AVIGNON UNIVERSITÉ

Stage recherche opérationnelle

Jean-Christophe MOUNIER

12 juillet 2022

[1] Laboratoire Informatique d'Avignon – LIA

Licence d'Informatique
Ingénierie Logicielle
UE Stage

Encadrants du stage[1]
Serigne GUEYE

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Introduction

1.1 Présentation du LIA

Le Laboratoire Informatique d'Avignon est un laboratoire de recherche rattaché à l'Université d'Avignon et plus précisément au Centre d'Enseignement et de Recherche en Informatique (CERI). Il accueille des enseignants-chercheurs, des doctorants, des alternants et des stagiaires qui travaillent sur trois grandes thématiques que sont le Traitement Automatique du Langage Naturel (qu'il soit écrit ou oral), la Recherche Opérationnelle et les Réseaux.

1.2 Description de la mission

L'objectif à travers ce stage est de mettre au point une méthode pour prédire la probabilité qu'une équipe puisse battre une autre. Cette méthode se base uniquement sur des matchs entre deux équipes ou joueurs et leurs scores, le contexte peut être sportif ou non. En plus de la méthode de prédiction, on cherche également à déterminer un classement des équipes.
Pour réaliser un classement et une prédiction nos calculs s'appuieront sur la méthode Elo. Enfin pour avoir un rendu graphique nous utilisons la librairie Graphstream de Java.

Méthode de classement Elo

La méthode de classement Elo est un système d'évaluation comparant les joueurs en fonction de leurs probabilités de gagner. Cette méthode est utilisée pour des matchs en un contre un.

La méthode Elo se base sur la force relative de deux joueurs. La force relative de i contre j est dite directe lorsqu'elles est calculée depuis les rencontres entre i et j. On définit $P(A/B)$ la probabilité que le joueur A batte le joueur B. Ainsi la force relative de A contre B s'exprime comme ceci :

$$f_{AB} = \frac{P(A/B)}{1 - P(A/B)} \tag{1}$$

Grâce au calcul des forces relatives directes entre les joueurs il est possible d'estimer la force relative indirecte entre deux joueurs. On considère la force et les probabilités suivantes :

- $f(p)$ la force indirecte de A contre C
- $q = P(A/B)$ la probabilité que A gagne B
- $r = P(B/C)$ la probabilité que B gagne C

La force relative indirecte de A contre C s'exprime :

$$f(p) = f(q) \times f(r) \tag{2}$$

2 Travail réalisé

2.1 Préambule

Pour effectuer un classement et une prédiction, nous représentons le problème par un graphe $G = \{V,E\}$. Les sommets représentent les équipes (ou joueurs) et les arcs représentent les rapports de force. Dans ce graphe, nous gardons uniquement les arcs de poids fort, autrement dit les arcs de poids supérieur ou égal à 1.
La force relative directe de i contre j se calcule à partir de la probabilité que i batte j. Cette probabilité est déterminée grâce à l'ensemble des matchs k entre i et j. Cet "indice de performance" est noté P_{ij}^k.

2.2 Méthode proposée

Notre méthode est divisée en plusieurs étapes :

- Calcul des P_{ij}^k soit l'indice de performance entre i et j sur le match k
- Calcul de P_{ij} soit la probabilité que i soit plus fort que j
- Calcul de $f_{i,j}$ soit la force relative de i contre j
- Ajustement des poids des arcs (méthode de **Floyd-Warshall**)
- Établissement d'un classement

Calcul des P_{ij}^k

Afin de pouvoir calculer la probabilité que i soit plus fort que j, on commence par calculer laquelle des deux équipes (ou joueurs) a le mieux performé lors de chaque match k opposant i à j.

Cet indice de performance entre i et j lors du match k se calcule à l'aide de la formule suivante :

$$P_{ij}^k = \frac{1}{2} + \frac{min(d_{ij}^k, M)}{2M} \tag{3}$$

Où :

- d_{ij}^k est la différence du score entre i et j lors du match k.
- M est une borne que l'on fixe afin de limiter l'écart de score.

Calcul des P_{ij}

Une fois toutes les valeurs P_{ij}^k calculées, nous pouvons calculer la probabilité que i soit plus fort que j en se basant sur l'ensemble de leurs rencontres.
Cette probabilité s'exprime :

$$P_{ij} = \sum_{k=1}^{K} \lambda_k \times P_{ij}^k \tag{4}$$

Où :

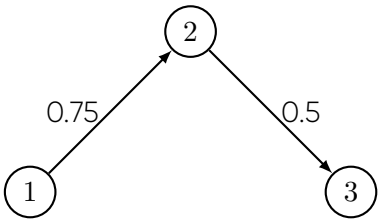
- K est le nombre total de match
- λ_k est une pondération déterminant l'importance du match

La nature de λ_k fait que nous avons :

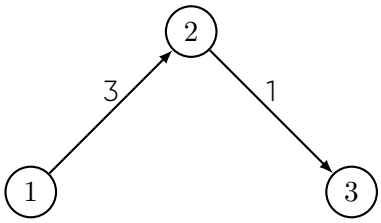
$$\sum_{k=1}^{K} \lambda_k = 1 \tag{5}$$

Transformation des probabilités en forces

L'objectif à travers la transformation des probabilités en forces est de rendre les résultats des calculs cohérents. Prenons le cas suivant :



(a) graphe pondéré par des probabilités



(b) graphe pondéré par des forces

Figure 1. Exemple de graphe

Remarque : Les poids sur les arêtes de la figure 1a correspondent aux valeurs des P_{ij}

Si on calcule P_{13} sur graphe 1a, on obtient :

$$P_{13} = P_{12} \times P_{23} = 0.75 \times 0.5 = 0.375 \tag{6}$$

Cependant si on observe les valeurs sur les arcs, on se rend compte que la probabilité que 1 batte 2 est de 0.75 et que la probabilité que 2 batte 3 est de 0.5 (autrement dit 2 et 3 ont le même niveau).
Puisque 2 est équivalent à 3, il semble logique de dire que 1 est aussi fort contre 2 que contre 3. On s'attend donc à avoir $P_{12} = P_{13}$, ce qui n'est pas le cas.

Il est donc important de rendre les résultats des calculs cohérents. Pour ce faire, on utilise les forces relatives[1]. En associant les valeurs des forces relatives f_{ij} aux poids des arcs et non plus les valeurs des probabilités P_{ij}, on obtient le graphe de la figure 1b. D'après la formule de calcul des force relative[2], on obtient : $f_{13} = f_{12} \times f_{23} = 3 \times 1 = 3$.
Cette fois on a bien $f_{13} = f_{12}$.

Les algorithmes sur les graphes sont généralement basés sur des sommes de poids (ex : calcul du plus court chemin). Cependant, les calculs effectués sur les probabilités et les forces se basent sur des produits. Pour se ramener à des calculs de somme, on utilise le logarithme base 10. On se retrouve avec l'équivalence suivante :

$$f_{AC} = f_{AB} \times f_{BC} \iff \log_{10}(f_{AC}) = \log_{10}(f_{AB}) + \log_{10}(f_{BC}) \tag{7}$$

Une fois les poids ajustés, on calcule le plus court chemin entre tous les noeuds du graphe à l'aide de l'algorithme de **Floyd-Warshall**. Ainsi la force sur les arcs correspondra à la force minimale d'une équipe contre une autre.

Classement

Grâce au graphe complété on peut établir un classement des équipes ou joueurs. Plusieurs méthodes ont été implémentées :

- Faire la somme des forces pour chaque équipe et les classer selon cette somme
- Calculer les composantes fortement connexes dans un graphe
- Faire la somme des estimations de force pour chaque équipe et les classer (3 variantes ont été implémentées)
- Utiliser l'algorithme PageRank (3 variantes ont été implémentées)

2.3 Implémentation

Pour implémenter notre méthode on utilise cinq classes :

- Main
- GraphePoids
- Poids
- FileManager
- Algo

(a) Menu dans la console **(b)** Arborescence des projets

Figure 2. Interface

Lors de l'exécution, on peut accéder aux différentes fonctionnalités du programme à travers un menu (2a).

La suppression et la sauvegarde des objets sauvegardés se font grâce aux méthodes de la classe FileManager. Pour chaque projet sauvegardé, on conserve deux fichiers (2b). Le fichier *graphePoids.ser* est un objet sérialisé, autrement dit une instance de la classe GraphePoids dont on souhaite conserver l'état. Le fichier *graph.dgs* correspond à une instance de la classe Graph de GraphStream.

Le but avec ces sauvegardes et de sauvegarder un graphe et de le modifier plus tard dans le temps. Ainsi, on peut faire évoluer un graphe au fur et à mesure de l'évolution d'un championnat par exemple.

La classe GraphPoids correspond à la structure de données qu'on a mis en place pour stocker les informations sur les matchs et les équipes. Son attribut principal est une matrice triangulaire, plus précisément une liste de tableau de Poids. Dans chaque Poids sont stockés la liste des P_{ij}^k[4] ainsi que la force en $\log_{10}$ calculée à partir de la liste. De ce fait, dans la structure pour chaque i et j appartenant au graphe, on a la liste des valeurs des P_{ij}^k et leur force relative f_{ij}.
De plus, chaque poids possède deux variables supplémentaires : une estimation de la force (voir les méthodes de classement) et la prédiction (chance de victoire en pourcentage pour la prochaine rencontre).

Puis en fonction des forces, on génère un graphe comportant uniquement les arcs de poids forts. Les arcs de poids fort sont les arcs ayant une force supérieure ou égale à 1.

Puisqu'on utilise des forces en $\log_{10}$, on ne conserve que les arcs de poids positifs.

Enfin tout les algorithmes de classement sont implémentés dans la classe Algo. Cette classe a pour unique attribut une instance de la classe GraphePoids. C'est sur cette instance que seront appliquer les différents algorithmes de classement.

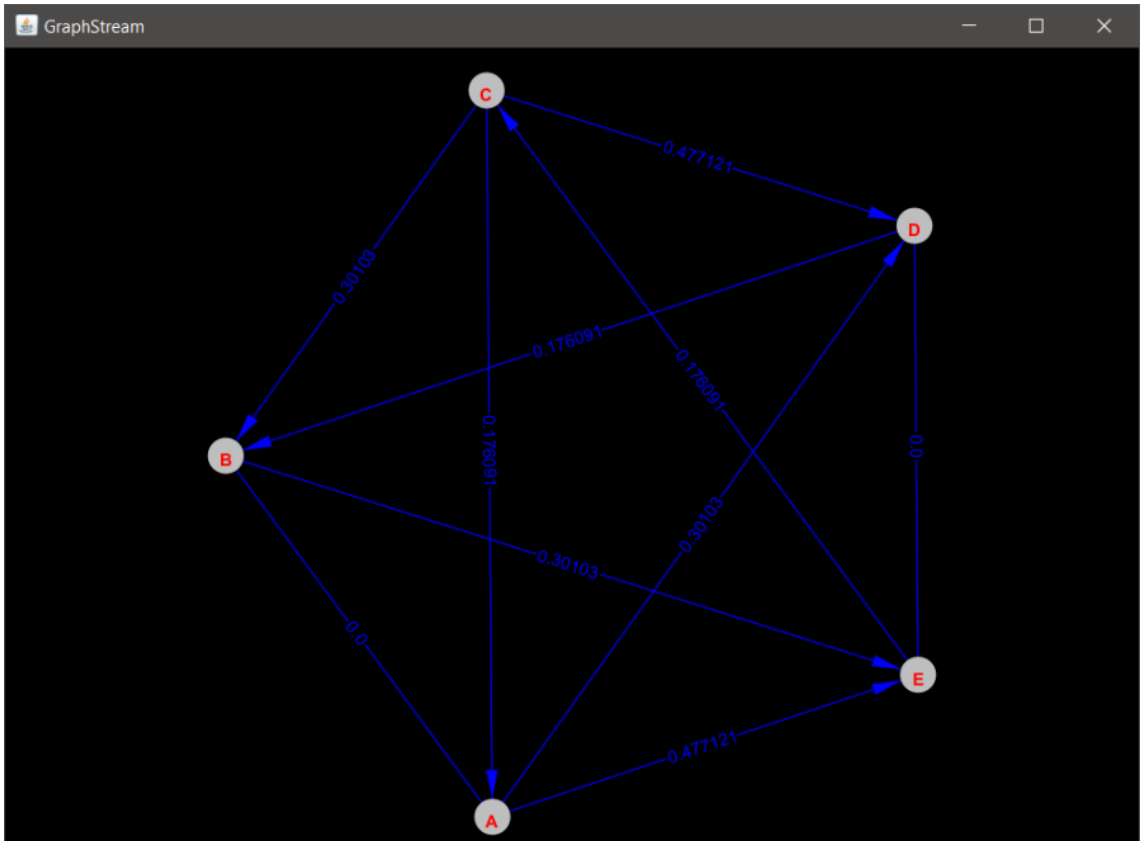


Figure 3. Exemple de graphe généré avec 5 équipes et 18 matchs

Remarque : Les poids utilisés ici sont en $log_{10}{}^{7}$ puisque l'algorithme de **Floyd-Warshall** de la librairie Graphstream calcule les plus courts chemins en les additionnant

La prochaine étape après la génération du graphe est de chercher un moyen pour classer les équipes. Cependant la présence de circuit pose problème :

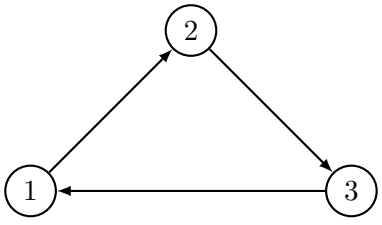


Figure 4. Exemple de circuit

Ici on observe que l'équipe 1 est directement plus forte que l'équipe 2. Cependant, l'équipe 2 est indirectement plus forte que l'équipe 1. Le problème qui se pose est de déterminer laquelle des 2 équipes est plus forte que l'autre. Plus généralement, la problématique est de savoir comment traiter les composantes fortement connexes dans un graphe.

Deux méthodes ont été envisagées. La première, consiste à traiter les composantes fortement connexes dans le graphe. La seconde, consiste à casser les circuits en prenant en compte les forces directes (f_{ij}) et indirectes (f_{ji}).

Première méthode

Ici il est question d'isoler les composantes fortement connexes et de les classer. On a implémenté un algorithme pour calculer ces composantes : pour un noeud i on récupère l'ensemble des noeuds descendants et ascendants et on fait l'intersection de ces deux ensembles.
Cependant, on a ajouté une modification afin d'avoir des résultats plus cohérents : on limite le nombre d'arc dans un circuit (pour ce faire on a utilisé un parcours en largeur).

Voici les résultats obtenus sur le graphe représentant le championnat de Ligue 1 lors de la saison 2021/2022 (20 équipes numérotées de 0 à 19, la numérotation correspond au classement final du championnat) :

```
Composantes fortement connexes sans limite de nombre d'arc:

    [[0, 1, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 3, 4]]

Avec limite d'arc:
En prenant en comptes les forces nulles:

-limite == 1:
[[0, 6, 8], [1, 3], [2, 7, 10], [4, 14, 15, 16], [5, 11], [9, 13, 17, 18], [12], [19]]

-limite == 2:
[[0, 1, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 2, 3, 4], [17, 18, 19]]

-limite == 3:
[[0, 1, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16], [2, 3, 4, 17, 18, 19]]

En prenant pas en compte les forces nulles:

-limite == 1:
Il y a autant de composante que de noeud dans le graphe.

-limite == 2:
[[0, 1, 5, 7, 9, 11, 12, 15, 2, 4, 6], [3, 10, 14, 18, 8], [13, 19, 16, 17]]

-limite == 3:
[[0, 1, 5, 7, 9, 11, 12, 15], [2, 3, 4, 13, 14, 16, 18], [6, 19], [8, 10], [17]]
```

Remarque : Un arc de poids nul est considéré comme une arête dans le graphe. Ainsi, si $f_{ij} = 0$ alors j est à la fois un noeud ascendant et descendant de i.

Seconde méthode

L'objectif est de casser les circuits. Pour y arriver, on calcule une estimation de la force relative de i contre j en prenant en compte la force directe et indirecte. Cette estimation est noté e_{ij}.

Concrètement on calcule le rapport de f_{ij} sur f_{ji} :

$$e_{ij} = \frac{f_{ij}}{f_{ji}} \tag{8}$$

Où :

- e_{ij} est l'estimation de la force relative de i contre j
- f_{ij} est l'arc de poids fort

f_{ji} étant une force indirecte on utilise la formule de calcul des forces indirectes[2]. On calcule la moyenne des forces des arcs du chemin de j à i.
Ainsi on a :

$$f_{ji} = \sum_{n=1}^{N} \frac{f_n}{N} \tag{9}$$

Où :

- N est le nombre total d'arc du chemin de j à i
- f_n est la force en log_{10} du n-ième arc du chemin de j à i

Soit un arc (i,j) de poids maximal dans un circuit. En calculant l'estimation de la force de i contre j, le rapport des forces f_{ij} sur fji donnera toujours une valeurs supérieure à 1.
Soit un arc (k,l) de poids minimal dans un circuit. Dans ce cas le calcul de e_{kl} donnera toujours une valeur inférieure à 1. Notre graphe ne comporte que des arcs de poids fort. En utilisant les estimations de force comme poids, l'arc de poids maximal gardera son orientation. Inversement l'arc de poids minimal changera d'orientation.
Le fait d'avoir ces deux propriétés nous garantit de casser des circuits dans des graphes.

Grâce à cette méthode on a pu concevoir un algorithme pour calculer les estimations de force. Néanmoins, il y a deux cas particuliers à prendre en compte. La formule ne fonctionne pas pour des circuits dont les poids sur les arcs sont identiques. Les arcs de poids nul ne doivent pas être pris en compte.

```
Soit un graphe G = {V,E}
Pour tout arc (i,j) dans V
   Si Fij == 0
      Eij = 0
   Sinon
      chemin = parcours largeur j -> i avec poids == Fij
      Si chemin != null
         Pour tout arc (k,l) dans chemin
            Ekl = 0
         Eij = 0

Pour tout arc (i,j) dans V
   chemin = parcours largeur j -> i avec poids > 0 et Eij != 0
   Si chemin != null
      Pour tout arc (k,l) dans chemin
         Ekl = Fkl/Flk
      Eij = Fij/Fji
```

Remarque : Les forces et estimations sont en $\log_{10}$

2.4 Méthodes de classement

À ce stade, nous sommes en capacité de procéder à un classement des équipes ou joueurs. Ce classement est calculé à partir d'un score. Les méthodes suivantes permettent de calculer un score pour chaque équipe.

Somme des forces

Pour chaque équipe, on calcule la somme de leurs forces. Le score d'une équipe i se calcule :

$$score_i = \sum_{j=1}^{N} f_{ij} \quad \forall\, j \neq i \tag{10}$$

Somme des estimations de forces

Faire la somme des forces prend uniquement en compte les forces relatives directes. On peut affiner les résultats en utilisant les estimations de force e_{ij}. Le score se calcule en utilisant ces estimations au lieu des forces directes :

$$score_i = \sum_{j=1}^{N} e_{ij} \quad \forall\, j \neq i \tag{11}$$

Somme des estimations de force (maximum)

L'estimation de la force de i contre j est égale au rapport de la force directe f_{ij} sur la force indirecte f_{ji}[8]. Cette force indirecte correspond à la moyenne des forces des arcs du chemin de j à i.
La variante étudiée ici consiste à prendre la valeur maximale de ces forces au lieu de la moyenne. La méthode de classement reste la même : on fait la somme des estimations de force.

Notons C, l'ensemble des arcs du chemin de j à i. La force indirecte s'exprime :

$$f_{ji} = \max(v) \implies score_i = \frac{f_{ij}}{max(v)} \quad \forall\, v \in C \tag{12}$$

Somme des estimations de force et des forces relatives directes

Nous avons vu dans l'algorithme de calcul des estimations de force, que parfois ces estimations étaient égales à 0. Lorsqu'on fait la somme de ces estimations, on somme donc des 0. Afin de prendre en compte un maximum d'information, on peut prendre la force f_{ij} lorsque e_{ij} est nulle.

Le score se calcule ainsi :

$$score_i = \sum_{j=1}^{N} \alpha_{ij} \quad \begin{cases} \alpha_{ij} = e_{ij} \; si \; e_{ij} \neq 0 \\ \alpha_{ij} = f_{ij} \; sinon \end{cases} \tag{13}$$

PageRank

Notre modélisation se base sur des graphes orientés. L'algorithme PageRank de Google permet d'attribuer un score à des pages en fonction des liens qu'elles ont entre elles. Les sommets du graphe représentent les pages et les arcs représentent les liens entre ces pages.

Cet algorithme prend en compte deux facteurs principaux :

- Le nombre d'arcs entrants dans une page p, ce qui correspond au demi-degré intérieur de p, noté $d^-(p)$
- Le score des pages, une page p aura un score plus élevé si les pages qui la citent ont un score important

Puisque les arcs dans nos graphes représentent des rapports de force, on peut utiliser l'algorithme PageRank pour classer les équipes. Le score de chaque équipe est calculé comme suit :

$$Pr(i) = \frac{(1-p)}{N} + p \times \sum_{j=1}^{N} \left(\frac{Pr(j)}{d^+(j)} \right) \quad \forall\, j \ tq \ (j,i) \in V \tag{14}$$

Où :

- $Pr(i)$ le score de la page i
- p est la probabilité que l'on change de page en passant par les liens
- $d^+(j)$ le demi-degré extérieur de j

Dans notre contexte, plus une équipe a un demi-degré intérieur élevé, plus son niveau est faible. Les scores calculés avec cette formule sont donc inversement proportionnels aux niveaux des équipes (ou joueurs). Avant d'effectuer un classement, on multiplie les scores par -1.

PageRank amélioré

L'algorithme PageRank est adapté aux graphes de navigation. Dans notre cas, on travaille sur des graphes de probabilités. La problématique est de savoir comment adapter cet algorithme à notre contexte. Nous avons vu que le score d'une page est lié à son demi-degré intérieur.

Notons μ_i ce score, il s'exprime :

$$\mu_i = d^-(i) \tag{15}$$

En considérant chaque arc (j,i) comme le vote de j pour i, l'équipe la plus forte sera celle avec le plus de vote. Cependant le vote d'une page pour une ou plusieurs pages n'a pas la même importance. En effet, si une page accorde son vote à toutes les autres pages, ses votes ne seront pas très informatifs.

Il est donc censé de relativiser l'importance du vote de j pour i par l'ensemble des votes de j, soit $d^+(j)$. La valeur d'un vote ne sera plus de 1 mais de $\frac{1}{d^+(j)}$.

μ_i s'exprime donc :

$$\mu_i = \sum_{j|(j,i)\in V} \frac{1}{d^+(j)} \tag{16}$$

Il est également nécessaire de prendre en compte le poids des pages : plus une page est importante, plus son vote l'est aussi. Au lieu de définir l'importance d'un vote d'une page j par $\frac{1}{d^+(j)}$, prenons son score, μ_j :

$$\mu_i = \sum_{j|(j,i)\in V} \frac{\mu_j}{d^+(j)} \tag{17}$$

Afin d'adapter ces calculs à notre contexte, on relativise l'importance du vote de j par la force totale de j au lieu d'utiliser $d^+(j)$[16]. De plus, nos arcs ont des poids, il faut également prendre en compte l'importance d'un vote par rapport à ce poids.

Pour simplifier les formules, notons F_j, la somme des poids pour tout arcs de sommet source j :

$$F_j = \sum_{k \mid (j,k) \in V} f_{jk} \tag{18}$$

Le calcul de μ_i devient :

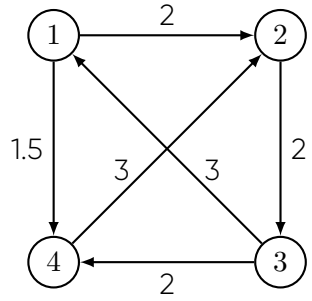
$$\mu_i = \sum_{j|(j,i)\in V} \frac{f_{ji} \times \mu_j}{F_j} \tag{19}$$

Ce calcul peut être traduit par un calcul matriciel. Prenons la matrice A définie par :

$$A_{ij} = \begin{cases} \frac{f_{ji}}{F_j} \text{ si } (j,i) \in V \\ 0 \text{ sinon} \end{cases} \tag{20}$$

Le calcul de μ_i est équivalent à :

$$\mu_i = \sum_{j=1}^{N} A_{ij} \times \mu_j \tag{21}$$



(a) graphe pondéré par des forces

A_{ij}	1	2	3	4
1	0	0	$\frac{3}{5}$	0
2	$\frac{2}{3,5}$	0	0	1
3	0	1	0	0
4	$\frac{1,5}{3,5}$	0	$\frac{2}{5}$	0

(b) matrice stochastique

Figure 5. Exemple

Considérons le graphe de la figure 5a. À partir de ce graphe, on peut déterminer la matrice de la figue 5b à l'aide de la formule 20.

Les valeurs dans la matrice sont basées sur les forces relatives directes. Cependant, nous utilisons des valeurs en log_{10} pour nos calculs. Prenons la troisième colonne, qui correspond au sommet 3, la force totale (notée F) est calculée avec la formule suivante (issue de la formule 18) :

$$F_3 = \sum_{k \mid (3,k) \in V} f_{3k} \tag{22}$$

$$F_3 = 2 + 3 = 5 \iff \log(F_3) = \log(2) + \log(3) = \log(2 \times 3) = \log(6)$$

Il est nécessaire de bien prendre en compte le passage en $\log_{10}$. Prenons la valeur $(1,3)$ dans la matrice, égale à $\frac{3}{5}$. En passant les valeurs directement en $\log_{10}$, nous n'aurions pas les bons résultats :

$$\frac{\log(3)}{\log(5)} \neq \frac{log(3)}{\log(6)}$$

Méthode basée sur PageRank

Cette méthode se base sur les principes fondamentaux de l'algorithme PageRank :

- Le nombre d'arcs entrants et sortants d'un sommet j, noté respectivement $d^-(j)$ et $d^+(j)$
- L'importance des sommets sources i, $\forall\ (i,j) \in E$

L'importance d'une équipe i est représentée par la moyenne de ses forces. Notons cette moyenne F_i :

$$F_i = \sum \frac{f_{ik}}{deg(i)}\ , \forall\ k\ tq\ (i,k), (k,i) \in E \tag{23}$$

Le concept lié à cette méthode, est de calculer un score grâce aux degrés d'un sommet :

$$score_i = \frac{d^+(i) - d^-(i)}{deg(i)} \tag{24}$$

Cependant, cette approche ne prend ni en compte les poids sur les arcs, ni la force globale des équipes[23]. Pour pallier ce problème, nous associons à chaque arc une valeur comprise entre 0 et 1. Pour un arc ayant i comme cible, cette valeur est notée ϕ_{ij}^-. Inversement, pour un arc ayant i comme source, on note cette valeur ϕ_{ij}^+.

Ces valeurs sont calculées de la manière suivante :

$$\phi_{ij}^- = \min(0, \frac{f_{ij} + F_j}{2 \times \log(4)})\ \forall\ f_{ij} < 0 \tag{25}$$

$$\phi_{ij}^+ = \max(0, \frac{f_{ij} + F_j}{2 \times \log(4)})\ \forall\ f_{ij} > 0 \tag{26}$$

Dans la démonstration qui suit, les forces sont bornées entre $\frac{1}{4}$ et 4. Ainsi, les valeurs (en $\log_{10}$) minimales et maximales de la force sont $-\log(4)$ et $\log(4)$.

Dans le calcul de ϕ_{ij}^-, lorsque f_{ij} et F_j tendent vers $-\log(4)$, la somme de ces termes tend vers $-2\log(4)$. Ainsi la valeur théorique minimale de ϕ_{ij}^- est -1. De la même manière, on peut démontrer que la valeur théorique maximale de ϕ_{ij}^+ est 1.

Pour un i donné, il est possible à partir de toutes les valeurs ϕ_{ij}^- et ϕ_{ij}^+ de calculer ϕ_i^- et ϕ_i^+ :

$$\phi_i^- = \sum_{j \mid \forall\ (j,i) \in E} \phi_{ij}^- \tag{27}$$

$$\phi_i^+ = \sum_{j \mid \forall\ (i,j) \in E} \phi_{ij}^+ \tag{28}$$

Grâce à ces valeurs, nous pouvons reprendre le calcul du score[24] :

$$score_i = \frac{\phi_i^+ + \phi_i^-}{deg(i)} \tag{29}$$

Ce calcul donnera une valeur comprise entre -1 et 1. Contrairement aux deux autres algorithmes basés sur PageRank, ce score est proportionnel à la force d'une équipe. Il n'est pas nécessaire de le multiplier par -1 avant d'effectuer un classement.

2.5 Méthode de prédiction

La méthode de prédiction se base sur les résultats de l'algorithme de **Floyd-Warshall**. L'utilisation de cet algorithme permet de connaître la force minimale, directe ou non, d'une équipe i contre une équipe j. Notons β_{ij} cette force minimale.

Afin d'avoir des prédictions facilement interprétables, les résultats sont sous forme de pourcentage. Pour ce faire, on définie la probabilité que i soit plus fort que j à partir de β_{ij}. β_{ij} étant une force, on utilise la formule de la méthode Elo[1] :

$$\beta_{ij} = \frac{P_{ij}}{1 - P_{ij}}$$

$$\beta_{ij}(1 - P_{ij}) = P_{ij}$$

$$\beta_{ij} - \beta_{ij} \times P_{ij} = P_{ij}$$

$$\frac{\beta_{ij}}{P_{ij}} - \beta_{ij} = 1$$

$$\frac{\beta_{ij}}{Pij} = 1 + \beta_{ij}$$

$$\frac{P_{ij}}{\beta_{ij}} = \frac{1}{1 + \beta_{ij}}$$

$$\boxed{P_{ij} = \frac{\beta_{ij}}{1 + \beta_{ij}}} \tag{30}$$

Remarque : À la quatrième étape on divise l'équation par P_{ij}. Cette valeur est bornée entre $0{,}2$ et $0{,}8$, on ne divise donc jamais par 0.

Ainsi, nous avons la probabilité associée à la force minimale β_{ij}. En multipliant cette probabilité par 100, on obtient des pourcentages.

2.6 Aspects techniques

Lors du stage, j'ai utilisé une grande partie de mes connaissances en **Java**. J'ai aussi utilisé de nouvelles librairies comme Graphstream, decimal4j et Guava de Google. La librairie Graphstream m'a permis d'afficher les graphes et d'utiliser l'algorithme de **Floyd-Warshall** déjà implémenté avec la librairie. J'ai utilisé decimal4j pour arrondir des varibles de type double. Enfin Guava m'a permis d'utiliser des BiMap utiles pour passer d'un nombre à un nom et inversement (utilisée pour nommer les noeuds).

3 Conclusion

Pour conclure, nous avons vu comment représenter un championnat à partir d'un graphe orienté et pondéré par des probabilités. En se basant sur ce type de graphe, nous avons implémenté plusieurs méthodes de classement basées sur différentes approches (calcul des estimations de force, PageRank). Enfin nous avons utilisé l'algorithme de Floyd-Warshall pour établir des prédictions sur des rencontres à venir.

Tous les objectifs ont été remplis et nous envisageons d'intégrer le travail réalisé à un projet déjà existant. Nous pourrions ajouter une interface graphique à ce travail. De plus, de nouveaux axes de travail peuvent être étudiés comme par exemple l'organisation de championnat.