Elie Daher – 04/07/2017

- Modification du document "summary-uta" selon les remarques données la semaine passée
- Faire des recherches sur des littératures déjà effectué sur le nombre de critères et d'alternatives. Vous pouvez trouver un document "litterature" qui se trouve sur github sous le répertoire docs. Je n'ai pas trouvé grand choses sur google sur ce sujet (j'ai essaie encore à recherche sur www.scholar.google.fr). Mais je vais continuer ma recherche cette semaine afin de retrouver plus de livres et d'articles.
- Faire des recherches sur des articles sur la méthode UTA (UTASTAR) sur www.scholar.google.fr. J'ai identifié quelques cahiers, articles et livres qui pourront être utiles pour l'amélioration du document "summary-uta". Donc cette semaine je vais essaie d'améliorer un petit peu le document en se basant sur les littératures retrouvées.
- Supprimer l'exercice du UTASTAR : "Analyzing the choice of transportation" du document "summary-uta" et je l'ai mis dans le répertoire docs/exercices avec la classe java correspondante
- Création d'un programme AlternativeCriteria sous le répertoire src/alternative-criteria qui a comme paramètre `numCriteria` et `numAlternative` . Ce programme permet de générer des critères et des alternatives avec l'évaluation (au hasard entre la minValue et maxValue des critères concernés) des alternatives sur les critères. (Vous pouvez voire les captures d'écran dans le ReadMe file sur github)
- Création d'une classe Java GenerateNumbers sous le répertoire src/utils qui contient 2 méthodes qui permettent de créer une liste de nombres (integers ou double) par hasard qui ont une somme déjà prévue, par exemple pour créer 3 nombres qui ont une somme de 10. Pour réaliser cette méthode, j'ai créé la méthode `generateIntegers` qui prend comme paramètre `counters` (nombres = 3) et `targetSum` (somme ciblé = 10). Pour que le nombre générés soit pris au hasard j'avais deux possibilités :
    - Première possibilité
        - Pour la première itération, le nombre généré est égale à un nombre pris au hasard entre 1 et le targetSum
        - Pour les itérations qui suivent la première itération les nombres seront prisent au hasard entre 1 et targetSum – la somme des nombres générés.
        - Pour la dernière iteration, le nombre généré = targetSum – la somme des nombres générés.
    - Deuxième possibilité : Nombre générer est égale à un nombre pris au hasard entre 1 et (targetSum – les nombres déjà générer / counters - itération). Donc pour l'exemple de 3 nombre avec une somme de 10 on aura :
        - Pour la première itération, le nombre générer sera entre 1 et (10 – 0 / 4 – 1) = 10/3. Donc le premier nombre sera entre 1 et 3.33 ce qui va permettre d'avoir une dispersion plus équilibré. Par exemple si le premier nombre pris au hasard entre [1;10] est 10, les 2 autres nombres auront la valeur de

0 ce qui n'est pas très logique. Donc j'ai essaie de minimiser la dispersion des nombres générés à partir de cette formules.

J'ai exécuté les 2 possibilités 1000 fois dans un programme java sur un même exemple : « 10 nombres dont la somme est 100 » et à chaque fois j'ai calculé la dispersion des données. Pour la première possibilité j'ai eu une moyenne de 8 alors que pour la deuxième possibilité j'ai eu une moyenne de 6. Ce qui indique que la deuxième méthode permet une meilleure dispersion. (Je peux mettre en ligne le programme java que j'ai créé pour exécuter les méthodes 1000 fois et calculer la moyenne des dispersions, si ça vous intéresse). Donc laquelle vous préférer ? La première est plus aléatoire mais les nombres peuvent être plus dispersés que la deuxième méthode.

J'ai mis les captures d'écran sur des deux programmes « GenerateNumbers » et « Alternative – Criteria » dans le ReadMe File de github.

My Todo List :

- Formuler un exercice avec ça correction pour le mettre dans le document "summary-uta"
- Continuer la recherche sur des littératures à propos du nombre de critères et d'alternatives
- Améliorer le document "summary-uta" à partir des littératures retrouvées.