot médian-de-trois"
- Évite les cas dégénérés du quicksort classique

b) Tri par insertion pour les petites parties (≤ 10 éléments)
- Plus efficace que quicksort sur de petites tailles
- Réduit la surcharge de récursion
- Complexité : $O(n^2)$ mais très rapide pour $n < 10$

Critères de tri

Les éléments sont triés selon le nombre de correspondances.
- Secouristes : triés par nombre de compétences possédées
- Compétences : triées par nombre de secouristes qui les possèdent

3 - Critères de choix glouton

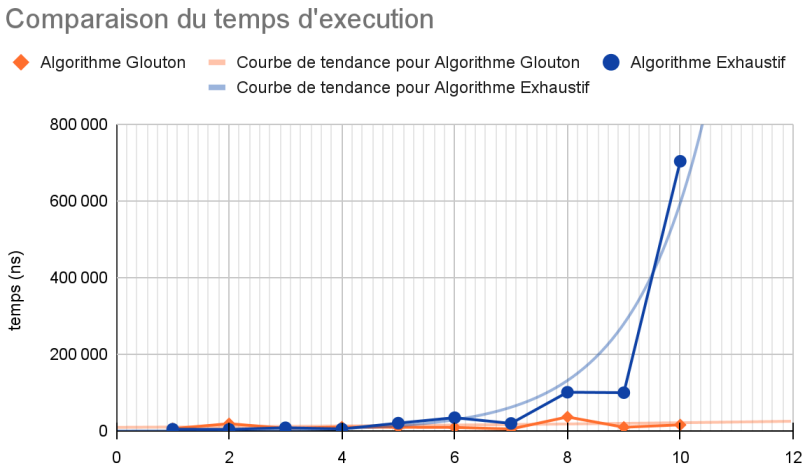
1. Ordre des secouristes : Les plus polyvalents d'abord (après tri décroissant)
2. Ordre des compétences : Les plus rares d'abord (parcours inverse du tri)

Pseudo-code :

```
Pour chaque secouriste i (du plus compétent au moins compétent) :
        Pour chaque compétence j (en partant de la compétence la moins “rare”) :
                Si le secouriste i possède la compétence j ET ni le secouriste ni la
                compétence ne sont déjà affectés :
                        → Créer l'affectation (secouriste i, compétence j)
                        → Marquer les deux comme affectés
                        → Passer au secouriste suivant
```

II - Analyse des deux algorithme

A - Analyse de temps



Comme nous pouvons voir grâce à ce graphique qui montre l'évolution du temps d'exécution en nanoseconde (ainsi que leurs courbes de tendance sur un modèle exponentiel) en fonction du nombre de besoin/secouriste disponible, que le temps d'exécution de l'algorithme exhaustif augmente extrêmement rapidement contrairement à l'algorithme glouton.

B - Analyse de résultat

Pour l'analyse des résultats nous exécutons 5000 fois les fonctions sur une matrice aléatoire de 9 par 9 case et nous prendrons la moyenne du pourcentage de compétence couverte pour le DPS généré.
Nous avons donce ses résultat :

```
PerformanceTest.testQualityResolution()-> Average Results for N=9 over 5000 iterations:
  Average Glouton Coverage Ratio: 0.7488888888888754
  Average Exhaustive Coverage Ratio: 0.7591555555555405
```

Comme nous pouvons les voir (supposant qu'il n'y est pas d'erreur de précision trop importante). Glouton ne réussit pas à couvrir en moyenne autant que l'exhaustif.

III - Conclusion

Nous pouvons donc conclure que bien que la précision de l'algorithme exhaustif la différence avec l'algorithme glouton est assez moindre pour la qualité de résultat.
De plus l'algorithme glouton à beaucoup moins de temps d'exécution que l'algorithme exhaustif. Par conséquent, l'algorithme glouton sera celui qui sera implémenté pour gérer les affectation automatique d'un DPS.
Cependant, l'algorithme glouton utilise de la récursivité qui pourrait lancer une exception dû au manque de mémoire.