Chapitre 2

Projet

2.1 Projet 1 : Analyse des préférences

2.1.1 Partie 1 : Relation binaire quelconque

Soit A l'ensemble des objets et R une relation binaire sur A. Répondre aux questions suivantes en vous basant sur un programme informatique (de preference en Python). Afin d'être testés, **vous devez prévoir que vos programmes puissent être exécutés sur une relation binaire donnée dans un fichier txt sous un format correspondant au fichier** $exemple.txt$.

Votre projet a plusieurs étapes :

— Ecrire un programme qui analyse les propriétés de R : R est elle symétrique, asymétrique, transitive, negativement transitive, complete ? (vous pouvez faire une fonction par propriété.)

— Continuer votre programme pour déduire s'il s'agit d'une structure ordonnée vue en cours (ordre total, ordre partiel, preordre total, preordre partiel, semi-ordre ou ordre d'intervalle). Votre programme doit donc donner un message sur la structure trouvée.

— Ecrire une fonction qui prend en argument une relation quelconque et qui renvoie l'ordre total le plus proche de R que l'on notera S.
Pour cela vous pouvez minimiser la distance de Kemeney (distance entre deux relations binaires R et S) :

$$D(R,S) = \sum_{a,b \in A} (d(a,b))$$

$d(a,b) = 1$ si ($R(a,b)$ et $\neg S(a,b)$) ou ($\neg R(a,b)$ et $S(a,b)$) , $d(a,b) = 0$ si $R(a,b)$ et $S(a,b)$.
N'oubliez pas que S doit être un ordre total !

Votre programme doit fonctionner de la manière suivante :

i. vous récupérez les données sur une relation binaire à l'aide d'un fichier .txt sous un format correspondant au fichier $exempleRelation.txt$.

ii. votre programme doit ensuite donner la liste des propriétés vérifiées et la liste des propriétés violées. Pour chaque propriété non vérifiée, votre programme doit donner au moins un exemple de violation.

iii. votre programme doit ensuite nous dire s'il s'agit d'une structure de préférence vue en cours. Si ce n'est pas un ordre total, il doit nous donner l'ordre total le plus proche de votre relation binaire de départ.

Document à rendre pour Partie 1 : (un rapport + le code)

Code : il faut que je puisse tester votre code avec de nouvelles relations qui seront rentrées sous format .txt. On attend bien sûr un code bien organisé et commenté.

Rapport : dans votre document, vous devez redonner le code de votre programme avec des explications ainsi que les résultats affichés pour les relations binaires ci-dessous :

1. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e
a	1	1	0	1	1
b	0	1	0	0	1
c	1	1	1	1	1
d	1	1	1	1	1
e	1	1	0	0	1

2. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e	f
a	1	1	1	1	0	0
b	0	1	1	1	0	0
c	0	1	1	1	0	0
d	0	1	1	1	0	0
e	1	1	1	1	1	1
f	1	1	1	1	1	1

3. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e	f
a	1	1	0	0	0	0
b	0	1	1	0	0	0
c	0	0	1	0	0	0
d	0	0	1	1	0	0
e	0	1	0	0	1	0
f	1	1	0	0	0	1

4. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e
a	1	0	1	1	0
b	1	1	1	1	0
c	1	0	1	0	0
d	1	0	1	1	0
e	1	1	1	1	1

5. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e	f
a	1	0	1	1	0	1
b	1	1	1	1	1	1
c	0	0	1	0	0	0
d	0	0	1	1	0	1
e	1	0	1	1	1	1
f	0	0	1	0	0	1

6. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e
a	1	1	0	0	0
b	0	1	1	0	0
c	1	0	1	1	0
d	0	0	0	1	0
e	0	0	0	1	1

7. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e	f
a	1	0	0	1	1	1
b	0	1	0	0	0	0
c	0	1	1	1	1	0
d	0	0	0	1	1	0
e	0	0	0	0	1	0
f	0	0	0	1	1	1

Bonus :

— Faire des tests pour savoir jusqu'à combien d'éléments votre programme tourne (surtout pour trouver l'ordre total le plus proche).

— Pour trouver l'ordre total le plus proche de votre relation binaire vous pouvez faire un deuxième programme avec une resolution via une modelisation sous forme de PL. Vous pouvez faire des tests pour savoir jusqu'à combien d'éléments votre programme tourne et comparer vos résultats avec votre premier code en python (recherche exhaustive).

2.1.2 Partie 2 : Semi-ordre

Soit A l'ensemble des objets et R un semi-ordre sur A. Répondre aux questions suivantes en vous basant sur un programme informatique.

— Ecrire un programme qui calcule les degrés des sommets du graphe de R.
Le degré du sommet qui représente l'objet a : $|x \neq a \in A, aRx| - |x \neq a \in A, xRa|$
exemple : Soit $A = \{a, b, c\}$ l'ensemble des objets avec $aRb, bRa, bRc, cRb, cRa, aRa, bRb, cRc$ (c.a.d aIb, bIc et cPa). Alors le degré de a est (1-2=-1) et le degré de b est (2-2=0), etc.

— Ecrire la suite du programme pour trouver une representation numérique de R, c.a.d associer des intervalles aux éléments de A qui vérifient la caractérisation numérique d'un semi-ordre (aPb Ssi l'intervalle de a est devant l'intervalle de b sans intersection, aIb Ssi les deux intervalles ont une intersection non vide, et tout ca avec des intervalles de meme longueur.)
suite de l'exemple : on peut par exemple associer les intervalles suivantes : $a = [1, 4], b = [3, 6]$ et $c = [5, 8]$

Document à rendre pour Partie 2 : (suite du rapport + suite du code)

Code : il faut que je puisse tester votre code avec de nouvelles relations qui seront rentrées sous format .txt. On attend bien sûr un code bien organisé et commenté

Rapport : dans votre document, vous devez redonner le code de votre programme avec des commentaires ainsi que les resultats affichés pour les relations binaires ci-dessous :

1. $A = \{a, b, c, d, e, f\}$

$M(i,j)$	a	b	c	d	e	f
a	1	1	0	0	1	1
b	0	1	0	0	1	1
c	1	1	1	1	1	1
d	1	1	1	1	1	1
e	0	1	0	0	1	1
f	0	0	0	0	0	1

2. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e
a	1	1	1	1	1
b	0	1	1	1	1
c	0	0	1	0	1
d	1	1	1	1	1
e	0	1	1	0	1

3. $A = \{a, b, c, d, e\}$

$M(i,j)$	a	b	c	d	e
a	1	0	0	0	0
b	1	1	0	1	1
c	1	1	1	1	1
d	1	1	0	1	1
e	1	1	0	0	1

2.2 Projet 2 : Analyse des methodes de choix social

Nous allons analyser les deux approches fondateurs du choix social avec des simulations. Voici les notations :

Soit $A = \{a, b, c, ...\}$ l'ensemble des candidats à une élection et $V = \{v_1, \ldots, v_n\}$ l'ensemble des votants. Chaque votant range les candidats sous forme d'un ordre total et on note R_i l'ordre total du votant v_i.

Il existe plusieurs méthodes pour agréger les préférences des votants qui peuvent être assez divergentes.

2.2.1 Méthode de Condorcet et de Copeland

Condorcet :

On compare deux à deux tous les candidats pour construire la relation de majorité (faible)(notée $\succeq_{maj}$) :

$$\forall x, y \in A, x \succeq_{maj} y \iff |i(v_i \in V), xR_iy| \geq |i(v_i \in V), yR_ix|$$

x est majoritairement préféré à y si et seulement s'il y a une majorité de votants pour qui x est au moisn aussi bon que y. On dit que le candidat x^* est le **vainqueur de Condorcet** Ssi $\forall x \in A, x^* \succeq_{maj} x$.

Remarque : La relation majoritaire peut engendrer des cycles car elle n'est pas nécessairement transitive et cela peut causer l'absence du vainqueur de Condorcet (exemple : cas de trois votants sur trois candidats : aR_1bR_1c, bR_2cR_2a et cR_3aR_3b).

Copeland :

La méthode de Copeland se base sur la relation de majorité pour définir un score à chaque candidat, ce qui garantit la transivité de la méthode :

$$score_{Copeland}(x) = |y \in A, x \succeq_{maj} y| - |y \in A, y \succeq_{maj} x|$$

On range les candidats dans l'ordre de leur score de Copeland : $\forall x, y \in A$, le classement de x est au moins aussi bon que celui de y Ssi $score_{Copeland}(x) \geq score_{Copeland}(y)$.

2.2.2 Méthode de Borda

Chaque candidat obtient un score qui est la somme de ses rangs dans les préférences des votants :

$$\forall x \in A, score_{Borda}(x) = \sum_i rang_i(x)$$

où $rang_i(x)$ represente le rang du candidat x dans l'ordre total du votant v_i (par exemple, si le votant 2 vote sur $\{a, b, c, d\}$ comme suit : $bR_2cR_2aR_2d$ alors $rang_2(a) = 3$ et $rang_2(b) = 1$, etc.).

On range ensuite les candidats grâce à leur score de Borda : le classement de x est au moins aussi bon que celui de y Ssi $score_{Borda}(x) \leq score_{Borda}(y)$.

Exemple :

Supposons que nous avons 4 votants qui donnent leur ordre sur trois candidats a, b, c comme ci-dessous :

$v_1 : aR_1cR_1b$
$v_2 : aR_2cR_2b$
$v_3 : aR_3cR_3b$
$v_4 : cR_4bR_4a$
$v_5 : cR_5bR_5a$

Remarque : pour faciliter la representation, on aura pu mettre ensemble les mêmes votes et préciser le nombre de votants, comme ci-dessous :

$3 : aRcRb$
$2 : cRbRa$

La relation majoritaire donne : $a \succeq_{maj} c \succeq_{maj} b$, donc a est le vainqueur de Condorcet.
Les scores de Copeland donnent : $score_{Copeland}(a) = 2, score_{Copeland}(b) = -2, score_{Copeland}(c) = 0$, donc a est le vainqueur de Copeland.
Les scores de Borda donnent : $score_{Borda}(a) = 9, score_{Borda}(b) = 13, score_{Borda}(c) = 8$, donc c est le vainqueur de Borda.

2.2.3 Simulation avec “impartial preferences”

Afin d’analyser expérimentalement ces deux approches, on va simuler des préférences sur des élections. Une des manières est de faire l’hypothèse que les probabilités sur les rangs des candidats soient uniformes (hypothèse très critiquable pour des élections réelles). Pour simuler une élection, vous pouvez simuler les préférences de chaque votant de manière indépendante des unes des autres. Et pour un votant donné, les rangs des candidats peuvent être simulés de manière uniforme.

2.2.4 Analyse des méthdoes

Les questions que vous allez analyser sont :

Question 1 : Quel est la probabilité que lors d’une élection, on ne trouve pas de vainqueur de Condorcet (pour cela on regardera le pourcentage de cas (élections) dans vos simulations, on n’analysera pas la réponse théorique) ? Est-ce que ce pourcentage depend du nombre de candidats et/ou de votants ?

Question 2 : Parmi les cas où il existe un vainqueur de Condorcet, quel est le pourcentage de cas où le vainqueur de Borda est aussi le vainqueur de Condorcet ? Est-ce que ce pourcentage depend du nombre de candidats et/ou de votants ? Que constatez vous ?

Question 3 : Parmi les cas où il existe un vainqueur de Condorcet, quel est le pourcentage de cas où le vainqueur de Copeland est aussi le vainqueur de Condorcet ? Est-ce que ce pourcentage depend du nombre de candidats et/ou de votants ? Que constatez vous ?

Question 4 : Quel est le pourcentage de cas où le vainqueur de Copeland est aussi le vainqueur de Borda ? Est-ce que ce pourcentage depend du nombre de candidats et/ou de votants ? Que constatez vous ?

Pour répondre à ces questions, suivez les points suivants :

Analyse par rapport au nombre de votants

On va fixer le nombre de candidats à 5 (a, b, c, d, e) et nous allons varier le nombre de votants (de 3 à 19, ne prenez que des cas impairs pour éviter des égalités dans la relation de majorité). Pour chaque nombre de votants, vous allez simuler 50 elections pour répondre aux questions 1, 2 et 3. Pour chaque question, dessiner le graphique qui montre l'evolution du pourcentage en fonction du nombre de votants.

On veut voir aussi les temps d'execution de vos programmes en fonction du nombre de votants. Marquez aussi pour chaque candidat, le pourcentage des fois où il est élu (avec les trois méthodes).

Analyse par rapport au nombre de candidats

On va fixer le nombre de votants à 19 et nous allons varier le nombre de candidats (de 2 à 7). Pour chaque nombre de votants, vous allez simuler 50 elections pour répondre aux questions 1, 2, 3. Pour chaque question, dessiner le graphique qui montre l'evolution du pourcentage en fonction du nombre de candidats. On veut voir aussi les temps d'execution de vos programmes en fonction du nombre de candidats.

Document à rendre (un rapport + le code)

Code : il faut que je puisse tester votre code avec différentes valeurs de votants et de candidats. On attend bien sûr un code bien organisé et commenté.

Rapport : dans votre document, vous devez redonner le code de votre programme avec des explications ainsi que les résultats obtenus sur les deux parties 2.2.4 et 2.2.4. Pour chaque partie vous devez fournir un graphe par question et argumenter votre graphe.

2.2.5 Bonus : manipulabilite

On va imaginer un cas où tous les votants sauf le dernier ont déjà présenté leur ordre total sur les candidats. Nous allons supposer que l'ordre total du dernier votant est de forme $a > b > c > d > e > ...$ et que ce dernier connaît les votes des autres mais ne montre pas encore ses préférences aux autres. La question que nous allons nous poser est la suivante : Serait il possible pour le dernier votant de mentir sur ses préférences pour modifier le résultat des votes en sa faveur ? Nous allons analyser cette question pour la méthode de Borda.

Voici un exemple de manipulation qui correspond à notre problème : Soit $A = \{a, b, c, d\}$ et $V = \{v_1, \ldots, v_6\}$. Les ordres totaux des 5 premiers votants sont :

$2 : c > d > b > a$
$2 : d > c > a > b$
$1 : a > d > b > c$

Avec les votes des 5 premiers, les scores de Borda sont :

$$score(a) = 15, score(b) = 17, score(c) = 10, score(d) = 8$$

Pour le moment c'est d qui gagne. Si v_6 vote sincèrement ($a > b > c > d$) les scores seront :

$$score(a) = 16, score(b) = 19, score(c) = 13, score(d) = 12$$

Ca sera toujours d, le pire candidat pour v_6, qui sera élu. En voyant ca, v_6 peut mentir et voter comme suit : $(c > a > b > d)$, ce qui donnerait comme scores :

$$score(a) = 17, score(b) = 20, score(c) = 11, score(d) = 12$$

Ca sera alors le candidat c qui sera élu, donc en mentant sur ses préférences, v_6 a amélioré le résultat en sa faveur car il préfère c à d.

Ecrire un algorithme qui prend en entréé les votes de tous les votants sauf dernier et qui dit si le dernier votant (dont les votes sont de la forme $a > b > c > d > e > ...$) peut manipuler l'élection en sa faveur. Si la réponse est oui, donner les préférences que le dernier votant doit présenter pour cette manipulation.

Remarque : Cette notion de manipulation est une notion très souvent étudiée en choix social, surtout depuis que le théorème de *Gibbard-Satterthwaite* [1], [2] a montré que toute règle de vote (qui vérifie quelques propriétés de base) est manipulable.

Par exemple pour ParcoursSup (qui est un problème d'affectation et non de vote), il sera bien sûr préférable d'avoir une méthode qui ne sera pas manipulable, ni par des étudiants, ni par des universités...

1. Allan Gibbard, Manipulation of Voting Schemes : A general result, Econometrica, vol. 41, no4, 1973, P. 587-601

2. Mark Satterthwaite, Strategy-proofness and Arrow's conditions : Existence and correspondence theorems for voting procedures and social welfare functions, Journal of Economic Theory, vol. 10, no 2, 1975, P. 187-217