Création de fichier Tournoi

1. Méthodologie générale

La création de tournoi est réalisée en deux passes.

- Première passe : la méthode « try-and-error ». Il s'agit d'un algorithme récursif qui réalise des essais successifs en revenant en arrière en cas d'échec. Cette passe produit un tournoi de mauvaise qualité (avec des joueurs qui se rencontrent sur plus d'une partie)
  https://en.wikipedia.org/wiki/Trial_and_error
  Citation extraite de l'article wikipedia : « Nevertheless, this method is often used by people who have little knowledge in the problem area »
  Il est déterministe, c'est-à-dire que son comportement est toujours le même pour une entrée donnée.

- Deuxième passe : la méthode « hill-climbing ». Il s'agit d'un algorithme itératif qui réalise des échanges successifs pour améliorer la qualité du tournoi, en s'arrêtant quand ce n'est plus possible, et en repartant de la situation initiale (produite par la première passe) dans ce cas
  https://en.wikipedia.org/wiki/Hill_climbing
  Citation extraite de l'article wikipedia : « The relative simplicity of the algorithm makes it a popular first choice amongst optimizing algorithms »
  Il n'est pas déterministe, c'est-à-dire que son comportement est imprévisible pour une entrée donnée.

2. Détail de la passe "try-and-error"

La boucle récursive s'arrête dans la situation où il n'y a plus de partie incomplète.
Sinon, elle choisit la première partie complète, et le premier rôle non attribué dans cette partie.
Les joueurs acceptables pour être placés dans cette partie et dans ce rôle sont ceux remplissant les critères suivants :

- Pas déjà un rôle dans la partie
- Pas déjà dans une partie avec ce rôle

Compte tenu du fait qu'on ne peut arbitrer une partie dans laquelle on joue, le critère suivant est appliqué, qui permet d'être certain que cette partie trouvera un arbitre :

- Si un des joueurs déjà dans la partie est arbitre, ne pas y placer un deuxième.

Dans le cas où un seuil a été exigé sur le nombre d'interactions, le critère suivant est pris en compte :

- Le nombre d'interactions avec chacun des joueurs déjà dans la partie doit être strictement inférieur à ce seuil.

Compte tenu de ces critères, l'ensemble des joueurs acceptables a été restreint. S'il est vide, il y a déjà échec.
Dans le cas contraire, les joueurs sont testés dans l'ordre selon les critères suivants (importance décroissante) :

- Somme des interactions du joueur avec chacun avec chacun des joueurs déjà dans la partie
- Nombre de parties dans lesquelles le joueur est déjà
- Rang du joueur dans son inscription (pour assurer le déterminisme)

Le joueur est donc mis dans la partie, et l'algorithme est appliqué à nouveau dans la nouvelle situation.
Si aucun joueur ne peut être mis dans la partie, il y échec et retour à la situation antérieure, et essai du joueur suivant…

3. Détail de la passe "hill-climbing"

Dans une boucle infinie, on réalise les opération suivantes:

- Si les interactions ont disparu, on s'arrête (en succès)
- On détermine les candidats, les joueur qui en rencontrent un autre sur plus d'une partie…

- On détermine l'ensemble des joueurs complémentaire aux candidats
- On tente chaque couple, un candidat et un non candidat (dans le cas où l'ensemble des complémentaires est vide, on prendra deux candidats distincts)
- On tente chaque pays
- On vérifie que l'échange sur ce pays entre ces deux joueurs est possible, c'est à dire que cela ne mettrait pas deux fois le même joueur dans une même partie
- Si c'est le cas, alors on réalise l'échange
- La qualité de la situation résultante est mesurée par deux critères:
    - la plus mauvaise interaction
    - son nombre d'occurrences
- Si cela n'améliore pas la situation on défait cet échange, sinon on recommence au début de la boucle infinie
- Si la liste des couples est épuisée et qu'aucun n'a permis un changement, on s'arrête (en échec)

Le programme réalise une infinité de tentatives tant qu'il ne réussit pas ou que l'opérateur ne l'interrompt pas (par CTRL-C)

4. Paramètres

Paramètre	Obligatoire	Mise au point	Explication
-p <fichier>	Oui		Permet de préciser le fichier contenant les noms des joueurs
-m <fichier>	Oui		Permet de préciser le fichier contenant les noms des arbitres
-g <préfixe>	Oui		Permet de préciser le préfixe pour les noms des parties
-o <fichier>	Oui		Permet de préciser le fichier en sortie
-l <n>		Oui	Se limite aux <n> premier joueurs du fichier (mise au point)

5. Restrictions

Les restrictions suivantes sont applicables :

1. Sous Windows 10, le programme provoque une exception aux alentours de 320 joueurs (à cause de l'utilisation de la récursivité).

2. Avec les heuristiques utilisées, l'algorithme ne réussit presque jamais à obtenir un jeu d'allocations avec au plus une seule interaction pour moins de 90 joueurs

3. Le cas des arbitres tous mis dans la même partie, créant ainsi une partie impossible à arbitrer n'est pas géré et peut se produire (infiniment peu probable)

6. Performances

La première phase consomme un temps proportionnel au nombre de joueurs.
La seconde phase consomme un temps inversement proportionnel au nombre de joueurs.
Mesures réalisées sur un ordinateur "moyen"

Nombre de joueur	Temps
1000	26 secondes
100	55 secondes
50	> 35 secondes (jusqu'à l'arrêt par l'opérateur)

7. Interactions produites

Ci-dessous un tableau récapitulant différentes interactions produites par l'algorithme.

Nombre de joueur	Interactions
1000	Aucune
500	Aucune
300	Aucune
200	Aucune
100	Aucune
90	~ 2 (12)
80	~ 2 (45)
70	~ 2 (75)
60	~ 2 (82)
50	~ 2 (100)

8. Mise en œuvre

Étape une : télécharger in interpréteur (ou compilateur) du langage python. Le plus connu est ici : https://www.python.org/

Étape deux : télécharger le programme fourni en téléchargement **`allocation.py`**

Étape trois : construire un fichier texte avec la liste de joueurs et des arbitres

Étape quatre : appeler le programme en ligne de commande avec les arguments (ou avec -h pour le rappel des arguments)

9. Support

Vous avez un tournoi en vue et vous ne comprendrez rien ? Contactez le support du site