Université Nationale du Vietname

Institut Francophone International

M1 Informatique : Systèmes Intelligents et Multimédia

Module : Recherche Opérationnelle

Rapport Mini-Projet :

PROBLÈME D'AFFECTATION DE TÂCHES A l'aide de L'ALGORITHME DE FORD-FULKERSON Dans un GRAPHE BIPARTI

Etudiant :
CIBAMBO M. STEVEN

Enseignant :
Dr. Hoang-Thach NGUYEN

Version du
23 novembre 2019

Résumé : Actuellement plusieurs organisations sont l'objet de problème de distribution des tâches dû au fait que la plupart des membres en leur sein ont des compétences variées et qu'il devient difficile de gérer la pertinence de leur préférence pour l'exécution d'une telle ou autre tâche. Afin de régler ce problème le chef d'équipe procède à assigner à chacun la tâche qu'il préfère. Pourtant à l'heure actuelle il existe de méthodes bien définies qui peuvent résoudre ce problème en peu de temps et avec plus de satisfaction. L'objectif de cette étude est d'implémenter l'une de ces méthodes afin de répondre au problème lié à l'affectation de tâches. La problématique est par conséquent la suivante ; Parmi les membres du groupe ou de l'organisation ; à qui affecter une ou telle autre tâche ? La méthode utilisée tient compte de la liste de tâches que chaque concerné souhaite exécuter ; et ce faisant, chacun se voit attribuer une tâche de son souhait.

Mots clés : Recherche opérationnelle, Aide à la décision, Algorithme et Complexité, Ford-Fulkerson, Graphe biparti.

ABSTRACT : Currently several organizations are subject to problem of distribution of tasks due to the fact that most of the members within them have varied competences and it becomes difficult to manage the relevance of their preference for the execution of such or other task. In order to solve this problem, the team leader proceeds to assign to each the task he prefers. Yet at present there are well-defined methods that can solve this problem in a short time and with more satisfaction. The purpose of this study is to implement one of these methods to address the problem of assigning tasks. The problem is therefore the following ; Among the members of the group or organization ; to whom to assign one or another task ? The method used takes into account the list of tasks that each concerned person wishes to execute ; and in doing so, each one is assigned a task of his wish.
Keywords : Operational Research, Decision Support, Algorithm and Complexity, Ford-Fulkerson, Bipartite Graph

Table des matières

Table des figures

1 Introduction

1.1 Contexte

Dans le cadre du module de la Recherche Opérationnelle, il est prévu que la partie théorique soit suivie d'une partie pratique réaliser par les étudiants afin de s'assurer que ces derniers ont bien assimilé leur cours. Deux à trois Étudiants se mettent ensemble afin de proposer une solution à un problème donné en se basant sur les contenus du cours. Ce présent rapport décrit la solution proposée à un problème d'affectation de tâches à l'aide de l'algorithme de Ford-Forlkerson utilisant un BFS (Breadth First Search) dans un graphe Biparti.

1.2 Domaine d'étude

L'idée sur laquelle se base ce travail est la résolution de problèmes plus précisement l'aide à la décision. L'aide à la décision fait appel aux outils de l'optimisation, la planification, la simulation, la gestion de la production et des opérations, la programmation et aux outils probabilistes et statistiques [1].

1.3 Objectif

Comme nous venons de le dire dans l'introduction ; l'objectif principal de ce travail est d'implémenter l'algorithme de Ford-Forlkerson utilisant un BFS1 avec le langage de programmation java permettant de résoudre le problème d'affectation de tâches au sein d'un groupe de personnes souhaitant effectuer au plus une tâche par individu.

2 L'Algorithme de Ford-Folkerson

Par définition ; l'algorithme de Ford-Folkerson est un algorithme utilisé pour les problèmes de flot maximum [2]. Considérant un graphe G(X,U) orienté, valué, connexe, anti-symétrique, sans circuit A chaque arc u on associe deux scalaires ;

C_u capacité de l'arc u tel que $C_u \geq 0$
ϕ_u le flot circulant sur u tel que $\phi_u \leq C_u$

2.1 Principe

A tout moment, la loi de Kirchhoff doit être vérifiée sur chaque sommet x de G

$$\sum_{u \in \omega^+} \phi_u = \sum_{u \in \omega^-} \phi_u$$

Le but d'augmenter le flot jusqu'à son maximum tout en respectant cette règle.

2.2 Exemple

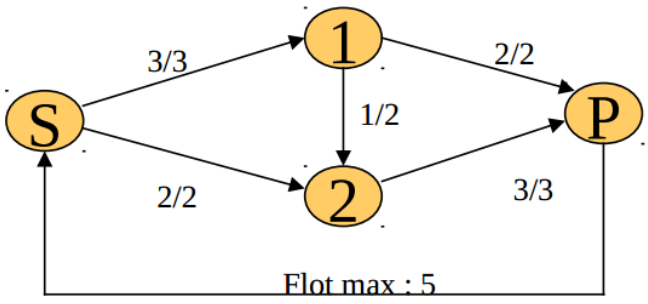

FIGURE 1 – Graphe orienté

Principe général de l'algorithme de Ford-Fulkerson :
— On part d'un flot compatible (généralement 0)
— On utilise deux fonctions alternativement :

1. **Procédure de marquage :** Le but est de trouver une chaîne améliorante. Son principe est de marquer les sommets selon deux critères : Delta (flot max que l'on peut faire parvenir au sommet) et Sommet de provenance

2. **Procédure d'augmentation du flot :** le but est d'augmenter le flot dans le graphe selon la valeur et le marquage obtenu par la procédure de marquage. Son principe est de parcourir le graphe du puit vers la source suivant les indications de provenance de la procédure de marquage

3 L'Algorithme de BFS

L'algorithme de parcours en largeur (ou BFS, pour Breadth First Search en anglais) permet le parcours d'un graphe ou d'un arbre de la manière suivante : on commence par explorer un nœud source, puis ses successeurs, puis les successeurs non explorés des successeurs, etc [3]. L'algorithme de parcours en largeur permet de calculer les distances de tous les nœuds depuis un nœud source dans un graphe non pondéré (orienté ou non orienté). Il peut aussi servir à déterminer si un graphe non orienté est connexe [1]

3.1 Principe

Cet algorithme diffère de l'algorithme de parcours en profondeur par le fait que, à partir d'un nœud source S, il liste d'abord les voisins de S pour ensuite les explorer un par un. Ce mode de fonctionnement utilise donc une file [2] dans laquelle il prend le premier sommet et place en dernier ses voisins non encore explorés.

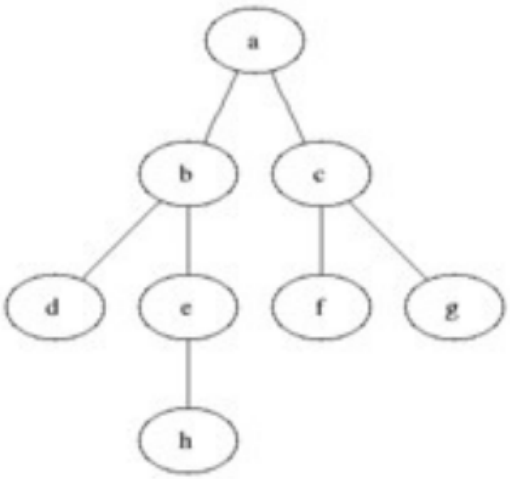

FIGURE 2 – Parcours d'un graphe en largeur

Les noeuds déjà visités sont marqués afin d'éviter qu'un même noeud soit exploré plusieurs fois. Dans le cas particulier d'un arbre, le marquage n'est pas

1. En théorie de graphe, un graphe non orienté est dit connexe s'il est d'un seul tenant.
2. En informatique, une file dite aussi file d'attente, est une structure de données basée sur le principe du premier entré, premier sorti

nécessaire.

Etapes de l'algorithme :

1. Mettre le noeud source dans la file
2. Retirer le noeud du début de la file pour le traiter.
3. Mettre tous les voisins non explorés dans la file (à la fin)
4. Si la file n'est pas vide reprendre à l'étape 2.

Note : l'utilisation d'une pile au lieu d'une file transforme l'algorithme du parcours en largeur en l'algorithme de parcours en profondeur.

3.2 Exemple

Sur le graphe suivant, cet algorithme va alors fonctionner ainsi :

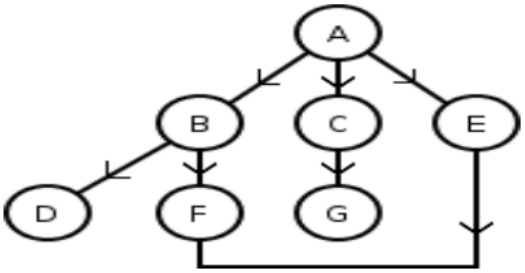

FIGURE 3 – Exemple de graphe pour l'algorithme DFS et BFS

Il explore dans l'ordre les sommets A, B, C, E, D, F, G, contrairement à l'algorithme de parcours en profondeur qui cherche dans cet ordre : A, B, D, F, E, C, G.

4 Implémentation

4.1 Architecture

— **Graphe biparti :** Au démarrage du programme, celui-ci prend paramètre un fichier représentant les liaisons possible entre les éléments de deux parties principales d'un graphe biparti. En théorie de graphe, un graphe est dit biparti s'il existe une partition de son ensemble de sommets en deux sous-ensembles et elle que chaque arête ait une extrémité dans et l'autre dans.

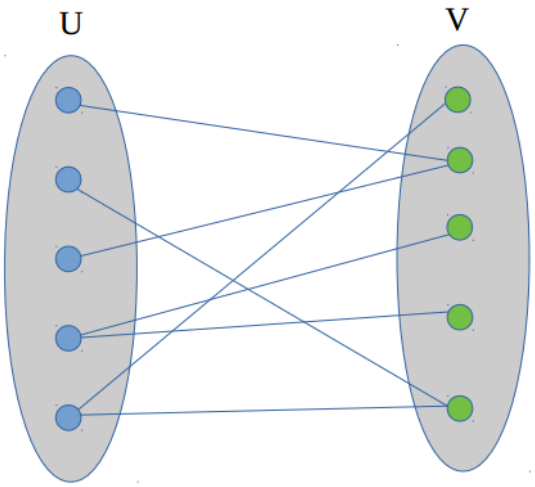

FIGURE 4 – Graphe biparti quelconque

Étant donné que ; les nœuds du vecteur U sont considérés comme les successeurs du nœud source S et le nœud puit et le successeur des nœuds du vecteur V, cela n'est pas nécessaire d'être représenté. Le programme considère que, les nombres de nœuds de deux parties sont égaux. La première ligne du fichier donné en paramètre indique le nombre de nœuds d'un vecteur. Le reste de lignes, représente la correspondance de chaque nœud du vecteur U vers un ou plusieurs nœuds du vecteur V.

— Les classes

1. Noeud.java
2. Couplage.java
3. Graphe.java

Ces trois classes sont constituées de différentes méthodes pour la mise en fonctionne de quelques fonctionnalités.

4.2 Résultat Obtenu

L'objectif étant le couplage maximum de personnes à leurs tâches souhaitées. La figure 5 montre les contenus du fichier data.txt qui liste les tâches souhaité par

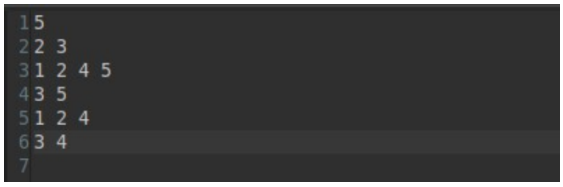

FIGURE 5 – Tâches souhaitées par personne

personne. La première ligne contient un chiffre représentant le nombre tâches qui bien sûr doit être égal au nombre de personne. Dans le cas présent 5 ; soit 5 personne 5 tâches. Les lignes suivantes contiennent les tâches de préférence de chaque personne ; soit la personne souhaite effectué la tâche 2 et 3, la deuxième personne souhaite la tâche 1, 2, 4 et 5 ; etc. La figure 6, illustre le résultat obtenu

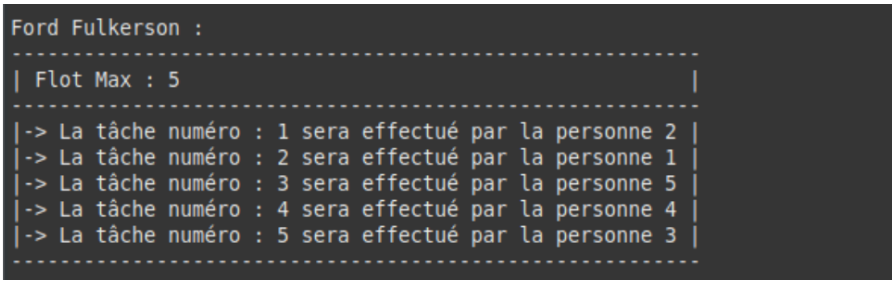

FIGURE 6 – Résultat obtenu

avec les données à la figure 5 et remarquons que chaque personne du groupe à obtenu une de ses tâches souhaité parmis tant d'autre tâches. L'objectif de l'algorithme de Ford-Fulkerson étant d'obtenir le flot maximum ; nous pouvons dire que le problème d'affectation de tâches est résolu avec le flot max qui égal à 5

— Graphe obtenu et le chemin BFS
La figure 7, représente les nœuds constituant notre graphe et le chemin BFS de la source 0 au puit 11. Les nœuds 6 à 10 représente les personnes et les nœuds 1 à 5 les tâches.

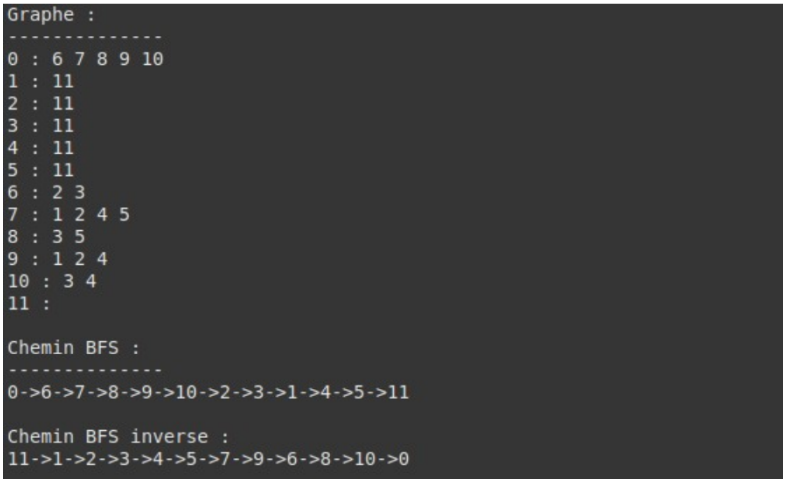

FIGURE 7 – Graphe généré,le chemin BFS et son inverse

5 Conclusion

Au terme de notre travail ; nous estimons avoir atteint l'objectif que nous nous sommes fixé à savoir la l'implémentation de l'algorithme de Ford-Fulkerson utilisant le BFS pour résoudre le problème d'affectation de tâche au sein d'un groupe de personnes. Tout au début nous avons commencé par une brève introduction de notre travail en parlant du contexte dans lequel il fait cela à était suivi de l'objectif visé. Dans la troisième partie et deuxième partie nous avons parlé de deux type d'algorithme auxquels nous avons fait recours à savoir l'algorithme de FordFulkeson et l'algorithme de Breadth First Search (BFS) et en fin nous avons montré les résultats obtenus et l'architecture du présent travail dans la partie implémentation.

Références

[1] Université de BORDEAUX : Recherche opérationnelle et aide à la décision, Novembre 2019. https ://math-interactions.u-bordeaux.fr/Formations/Master-Mathematiques-appliquees-et-statistique/Recherche-operationnelle-et-aide-a-la-decision.

[2] WIKIPEDIA : Algorithme de ford-fulkerson, Fevrier 2019. https ://fr.wikipedia.org/wiki/Algorithme$_d e_F ord - Fulkerson$.

[3] WIKIPEDIA : Algorithme de parcours en largeur, Fevrier 2019. https ://fr.wikipedia.org/wiki/Algorithme$_d e_p arcours_e n_l argeur$.

6 Annexe

6.1 Code source

L'ensemble du projet inclus toutes les dependances est sur ce lien ici