Stage de recherche

Problème d'appariement : application à l'affectation de stages aux Professeurs des écoles

Auteur :
Éric Piette

Encadrant :
Maxime Morge

Équipe SMAC

20 août 2013

Table des matières

1 Introduction

Ce rapport présente les travaux de recherche qui ont été effectués dans le cadre de mon stage de Master 2. Ce stage a été effectué entre le 04 mars 2013 et le 19 juillet 2013 dans l'équipe SMAC (Systèmes Multi-Agents et Comportements) du LIFL (Laboratoire d'Informatique Fondamentale de Lille) sous la direction de Maxime Morge.

Le sujet de ce stage s'inscrit dans la continuité du projet de recherche « Sur la stabilité et l'équité des solutions aux problèmes des mariages ». Ce projet se consacrait au problème des mariages stables - en anglais, *Stable Marriage Problem (SMP)*, bien connu en Informatique [5] et en Économie [7]. Dans ce problème nous considérons deux communautés dont chaque membre a des préférences vis-à-vis des individus de l'autre communauté. L'objectif est de former des couples en prenant en compte les préférences que possède chaque individu vis-à-vis de leurs partenaires potentiels.

Dans notre travail, nous étudions des extensions du problème du mariage stable où dans un premier temps, un individu peut préférer être seul que mal accompagné - en anglais, *Stable Marriage with Incomplete Lists (SMI)* - et dans un second temps, un problème dit d'hôpital/résident (HR). Pour chacune de ces extensions, nous nous intéressons à des variantes de l'algorithme de Gale-Shapley [3] qui consiste à distinguer deux communautés : une de proposants et une de disposants. L'échange de proposition entre ces communautés permet d'atteindre une solution stable.

Au cours de notre travail, nous nous intéressons à une application pratique : l'affectation de stages pour les professeurs des écoles en formation dans l'académie de Lille. En effet, l'`IUFM` (Institut Universitaire de Formation des Maîtres) souhaite automatiser cette procédure qu'elle veut équitable et rentable autant que faire se peut. Dans ce document, nous proposons et évaluons nos solutions afin de résoudre ce problème pratique.

Nous commençons par une formalisation du problème des mariages stables et de ces extensions dans la section 2. Ensuite nous décrivons l'application pratique puis nous proposons et évaluons nos algorithmes de résolution en section 3. Notre implémentation est décrite en section 4. Enfin, nous terminons par une discussion.

2 Problème du mariage stable

Nous étudions ici le problème des mariages stables, en anglais *Stable Marriage Problem*, qui a été introduit par Gale et Shapley [3] en 1962. Ce problème s'énonce simplement. On considère n individus d'une communauté (e.g. la gent féminine) et n individus d'une autre communauté (e.g. la gent masculine). Chaque individu a des penchants pour les individus de l'autre partition avec lesquels il souhaite être apparié. Dans une première partie 2.1, nous définissons un problème de mariable stable complet (SMC) où chaque individu ordonne l'ensemble de ses partenaires potentiels. Dans une deuxième partie 2.2, nous étudions un problème de mariable stable incomplet (SMI) où les préférences de chaque individu peuvent être incomplètes. Finalement, dans une dernière partie 2.3, nous nous intéressons au problème d'hôpitaux/résidents (HR), une extension du problème du mariage stable qui autorise la polygamie.

2.1 SMC

Un problème de mariage stable est complet si chaque individu classe tous les individus de l'autre communauté.

Définition 1 (SMC) *Un **problème de mariage stable complet** de taille n (avec $n \geq 1$) est un couple $SMC = \langle X, Y \rangle$ avec $|X| = |Y| = n$, où :*

- *$X = \{x_1, \ldots, x_n\}$ est un ensemble de n hommes, i.e. relations d'ordre **strict** et **complètes** sur Y, représentant les préférences des hommes vis-à-vis des femmes ;*
- *$Y = \{y_1, \ldots, y_n\}$ est un ensemble de n femmes, i.e. relations d'ordre **strict** et **complètes** sur X, représentant les préférences des femmes vis-à-vis des hommes.*

On note $y_2 \succ_{x_1} y_3$ le fait que l'homme x_1 préfère strictement la femme y_2 à la femme y_3 et $|z|$ la taille de la liste de préférence d'un individu z.
Dans un problème SMC, une relation de préférence est :

- **un ordre strict**, i.e. si $x_1 \succ_{y_1} x_2$ alors on n'a pas $x_2 \succ_{y_1} x_1$;
- **un ordre complet**, i.e. chaque liste de préférences est composée de toute la communauté opposée ($\forall z \in X \cup Y, |z| = n$).

Résoudre un problème SMC de taille n consiste à assortir ces communautés. On appelle appariement, en anglais *matching*, un ensemble de n mariages monogames et hétérosexuels.

Définition 2 (Appariement) *Un **appariement** M pour le problème $SMC = \langle X, Y \rangle$ de taille n est une application $p_M : X \cup Y \to X \cup Y$ telle que :*

1. *$\forall x \in X, p_M(x) \in Y$;*
2. *$\forall y \in Y, p_M(y) \in X$;*
3. *$\forall z \in X \cup Y, p_M(p_M(z)) = z$.*

$p_M(z)$ représente le partenaire de z selon l'appariement M. Dans un appariement, chaque individu est marié et l'application p est bijective.
$p_M(z) \succ_z p_{M'}(z)$ signifie que z préfère (son partenaire dans) l'appariement M à (son partenaire dans) M'. On note $p_M(z) \succeq_z p_{M'}(z)$ si $p_M(z) \succ_z p_{M'}(z)$ ou $p_M(z) = p_{M'}(z)$.

Dans un problème SMC, un appariement est stable s'il n'existe pas de couple qui préférerait être ensemble plutôt qu'avec leur partenaire actuel. En d'autres termes, il n'existe pas de relation extra-conjugale menaçant l'appariement. Ces relations prennent la forme de couples hypothétiques bloquants.

Définition 3 (Stabilité) *Soient un problème SMC* $= \langle X, Y \rangle$ *de taille* n *et* M *un appariement pour SMC. Un couple* $(x_i, y_i) \in X \times Y$ *est* ***bloquant*** *si* $y_i \succ_{x_i} p_M(x_i)$ *et* $x_i \succ_{y_i} p_M(y_i)$. M *est* ***instable*** *s'il existe au moins un couple bloquant.*

Une solution admissible à un problème SMC de taille n est un appariement stable comportant n couples. En 1962, David Gale et Lyold Shapley ont proposé un algorithme [3] qui constitue une preuve constructive de l'existence systématique d'une solution pour toute instance de SMC.

L'algorithme Gale-Shapley (GS) peut être décrit comme un cérémonial au cours duquel les femmes attendent dans une salle et les hommes, qui sont à l'extérieur, rentrent à tour de rôle dans un ordre arbitraire. Quand un homme entre, il se présente à la femme qu'il préfère. Si la femme est libre alors elle devient son partenaire. Si elle a déjà un partenaire, il y a conflit et elle choisit l'homme qu'elle préfère. L'élu reste et l'autre, rejeté, sort de la pièce. Ce dernier se souvient de cet échec et reviendra plus tard pour se présenter à la femme qui suit dans l'ordre de ses préférences. On s'arrête quand tous les hommes sont mariés.

Dans GS (cf. algorithme 1), on représente la liste de préférence d'un individu z par $z.\pi$. θ dénote l'individu fantomatique. $p_M(z) = \theta$ signifie que z est libre dans l'appariement M. Initialement, l'appariement ne contient aucun couple (lignes 1 et 2). Dans l'algorithme ainsi présenté, les hommes proposent et les femmes disposent. Une exécution consiste en une séquence de propositions des hommes envers les femmes. En d'autres termes, les hommes jouent le rôle de proposant (ligne 5) et les femmes jouent le rôle de disposant (ligne 8). Après s'être mariée, une femme ne divorcera que pour les hommes qu'elle préfère à son partenaire actuel. Elle peut donc supprimer de sa liste de préférences les autres hommes (ligne 15) et les informer qu'il leur est inutile de s'adresser à elle (ligne 16). L'algorithme que nous avons présenté est **orienté homme**. Les hommes proposent et les femmes disposent. Lorsque les femmes sont proposantes et les hommes sont disposants, l'algorithme est dît **orienté femme.**

Exemple 1 *Soient les relations de préférence suivantes* $SMC = \langle \{x_1, x_2, x_3\}, \{y_1, y_2, y_3\} \rangle$ *avec :*

$y_2 \succ_{x_1} y_1 \succ_{x_1} y_3$ $\quad$ $x_2 \succ_{y_1} x_1 \succ_{y_1} x_3$
$y_3 \succ_{x_2} y_2 \succ_{x_2} y_1$ $\quad$ $x_3 \succ_{y_2} x_2 \succ_{y_2} x_1$
$y_1 \succ_{x_3} y_3 \succ_{x_3} y_2$ $\quad$ $x_1 \succ_{y_3} x_3 \succ_{y_3} x_2$

Pour cette instance, l'appariement M_0 *est instable car* (x_2, y_2) *est un couple bloquant :*

- $p_{M_0}(x_1) = y_2$, $p_{M_0}(x_2) = y_1$, $p_{M_0}(x_3) = y_3$

À l'inverse, les trois appariements M_1, M_2 *et* M_3 *sont stables :*

- $p_{M_1}(x_1) = y_1$, $p_{M_1}(x_2) = y_2$, $p_{M_1}(x_3) = y_3$
- $p_{M_2}(x_3) = y_1$, $p_{M_2}(x_1) = y_2$, $p_{M_2}(x_2) = y_3$
- $p_{M_3}(x_2) = y_1$, $p_{M_3}(x_3) = y_2$, $p_{M_3}(x_1) = y_3$

Algorithme 1 : GS orienté homme

Données : un problème SMC $= \langle X, Y \rangle$
Résultat : un appariement M

```
pour tous les z ∈ X ∪ Y faire
    p_M(z) ← θ;
pour tous les x ∈ X, p_M(x) = θ faire
    y ← x.π[0];
    //x se propose à y
    x_2 ← p_M(y);
    si x ≻_y x_2 alors
        //y dispose
        //x et y se marient
        p_M(x) ← y;
        p_M(y) ← x;
        si x_2 ≠ θ alors
            p_M(x_2) ← θ; //y divorce de x_2
        pour tous les x_3 ∈ y.π, x ≻_y x_3 faire
            y.π.supprimer(x_3); //y ne concède plus
            x_3.π.supprimer(y); //y informe les concernés
retourner M;
```

On peut noter que M_1 ne peut pas être atteint par GS. M_2 est le résultat de l'algorithme GS orienté homme. M_3 est le résultat de l'algorithme GS orienté femme.

Étudions le problème de mariage incomplet où au moins un individu ne considère pas l'ensemble de la communauté opposée.

2.2 SMI

Un problème de mariage est incomplet si au moins l'un des individus ne classe pas tous les membres de la communauté opposée.

Définition 4 (SMI) *Un **problème de mariage stable incomplet** de taille n (avec $n \geq 1$) est un couple SMI $= \langle X, Y \rangle$ avec $|X| = |Y| = n$, où :*

- *$X = \{x_1, \ldots, x_n\}$ est un ensemble de n hommes, i.e. relations d'ordre **strict** sur Y, représentant les préférences des hommes sur l'ensemble des femmes ;*
- *$Y = \{y_1, \ldots, y_n\}$ est un ensemble de n femmes, i.e. relations d'ordre **strict** sur X, représentant les préférences des femmes sur l'ensemble des hommes.*

Comme un problème SMC, les relations d'ordre d'un problème SMI sont stricts. Toutefois, elles peuvent être incomplètes ($\exists z \in X \cup Y, |z| < n$).
La résolution d'un problème SMI de taille n consiste à réaliser des appariements composés de 0 à n mariages monogames et hétérosexuels.

Définition 5 (Appariement) *Un* ***appariement*** M *pour le problème SMI* $= \langle X, Y \rangle$ *de taille* n *est une application* $p_M : X \cup Y \to X \cup Y \cup \{\theta\}$ *telle que :*

1. $\forall x \in X, p_M(x) \in Y \cup \{\theta\}$;
2. $\forall y \in Y, p_M(y) \in X \cup \{\theta\}$;
3. $\forall z \in X \cup Y,\ p_M(z) \neq \theta \to p_M(p_M(z)) = z$.

Dans le cas de relation d'ordre incomplet, un individu préfére être seul plutôt que mal accompagné. Autrement dit, un individu est irrationnel s'il se marie avec un partenaire qui n'est pas dans sa liste de préférence. Un appariement est rationnel si tous les individus sont rationnels.

Définition 6 (Rationalité) *Soient un problème SMI* $= \langle X, Y \rangle$ *de taille* n *et* M *un appariement pour SMI.* M *est* ***rationnel*** *si :*

$$\forall (x_i, y_i) \in M,\ x_i \in y_i.\pi \ \wedge\ y_i \in x_i.\pi$$

Un appariement de SMI est stable s'il n'existe pas de couple bloquant (cf définition 3).

Une solution admissible à un problème SMI de taille n est un appariement stable et rationnel composé de m couples où $m \in [0; n]$. David Gale a démontré dans [4] que toute solution d'une même instance de SMI est de même taille.

L'algorithme GS peut être étendu afin de résoudre des problèmes SMI. Dans cette variante (cf algorithme 2), une femme libre n'accepte un homme que si celui-ci est dans sa liste de préférences (cf ligne 7). On s'arrête quand tous les hommes qui ne sont pas libres sont désespérés, c.-à-d. leur liste de préférences est vide (cf ligne 3).

Exemple 2 *Soient les relations de préférences suivantes* $SMI = \langle \{x_1, x_2, x_3, x_4\}, \{y_1, y_2, y_3, y_4\} \rangle$ *avec :*

$y_2 \succ_{x_1} y_1 \succ_{x_1} y_3$ $\quad$ $x_2 \succ_{y_1} x_4 \succ_{y_1} x_1$
$y_1 \succ_{x_2} y_2 \succ_{x_2} y_4 \succ_{x_2} y_3$ $\quad$ $x_3 \succ_{y_2} x_4 \succ_{y_2} x_1$
$y_4 \succ_{x_3} y_1 \succ_{x_3} y_2$ $\quad$ $x_3 \succ_{y_3} x_4 \succ_{y_3} x_2$
$y_1 \succ_{x_4} y_2 \succ_{x_4} y_4$ $\quad$ $x_4 \succ_{y_4} x_2 \succ_{y_4} x_3$

Pour cette instance, l'appariement M_0 *est instable et irrationnel car* (x_2, y_1) *est un couple bloquant et il est irrationel pour* x_4 *de se marier avec* y_3 *qui n'est pas dans sa liste de préférences :*

– $p_{M_0}(x_1) = y_1$, $p_{M_0}(x_2) = y_4$, $p_{M_0}(x_3) = y_2$, $p_{M_0}(x_4) = y_3$.

À l'inverse, les deux appariements M_1, M_2 *sont stables et rationnels :*

– $p_{M_1}(x_2) = y_1$, $p_{M_1}(x_3) = y_4$, $p_{M_1}(x_4) = y_2$;

– $p_{M_2}(x_2) = y_1$, $p_{M_2}(x_3) = y_2$, $p_{M_1}(x_4) = y_4$.

On peut noter que M_1 *est le résultat de GS étendu orienté homme et* M_2 *le résultat de GS étendu orienté femme.*

Étudions le problème d'hôpitaux/résident, une extension du problème de mariage stable incomplet où la polygamie est autorisée.

Algorithme 2 : EGS orienté homme

Données : un problème SMI $= \langle X, Y \rangle$
Résultat : un appariement M

```
1  pour tous les z ∈ X ∪ Y faire
2      p_M(z) ← θ;
3  pour tous les x ∈ X, p_M(x) = θ ∧ |x| ≠ 0 faire
4      y ← x.π[0];
5      //x se propose à y
6      x_2 ← p_M(y);
7      si x ∈ y.π alors
8          si x ≻_y x_2 alors
9              //y dispose
10             //x et y se marient
11             p_M(x) ← y;
12             p_M(y) ← x;
13             si x_2 ≠ θ alors
14                 p_M(x_2) ← θ; //y divorce de x_2
15             pour tous les x_3 ∈ y.π, x ≻_y x_3 faire
16                 y.π.supprimer(x_3); //y ne concède plus
17                 x_3.π.supprimer(y); //y informe les concernés
18 retourner M;
```

2.3 HR

Le problème d'hopitaux/résidents est un problème qui consiste à affecter à différents hopitaux un certain nombre de résidents. Ce problème de mariage stable « un-à-plusieurs » a été introduit par Gale et Shapley comme un problème d'admissions dans des écoles [3].

Définition 7 (HR) *Un* ***problème d'hopitaux/résidents*** *de taille* (n, m)*, avec* $n \geq 1$ *et* $m \geq 1$*, est un couple* $HR = \langle R, H \rangle$ *avec* $|R| = n$ *et* $|H| = m$*, où :*

- $R = \{r_1, \ldots, r_n\}$ *est un ensemble de* n *résidents, i.e. relations d'ordre* ***strict*** *sur* H*, représentant les préférences des résidents sur l'ensemble des hopitaux ;*
- $H = \{h_1, \ldots, h_n\}$ *est un ensemble de* m *hopitaux. Chaque hopital* h *peut accueillir au plus* $h.q$ *résidents (avec* $1 \leq h.q \leq n$*). De plus, chaque* h *représente une relation d'ordre* ***strict*** *sur* R*, correspondant à ses préférences sur l'ensemble des résidents qui le considèrent.*

Il est important de noter que les listes de préférences sont éventuellement incomplètes et que, contrairement à SMI, la liste de préférences d'un hôpital ne contient que les résidents pour lesquels, cet hopital est rationnel.

Le problème HR peut être considéré comme une extension de SMI car les relations de préférences sont strictes et éventuellement incomplètes. Cependant, les communautés ne sont plus de même taille et les hopitaux peuvent être appariés à plusieurs résidents. Dans le cas particulier où l'ensemble des hopitaux ont

un quota de 1, un problème HR se réduit dans un problème SMI en assimilant les résidents à des hommes et les hopitaux à des femmes.
Une solution à un problème HR est un appariement de 0 à n résidents en poste à chacun des m hopitaux.

Définition 8 (Appariement) *Un* ***appariement*** M *pour le problème* $HR = \langle H, R\rangle$ *de taille* (n, m) *est représentée par* $a_M : R \to H \cup \{\theta\}$ *et* $p_M : H \to \mathcal{P}(R)$ *tel que :*

1. $\forall r \in R,\ a_M(r) \in H \cup \{\theta\}$;
2. $\forall h \in H,\ p_M(h) \subseteq R$;
3. $\forall r \in R,\ \{r\} \subseteq p_M(a_M(r))$;
4. $\forall h \in H,\ \forall r \in p_M(h),\ a_M(r) = h$.

L'affectation (a_M) d'un résident est un hôpital, éventuellement réduit à l'individu fantome (cf équation 1). Les postes (p_M) des hopitaux sont des ensembles de résidents potentiellement vides (cf équation 2). Comme dans un problème SM, l'affectation est réciproque (cf équations 3 et 4).

Notons que dans un appariement M, si $a_M(r) = \theta$, on dit que r n'est pas affecté. On dit qu'un hôpital h :
- est surchargé si $|p_M(h)| >$ h.q ;
- est plein si $|p_M(h)| =$ h.q ;
- est sous chargé si $|p_M(h)| <$ h.q.

Une affectation de HR est dîte rationnelle si tous les résidents considèrent leur affectation comme acceptable.

Définition 9 (Rationalité) *Soient un problème* $HR = \langle R, H\rangle$ *de taille* (n, m) *et* M *une affectation pour HR.* M *est* ***rationnelle*** *si :*

$$\forall r \in R, a_M(r) \in r.\pi$$

On peut remarquer que la rationalité individuelle des résidents est une condition suffisante pour garantir la rationalité d'un appariement car, par définition, la liste de préférences des hôpitaux est resteinte aux résidents qui y postulent.

Pour un appariement de type HR, un couple $(h,\ r)$ est bloquant si r préfère h à son affectation et : soit h est sous chargé ; soit h préfère r à l'un au moins de ses résidents.

Définition 10 (Stabilité) *Soient un problème* $HR = \langle H, R\rangle$ *de taille* (n, m) *et* M *un appariement pour HR. Un couple* $(h, r) \in H \times R$ *est* ***bloquant*** *ssi :*

1. $h \succ_r a_M(r)$ *et ;*
2. $(|p_M(h)| < h.q) \vee (\exists r^{'} \in p_M(h), r \succ_h r^{'})$.

M *est* ***instable*** *s'il existe un couple bloquant.*

Une **solution admissible** au problème HR est un appariement stable, rationnel et ne possédant aucun hopital surchargé.

Inspiré par Gale et Shapley [3], Gusfield et Irving [2] décrivent une variante de GS orientée résident (RGS) et une variante orientée hôpital (HGS) qui peuvent être appliquées à HR.

Dans RGS (cf. algorithme 3), le moins bon résident en poste dans un hopital h est noté $min_p(h)$. Initialement, aucun résident n'est affecté (lignes 1 à 4). Chaque résident non affecté et qui n'est pas désespéré (ligne 5) se propose à l'hôpital qu'il préfère. L'hôpital affecte ce nouveau résident (ligne 8 et 9). En cas de surcharge (ligne 10), l'hôpital débauche le pire de ses résidents (lignes 14 et 15). Si l'hôpital est plein (ligne 16), celui-ci supprime de sa liste de préférences tous les résidents moins bons que le pire résident qui lui est actuellement affecté (ligne 20) et informe ces derniers (ligne 21).

Algorithme 3 : RGS

Données : un problème $HR = \langle H, R \rangle$
Résultat : un appariement M

```
1  pour tous les r ∈ R faire
2      a_M(r) ← θ ;
3  pour tous les h ∈ H faire
4      p_M(h) ← ∅ ;
5  pour tous les r ∈ R, a_M(r) = θ ∧ |r.π| ≠ 0 faire
6      h ← r.π[0] ;
7      //r postule à h
8      p_M(h) ← p_M(h) ∪ {r} ;
9      a_M(r) ← h ;
10     si |p_M(h)| > h.q alors
11         //h est surchargé
12         r_2 ← min_p(h) ;
13         //Le moins bon résident (r_2) est débauché
14         p_M(h) ← p_M(h) \ {r_2} ;
15         a_M(r_2) ← θ ;
16     si |p_M(h)| = h.q alors
17         //h est plein ;
18         r_2 ← min_p(h) ;
19         pour tous les r_3 ∈ h.π, r_2 ≻_h r_3 faire
20             h.π.supprimer(r_3) ;
21             r_3.π.supprimer(h) ;
22 retourner M ;
```

Dans HGS (cf. algorithme 4), initialement, aucun résident n'est affecté (lignes 1 à 4). Chaque hôpital sous chargé pour lequel au moins un résident de sa liste de préférences ne lui est pas affecté (ligne 5) propose une place au meilleur d'entre eux (lignes 6 à 10). Si ce résident est déjà dans un autre hôpital (ligne 12) alors on le désaffecte (lignes 13 et 15). Ce résident est affecté à ce nouvel hôpital (lignes 17 et 19), puis il supprime de ses listes de préférences tous les hôpitaux moins bons (ligne 21) et les informe de son choix (ligne 22).

Dans [2], Gusfield et Irving démontrent que toutes les solutions atteintes par RGS et HGS sont admissibles et que RGS (resp. HGS) atteint une solution resident-optimal (resp. hôpital-optimal).
Les solutions admissibles d'une même instance de HR possèdent les propriétés suivantes : i) chaque hopital possède le même nombre de résidents ; ii) les mêmes

Algorithme 4 : HGS

Données : un problème $HR = \langle H,\ R \rangle$
Résultat : un appariement M

```
1  pour tous les r ∈ R faire
2      a_M(r) ← θ ;
3  pour tous les h ∈ H faire
4      p_M(h) ← ∅ ;
5  pour tous les h ∈ H, |p_M(h)| < h.q ∧ ∃r_h ∈ h.π, a_M(r_h) ≠ h faire
6      i ← 0 ;
7      r ← h.π[i] ;
8      tant que a_M(r) = h faire
9          i ← i + + ;
10         r ← h.π[i] ;
11     //si r est déjà affecté
12     si a_M(r) ≠ θ alors
13         h_2 ← a_M(r) ;
14         //r est débauché
15         p_M(h_2) ← p_M(h_2) \ {r} ;
16     //h propose une place à r
17     p_M(h) ← p_M(h) ∪ {r} ;
18     //r est affecté
19     a_M(r) ← h ;
20     pour tous les h_3 ∈ r.π, h ≻_r h_3 faire
21         r.π.supprimer(h_3) ;
22         h_3.π.supprimer(r) ;
23 retourner M ;
```

résidents sont non affectés ; iii) chaque hopital sous chargé emploie les même résidents.

Théorème 1 *[2] Soient un problème HR* $= \langle H, R\rangle$ *de taille (n,m) et deux appariements stables* M_1 *et* M_2 *pour HR :*

1. $\forall h \in H, |p_{M_1}(h)| = |p_{M_2}(h)|$;
2. $\forall r \in R, a_{M_1}(r) = \theta \rightarrow a_{M_2}(r) = \theta$;
3. $\forall h \in H, |p_{M_1}(h)| < h.q \rightarrow p_{M_1}(h) = p_{M_2}(h)$.

Exemple 3 *Soient les relations de préférence suivantes* $HR = \langle\{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}, \{h_1, h_2, h_3, h_4, h_5\}\rangle$ *avec* $h_1.q = 2$, $h_2.q = 3$, $h_3.q = 1$, $h_4.q = 1$, $h_5.q = 1$ *tel que :*

$h_1 \succ_{r_1} h_3$
$h_1 \succ_{r_2} h_5 \succ_{r_2} h_4 \succ_{r_2} h_3$
$h_1 \succ_{r_3} h_2 \succ_{r_3} h_5$
$h_1 \succ_{r_4} h_2 \succ_{r_4} h_4$
$h_3 \succ_{r_5} h_1 \succ_{r_5} h_2$
$h_3 \succ_{r_6} h_2 \succ_{r_6} h_1 \succ_{r_6} h_5$
$h_3 \succ_{r_7} h_4 \succ_{r_7} h_5 \succ_{r_7} h_1$
$h_5 \succ_{r_8} h_4$

$r_3 \succ_{h_1} r_7 \succ_{h_1} r_5 \succ_{h_1} r_2 \succ_{h_1} r_4 \succ_{h_1} r_6 \succ_{h_1} r_1$
$r_5 \succ_{h_2} r_6 \succ_{h_2} r_3 \succ_{h_2} r_4$
$r_2 \succ_{h_3} r_5 \succ_{h_3} r_6 \succ_{h_3} r_1 \succ_{h_3} r_7$
$r_8 \succ_{h_4} r_2 \succ_{h_4} r_4 \succ_{h_4} r_7$
$r_3 \succ_{h_5} r_7 \succ_{h_5} r_6 \succ_{h_5} r_8 \succ_{h_5} r_2$

On vérifie que, pour chaque hôpital, sa liste de préférences ne contient que les résidents pour lesquels cet hôpital est rationnel.

Considérons pour cette instance, l'appariement M_0 *avec :*
$a_{M_0}(r_1) = h_5$, $a_{M_0}(r_2) = h_3$, $a_{M_0}(r_3) = h_1$, $a_{M_0}(r_4) = h_2$, $a_{M_0}(r_5) = h_2$, $a_{M_0}(r_6) = h_1$, $a_{M_0}(r_7) = h_5$, $a_{M_0}(r_8) = h_4$.
M_0 *est instable, irrationnel et* h_5 *est surchargé* ($|p_{M_0}(h_5)| > q_5$) *car* (r_5, h_1) *est un couple bloquant (car* $r_5 \succ_{h_1} r_6$ *et* $h_1 \succ_{r_5} h_2$*) et il est irrationnel pour* r_1 *d'être affecté à* h_5 *qui n'est pas dans sa liste de préférence.*

Considérons les deux appariements M_1, M_2 *tels que :*

- $a_{M_1}(r_2) = h_1$, $a_{M_1}(r_3) = h_1$, $a_{M_1}(r_4) = h_2$, $a_{M_1}(r_5) = h_3$, $a_{M_1}(r_6) = h_2$, $a_{M_1}(r_7) = h_4$ *et* $a_{M_1}(r_8) = h_5$;
- $a_{M_2}(r_2) = h_3$, $a_{M_2}(r_3) = h_1$, $a_{M_2}(r_4) = h_2$, $a_{M_2}(r_5) = h_1$, $a_{M_2}(r_6) = h_2$, $a_{M_2}(r_7) = h_5$ *et* $a_{M_2}(r_8) = h_4$.

M_1 *et* M_2 *sont stables, rationnels et aucun hôpital n'est surchargé. On peut noter que* M_1 *est le résultat de l'algorithme RGS tandis que* M_2 *est le résultat de l'algorithme HGS.*

On peut remarquer que dans un problème HR les préférences des hôpitaux sont distinctes. Cependant, si nous n'associons plus aucune préférence aux hôpitaux mais uniquement un quota alors l'algorithme du dictateur aléatoire en série (`random serial dictatorship` (RSD)) proposé par Abdulkadiroglu et Sonmey [1] (cf algorithme 5) permet d'atteindre un appariement rationnel où aucun hôpital n'est surchargé. Initialement, aucun résident n'est affecté (lignes 1 à 4). Chaque résident r est choisi aléatoirement (ligne 6). Tant que r est libre et qu'il n'est pas désespéré (ligne 9), il postule à l'hôpital h qu'il préfère. Si h est sous-chargé (ligne 12) alors r est affecté à h (lignes 14 et 15). Sinon r ne considère plus son meilleur choix (ligne 18).

Algorithme 5 : Dictateur en série aléatoire (RSD)

Données : un problème $HR = \langle H,\ R \rangle$ où les hôpitaux n'ont pas de préférences
Résultat : un appariement M

```
pour tous les r ∈ R faire
    a_M(r) ← θ ;
pour tous les h ∈ H faire
    p_M(h) ← ∅ ;
tant que R ≠ ∅ faire
    //on choisit aléatoirement un résident
    r = random(R) ;
    //tant que r est libre et garde espoir
    tant que a_M(r) = θ ∧ |r.π| ≠ 0 faire
        h = r.π(0) ;
        //si le quota du premier choix de r n'est pas atteint
        si |p_M(h)| < h.q alors
            //r est affecté à h
            a_M(r) = h ;
            p_M(h) ← p_M(h) ∪ {r} ;
        sinon
            //r ne considère plus son premier choix
            r.π.remove(h) ;
    R ← R \ {r}
retourner M ;
```

2.4 Conclusion

Dans ce chapitre, nous avons formalisé et proposé des algorithmes pour trois problèmes abstraits d'affectation :

- le problème du mariage stable complet (SMC) ;
- le problème du mariage stable incomplet (SMI) ;
- le problème hôpitaux/résidents (HR).

Nous avons supposé ici que chaque individu, devant être affecté à un ou plusieurs partenaires, est associé à une liste de préférences (complète ou non) lui permettant de classer ses partenaires potentiels. Dans la suite de ce rapport, nous nous intéressons uniquement au problème HR. Les algorithmes HGS et RGS atteignent une solution admissible pour l'ensemble des instances de HR à la différence de RSD. Toutefois, les trois algorithmes favorisent une communauté au détriment de la seconde.

Dans le but d'étudier l'impact de chaque algorithme sur un problème réel, nous nous intéressons à l'affectation de stages aux Professeur des Écoles. Il est ainsi nécessaire de construire des listes de préférences à partir des contraintes et des inclinaisons exprimées par les utilisateurs.

3 Application pratique

Dans ce chapitre, nous nous intéressons à l'affectation des stages aux Professeurs des Écoles (PE) dans l'académie de Lille. Pour compléter la formation pratique des Professeurs des Écoles, l'IUFM (Institut Universitaire de Formation des Maîtres) répartit les stagiaires dans les différents établissements en fonction de leur desiderata et des contraintes des titulaires.

Pour faire face à l'augmentation du nombre de recrutement, l'IUFM souhaite automatiser cette procédure. Nous montrons ici que ce problème d'affectation peut être résolu à l'aide des algorithmes présentés précédemment.

Dans une première section 3.1, nous décrivons l'application. Par la suite, en 3.2 nous formalisons l'application à un problème PE. Dans la section 3.3, nous mettons en avant l'algorithme permettant de transformer un problème PE en un problème HR. Finalement, nous comparons les solutions atteintes par les différentes méthodes de résolution (cf section 3.4).

3.1 Description du problème

Dans cette section, nous nous sommes intéréssés à un problème réel : l'affectation de stages pour les Professeurs des Écoles (PE) en formation dans l'académie de Lille. Les différentes données du problème sont représentées en figure 1 dans un Modèle Conceptuel des Données (MCD).

Lors de leur formation à l'IUFM, chaque stagiaire doit effectuer trois stages, un lors de chaque trimestre. Réciproquement, chaque titulaire suffisamment qualifié pour encadrer ces stages peut accueillir au plus deux stagiaires par trimestre. L'académie est divisée en bassin. Chaque bassin regroupe un ensemble de villes dans lesquelles se trouvent les écoles. Pour être affecté, chaque stagiaire doit exprimer deux voeux, c.-à-d. sélectionner et classer deux bassins dans lesquels il souhaite effectuer ses stages. On retrouve en figure 1 ce découpage administratif et les voeux des stagiaires.

Les encadrants enseignent dans une classe qui regroupe un ou plusieurs niveaux, appelés sections qui sont regroupées en cycle. L'enseignement du premier degré (maternelle/élémentaire) est divisée en trois cycles :

- le `cycle 1` comprenant la très petite section (noté `TPS`), la petite section (noté `PS`) et la moyenne section (noté `MS`) ;
- le `cycle 2` comprenant la grande section (noté `GS`), le cours préparatoire (noté `CP`) et le cours élémentaire 1 (noté `CE1`) ;
- le `cycle 3` comprenant le cours élémentaire 2 (noté `CE2`), le cours moyen 1 (noté `CM1`) et le cours moyen 2 (noté `CM2`).

Quand un titulaire enseigne dans l'une des sections d'un cycle, on considère qu'il enseigne dans ce cycle. Cependant, il existe un cas particulier concernant la section `GS` qui si elle est associée à `MS` ou `PS` est considérée comme cycle 1. Dans la figure 1, on retrouve l'école associée à un titulaire à l'aide de l'association `Travaille` mais également les sections auxquelles il enseigne avec l'association `Enseigne`.

L'objectif du problème est d'affecter un stage par trimestre à chaque stagiaire en respectant ses voeux comme le montre l'association `Affectation` de la figure 1. De plus, l'académie de Lille veut minimiser les remboursements kilomé-

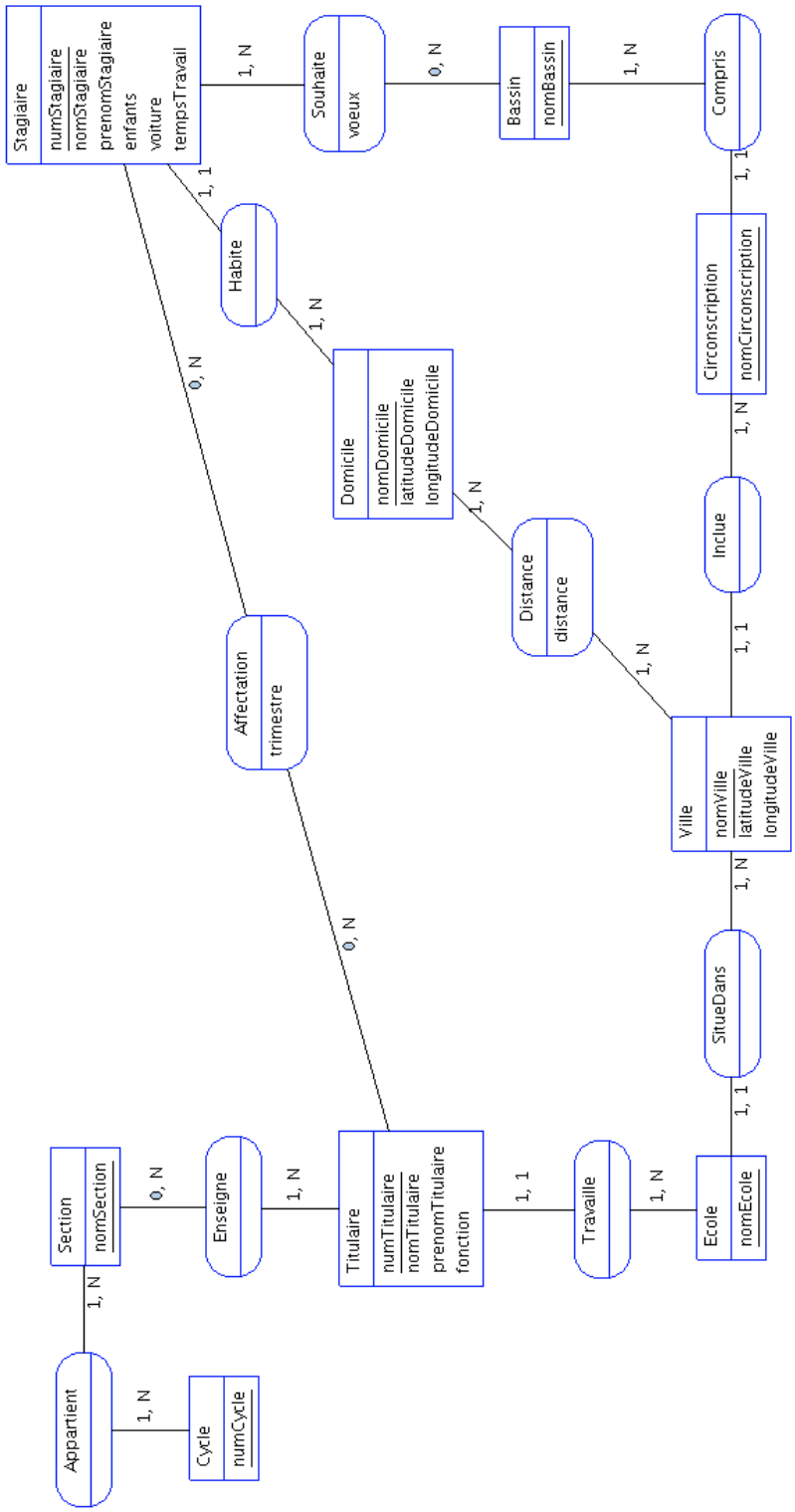

FIGURE 1 – MCD représentant les entrées et les sorties de la résolution d'un problème PE

triques des stages, c'est la raison pour laquelle nous avons ajouté dans le MCD les informations relatives aux distances. Nous avons fait le choix de différencier les domiciles des stagiaires, des villes contenant les écoles afin de minimiser le nombre de distances à calculer.

La priorité est donnée aux stagiaires ayant le plus d'enfants, puis à ceux travaillant à temps partiel et enfin à ceux n'ayant pas de véhicule. Réciproquement, le niveau de qualification des titulaires (`EMF`, `MAT CAFIPEMF` ou `MAT`) permet de les prioriser. Finalement, l'académie souhaite minimiser le nombre d'encadrants pour réduire les frais de gestions liés à leur dédommagement.

Par la suite, nous montrons que ce problème peut être assimilé à un ensemble de problèmes HR (cf section 2.3). Afin de générer les problèmes correspondants, nous formalisons ce problème dans le but de l'adapter au problème HR.

3.2 Formalisation

Afin de résoudre notre problème concret, nous formalisons un problème PE comme nécessitant trois appariements, un pour chaque trimestre.

Définition 11 (PE) *Un problème de Professeur des Écoles de taille (n, m) avec $n \geq 1$ et $m \geq 1$ est un couple $PE = \langle S, T \rangle$ avec $|S| = n$ et $|T| = m$, où :*
- *$S = \{s_1, \ldots, s_n\}$ est un ensemble de n stagiaires ;*
- *$T = \{t_1, \ldots, t_m\}$ est un ensemble de m titulaires.*

Nous décrivons ci-dessous les informations liées aux stagiaires et titulaires.

Un stagiaire a un certain nombre d'enfants, travaille à temps partiel (ou non), possède un véhicule (ou non) et associe deux listes de titulaires à ses deux voeux.

Définition 12 (Stagiaire) *Soit $PE =< S, T >$ un problème de taille (n, m). Un stagiaire $s \in S$ est un n-uplet $s =< nbE, tpsPartiel, vehicule, v_1, v_2, sec, c >$ où :*
- *$nbE \in [0; 5]$ correspond au nombre de ses enfants (0,1, ..., 5 ou plus) ;*
- *$tpsPartiel \in \{true, false\}$ est true si le stagiaire travaille à temps partiel, $false$ sinon ;*
- *$vehicule \in \{true, false\}$ est true si le stagiaire a un véhicule, $false$ sinon ;*
- *$v_1 \in \mathcal{P}(T)$ correspond aux titulaires considérés par le voeu 1 ;*
- *$v_2 \in \mathcal{P}(T)$ correspond aux titulaires considérés par le voeu 2 ;*
- *$sec \in \mathcal{P}(\{\texttt{TPS}, \texttt{PS}, \texttt{MS}, \texttt{GS}, \texttt{CP}, \texttt{CE1}, \texttt{CE2}, \texttt{CM1}, \texttt{CM2}\})$ correspond aux sections déjà pratiquées ;*
- *$c \in \mathcal{P}(\{\texttt{C1}, \texttt{C2}, \texttt{C3}\})$ correspond aux cycles déjà pratiqués.*

Dans le but de maximiser le nombre de stagiaires enseignant dans des sections différentes mais également dans les trois cycles d'enseignements, nous ajoutons deux attributs *sec* et *c* à chaque stagiaire afin de représenter respectivement les sections et cycles déjà pratiqués par le candidat.

Un titulaire possède une qualification et enseigne dans des cycles et sections.

Définition 13 (Titulaire) *Soit $PE =< S, T >$ un problème de taille (n, m). Un titulaire $t \in T$ est un n-uplet $t =< f, c, sec >$ où :*

- $f \in \{\texttt{EMF}, \texttt{MAT CAFIPEMF}, \texttt{MAT}\}$ *correspond à sa qualification ;*
- $c \in \mathcal{P}(\{\texttt{C1}, \texttt{C2}, \texttt{C3}\})$ *correspond aux cycles auxquels il enseigne ;*
- $sec \in \mathcal{P}(\{\texttt{TPS}, \texttt{PS}, \texttt{MS}, \texttt{GS}, \texttt{CP}, \texttt{CE1}, \texttt{CE2}, \texttt{CM1}, \texttt{CM2}\})$ *correspond aux sections auxquelles il enseigne.*

Chaque titulaire peut accueillir deux stagiaires par trimestre et ne considère que les stagiaires qui candidatent dans son bassin.

Dans le but d'évaluer les remboursements kilométriques, les distances séparant chaque stagiaire de chaque titulaire sont stockées en base de données.

Une solution à un problème PE correspond à un appariement par trimestre, c.-à-d. l'affectation de 0, 1 ou 2 stagiaires à chaque titulaire.

Définition 14 (Un appariement) *Un appariement* $W = \langle W_1, W_2, W_3 \rangle$ *pour le problème* $PE = \langle S, T \rangle$ *de taille (n, m) est représenté par* $s_{W_i} : S \to T \cup \{\theta\}$ *et* $e_{W_i} : T \to \mathcal{P}(S)$ *pour le trimestre* $i \in [1, 3]$ *tel que :*

1. $\forall s \in S,\ s_{W_i}(s) \in T \cup \{\theta\}$;
2. $\forall t \in T,\ e_{W_i}(t) \subseteq S$;
3. $\forall t \in T,\ |e_{W_i}(t)| \leq 2$;
4. $\forall s \in S,\ \{s\} \subseteq e_{W_i}(s_{W_i}(s))$;
5. $\forall t \in T,\ \forall s \in e_{W_i}(t),\ s_{W_i}(s) = t$.

Le stage trimestriel (s_{W_i}) d'un stagiaire est un titulaire éventuellement réduit à l'individu fantome (cf équation 1). Les étudiants trimestriels (e_{W_i}) des titulaires sont des ensembles de stagiaires potentiellement vides (cf équation 2), au maximum 2 (cf équation 3). Comme dans un problème HR, l'affectation est réciproque (cf équations 4 et 5).

3.3 Transformation de PE en HR

Un problème PE se transforme en trois problèmes HR où chaque problème correspond à un trimestre de PE. Afin de réaliser l'affectation des stagiaires pour un trimestre, il est nécessaire de générer le problème HR correspondant et de le résoudre.

Considérons une instance d'un problème PE. Afin de déterminer l'appariement trimestriel W_i, nous transformons ce problème PE en un problème HR pour en calculer un appariement stable M_i avec les algorithmes présentés dans la section précédente. Cet appariement trimestriel M_i permet d'en déduire W_i. Afin de générer le problème HR correspondant, nous considérons la mise en correspondance illustrée dans le tableau 1

PE	HR
Titulaire	Hôpital
Stagiaire	Résident
Préférences et contraintes	Listes de préférences
Capacité d'accueil d'un titulaire	Quota d'un hôpital

TABLE 1 – Correspondance entre PE et HR

Lors de la génération du problème HR correspondant, nous établissons les listes de préférences des hôpitaux et des résidents afin de répondre aux exigences de l'IUFM et de respecter la priorisation des stagiaires et des titulaires.

Pour permettre la correspondance entre un hôpital et un titulaire mais aussi entre un résident et un stagiaire, on définit une bijonction :

Définition 15 (Correspondance) *Soient un problème $PE = \langle S,\ T\rangle$ de taille (n, m) et un problème $HR = \langle H,\ R\rangle$ de taille (n,m). La mise en correspondance de PE et HR est définit par : $map : H \cup R \to T \cup S$ tel que :*
- $\forall r \in R, \exists!\ s \in S, map(r) = s$;
- $\forall h \in H, \exists!\ t \in T, map(h) = t$.

La fonction inverse est notée map^{-1}.

L'algorithme 6 représente la résolution d'un problème PE. Il traite le problème par trimestre (lignes 1 et 2) et initialise chaque appariement trimestriel (lignes 3 à 6). Pour chaque trimestre, un problème HR est généré (ligne 8) puis résolu (ligne 10). L'appariement M_i obtenu permet d'établir l'appariement trimestriel correspondant à W_i (lignes 12 à 21). Finalement on passe au trimestre suivant (ligne 23).

L'algorithme 7 permet de transformer un problème PE en un problème HR. Initialement, la correspondance entre les hôpitaux et les titulaires (lignes 2 à 6) et entre les résidents et les stagiaires (lignes 8 à 11) est réalisée. Par la suite on génére les listes de préférences des résidents (lignes 13 et 14) et des titulaires (lignes 16 et 17). Finalement, le problème HR est retourné (ligne 18).

Génération des listes de préférences pour les résidents. L'algorithme 8 génére la liste de préférences d'un résident. Tout d'abord, on initialise la liste de préférences d'un résident avec les hôpitaux correspondants à ces voeux (lignes 5 à 7). Ensuite ces hôpitaux sont répartis par cycle (lignes 9 à 16). Les hôpitaux d'un même cycle sont triés par voeu, fonction et enfin par distance (lignes 18 et 19). Ces différentes listes sont alors concaténées dans la liste de préférences du résident afin de favoriser les cycles qui n'ont pas encore été réalisé (lignes 21 à 29). Finalement, on supprime de ses préférences l'ensemble des hôpitaux qui correspondent à des titulaires enseignants à des sections déjà pratiquées (lignes 31 à 37).

L'algorithme 9 effectue le tri d'une liste d'hôpitaux correspondants à des titulaires d'un même cycle. On commence par répartir chaque hôpital dans deux listes correspondant à chacun des voeux (lignes 4 à 9). Chacune de ces listes est alors triée en fonction des qualifications des titulaires correspondants (lignes 11 à 21). Ensuite, les hôpitaux sont triés en fonction de la distance entre le domicile du stagiaire et l'école du titulaire correspondant. (lignes 22 à 24). Finalement, les différentes listes sont concaténées (lignes 25 et 26).

Génération des listes de préférences pour les hôpitaux. L'algorithme 10 génére la liste de préférences d'un hôpital. Cette liste de préférences est initialisée avec les résidents qui le considèrent (lignes 3 à 5). Les résidents sont classés en fonction du nombre d'enfants du stagiaire correspondant (lignes 8 à 13), puis en fonction de leur temps de travail (lignes 15 à 22) et finalement en fonction du fait qu'il possède (ou non) un véhicule (lignes 24 à 31). Finalement, ces listes sont concaténées (lignes 32 à 34).

Algorithme 6 : Résolution d'un problème PE

Données : un problème PE $= \langle S,\ T \rangle$ et trois problèmes $\mathrm{HR}_i = \langle H_i,\ R_i \rangle$
Résultat : un appariement $W =< W_1,\ W_2,\ W_3 >$

```
1  i ← 1 ;
2  tant que i ≤ 3 faire
3      pour tous les s ∈ S faire
4          s_{w_i}(s) = θ ;
5      pour tous les t ∈ T faire
6          e_{w_i}(t) = ∅ ;
7      //transformation de PE en HR
8      HR_i ← PEtoHR(PE) ;
9      //résolution de HR_i
10     M_i ← HR_i.solve() ;
11     //récupération des résultats de M_i dans W_i
12     pour tous les h ∈ H_i faire
13         pour tous les r ∈ p_{M_i}(h) faire
14             s ← map(r) ;
15             t ← map(h) ;
16             e_{W_i}(t) ← e_{W_i}(t) ∪ {s} ;
17     pour tous les r ∈ R_i faire
18         h ← a_{M_i}(r) ;
19         s ← map(r) ;
20         t ← map(h) ;
21         s_{W_i}(s) ← t ;
22     //passage au trimestre suivant
23     i ← i + + ;
24 retourner W ;
```

Algorithme 7 : PEtoHR

Données : un problème PE = $\langle S,\ T\rangle$
Résultat : un problème HR = $\langle H,\ R\rangle$

```
1  //mise en correspondance des hôpitaux et des titulaires
2  pour tous les t ∈ T faire
3      h = new Hospital() ;
4      HR.addHospital(h) ;
5      h.q ← 2 ;
6      map(h) ← t ;
7  //mise en correspondance des résidents et des stagiaires
8  pour tous les s ∈ S faire
9      r = new Resident() ;
10     HR.addResident(r) ;
11     map(r) ← s ;
12 //Génération des listes de préférences des résidents
13 pour tous les r ∈ R faire
14     r.π ← genPrefRes(r) ;
15 //Génération des listes de préférences des hôpitaux
16 pour tous les h ∈ H faire
17     h.π ← genPrefHosp(h,R) ;
18 retourner HR ;
```

Algorithme 8 : genPrefRes

Données : un résident r
Résultat : $r.\pi$

```
r.π ← [] ;
s ← map(r) ;
cycles ← [[], [], []] ;
//Génération de la liste des hôpitaux considérés par les résidents
pour tous les t ∈ (s.v1 ∪ s.v2) faire
    h ← map^-1(t) ;
    r.π.add(h) ;
//Trie des hôpitaux par cycle impliquant les titulaires
pour tous les h ∈ r.π faire
    t ← map(h) ;
    si C1 ∈ t.c alors
        cycles[0].add(h) ;
    sinon si C2 ∈ t.c alors
        cycles[1].add(h) ;
    sinon
        cycles[2].add(h) ;
//Chaque cycle est trié par voeu puis par fonction et enfin par distance
pour tous les 0 ≤ i ≤ 2 faire
    cycles[i] ← trieVoeuFonctionDistance(r, cycles[i]) ;
//Préférences ordonnées par cycle non réalisé
si C1 ∉ s.c alors
    si C2 ∉ s.c alors
        r.π ← concat(cycles[0], cycles[1], cycles[2]) ;
    sinon
        r.π ← concat(cycles[0], cycles[2], cycles[1]) ;
sinon si C2 ∉ s.c alors
    r.π ← concat(cycles[1], cycles[2], cycles[0]) ;
sinon
    r.π ← concat(cycles[2], cycles[1], cycles[0]) ;
//Suppression des hôpitaux qui correspondent à des titulaires impliqués dans une section déjà pratiquée
pour tous les r ∈ R faire
    s ← map(r) ;
    pour tous les h ∈ r.π faire
        t ← map(h) ;
        pour tous les sec ∈ t.sec faire
            si sec ∈ s.sec alors
                r.π.remove(h) ;
retourner s.π ;
```

Algorithme 9 : trieVoeuFonctionDistance

Données : un résident r et une liste d'hôpitaux l
Résultat : une liste d'hôpitaux triée par voeu, fonction et distance

```
1  s ← map(r) ;
2  V ← [[], []] ;
3  //Trie des hôpitaux par voeu des stagiaires
4  pour tous les h ∈ l faire
5      t ← map(h) ;
6      si t ∈ s.v_1 alors
7          V[0].add(h) ;
8      sinon
9          V[1].add(h) ;
10 //Chaque sous-liste est triée par fonction
11 pour tous les v ∈ V faire
12     F ← [[], [], []] ;
13     //Trie des hôpitaux par fonction des titulaires
14     pour tous les h ∈ v faire
15         t ← map(h) ;
16         si t.f == EMF alors
17             F.[0].add(h) ;
18         sinon si t.f == MAT CAFIPEMF alors
19             F[1].add(h) ;
20         sinon
21             F[2].add(h)
22     pour tous les f ∈ F faire
23         //Un tri rapide sur la distance
24         f ← trieParDistance(r, f) ;
25     v ← concat(F) ;
26 retourner concat(V) ;
```

Algorithme 10 : genPrefHosp

Données : un hôpital h et une liste de résidents R
Résultat : $h.\pi$

```
h.π ← [] ;
//Génération des listes de résidents considérés pour chaque hôpital
pour tous les r ∈ R faire
    pour tous les h ∈ r.π faire
        h.π.add(r) ;
E ← [[], [], [], [], [], []] ;
//Trie des résidents par nombre d'enfants des stagiaires correspondants
pour tous les r ∈ h.π faire
    s ← m_RS(r) ;
    si s.nbE ≥ 5 alors
        E[0].add(r) ;
    sinon
        E[5 − s.nbE].add(r) ;
//On priorise les résidents correspondants à des stagiaires à temps partiel
pour tous les e ∈ E faire
    Temps ← [[], []] ;
    pour tous les r ∈ e faire
        s ← map(r) ;
        si s.tpsPartiel= true alors
            Temps[0].add(r) ;
        sinon
            Temps[1].add(r) ;
    //On priorise les résidents correspondants à des stagiaires sans véhicules
    pour tous les temps ∈ Temps faire
        V ← [[], []] ;
        pour tous les r ∈ temps faire
            s ← map(r) ;
            si s.vehicule ≠ true alors
                V[0].add(r) ;
            sinon
                V[1].add(r) ;
        temps ← concat(V) ;
    enfant ← concat(Temps) ;
t.π ← concat(E) ;
retourner t.π ;
```

3.4 Expérimentations

Nous considérons ici la campagne 2012 d'affectation de stagiaires au sein de l'académie de Lille. Lors de cette campagne, 356 stagiaires doivent trouver trois stages, chacun encadré par l'un des 783 titulaires. Nous avons transformé cette instance du problème PE en un problème HR (comme décrit dans la section précédente) puis nous l'avons résolu avec les algorithmes RGS (cf algorithme 3) et HGS (cf algorithme 4). Nous avons également appliqué l'algorithme RSD (cf algorithme 5) qui ne considère pas les préférences des hôpitaux. Les trois algorithmes permettent de résoudre ce problème en une vingtaine de minutes. Les appariements de PE correspondants sont notés W_{RGS}, W_{RSD}, W_{HGS}.

Le tableau 2 permet de comparer les résultats obtenus par nos trois algorithmes. La partie haute du tableau présente les résultats du point de vue des stagiaires et la partie basse permet d'évaluer le coût de cette affectation.

Critère	W_{RGS}	W_{RSD}	W_{HGS}
Stagiaires avec 3 stages	356	356	356
Stagiaires avec 3 cycles	348	352	266
Stagiaires en voeu 1	185	186	190
Déplacement moyen (km)	22,26	22,24	29,29
Titulaires mobilisés	496	504	348

TABLE 2 – Résultats

On peut remarquer que les trois algorithmes permettent de pourvoir tous les stages. En effet, le nombre de titulaires est beaucoup plus important que le nombre de stagiaires. Contrairement à HGS, les algorithmes RGS et RSD permettent à la quasi-totalité des stagiaires de pratiquer les trois cycles car ils favorisent les stagiaires. Quelque soit l'algorithme utilisé, toutes les affectations sont conformes aux deux voeux exprimés par les stagiaires. Bien que les algorithmes RGS et RSD sont susceptibles de favoriser les stagiaires, ces algorithmes ne privilégient pas particulièrement le premier voeu des stagiaires. La distance moyenne entre le domicile d'un candidat et le lieu d'exercice de son stage est plus faible pour les appariements W_{RGS} et W_{RSD} qui favorisent les stagiaires. A l'inverse, appliquer HGS permet de réduire le coût de gestion des stages en réduisant le nombre de titulaires mobilisés.

Le tableau 3 illustre la répartition des affectations en fonction de la qualification des titulaires. Par exemple, pour l'algorithme RGS, 41,23 % des titulaires ayant la qualification EMF sont mobilisés.

Critère		W_{RGS}	W_{RSD}	W_{HGS}
Titulaires mobilisés	EMF	41,23 %	40,13 %	32,02 %
	MAT CAFIPEMF	39,13 %	38,41 %	27,05 %
	MAT	18,76 %	18,97 %	21,16 %

TABLE 3 – Répartition des affectations par qualification

Conformément aux exigences de l'application, les trois algorithmes mobilisent les titulaires les plus qualifiés. De plus, les algorithmes RGS et RSD mo-

bilisent plus spécifiquement les titulaires les plus qualifiés car ces algorithmes favorisent les stagiaires dont les listes de préférences sont générées en partie à l'aide de la qualification des titulaires considérés.

Pour mémoire, sont prioritaires les stagiaires ayant le plus d'enfants puis ceux travaillant à temps partiel et enfin ceux qui ne possèdent pas de véhicule. Le tableau 4 résume pour chacune de ces catégories de stagiaires le nombre de stagiaires concernés et la distance moyenne à parcourir pour chacun de ces stages. Nous ne considérons pas ici l'algorithme RSD qui ne permet pas de prendre en considération la priorisation des stagiaires.

Nb enfants	Temps de travail	Véhicule	Nb stag.	W_{RGS}	W_{HGS}
4 enfants	TPS	Sans	0	-	-
		Avec	0	-	-
	TC	Sans	1	2,01 km	20,20 km
		Avec	0	-	-
3 enfants	TPS	Sans	0	-	-
		Avec	0	-	-
	TC	Sans	2	5,84 km	67,90 km
		Avec	0	-	-
2 enfants	TPS	Sans	0	-	-
		Avec	0	-	-
	TC	Sans	1	45,79 km	45,80 km
		Avec	5	10,79 km	16,30 km
1 enfant	TPS	Sans	0	-	-
		Avec	1	6,92 km	73,90 km
	TC	Sans	12	17,03 km	20,20 km
		Avec	2	6,34 km	81,90 km
0 enfant	TPS	Sans	13	7,58 km	15,00 km
		Avec	9	19,39 km	16,30 km
	TC	Sans	202	27,43 km	35,70 km
		Avec	108	20,42 km	22,60 km

TABLE 4 – Distance moyenne par type de stagiaire

Les données de la campagne de 2012 ne possèdent que peu de stagiaires avec des enfants. Étant donné le faible nombre de stagiaires avec des enfants, nous ne sommes pas en mesure de vérifier que la distance moyenne pour un stage diminue quand le nombre d'enfants croît. Si on ne considère que les stagiaires sans enfant, on vérifie que les stagiaires à temps partiel sont favorisés par les deux algorithmes. Ce n'est pas le cas pour les stagiaires n'ayant pas de véhicule qui représentent deux tiers de la cohorte.

Le tableau 5 synthétise les résultats obtenus par les trois algorithmes.

Critère	RGS	RSD	HGS
3 stages	✓	✓	✓
3 cycles	✓	✓	X
Voeux	✓	✓	✓
Distance	✓	✓	X
Titulaires mobilisés	X	X	✓
Priorité à la qualification	✓	✓	X
Priorité aux enfants	?	X	?
Priorité au temps de travail	✓	X	✓
Prioritié aux sans véhicule	?	X	?

TABLE 5 – Comparaison des méthodes de résolution de HR appliquées à PE

D'après nos expérimentations, aucun algorithme ne semble satisfaire l'ensemble des critères. Malgré tout, RGS est susceptible de prioriser les stagiaires contrairement à RSD. Alors que RGS semble améliorer la qualité de l'affectation (couverture des cycles, minimisation de la distance à parcourir et qualification des encadrants), HGS est plus adapté pour réduire les coûts de gestion en minimisant le nombre de titulaires mobilisés. Il serait intéressant d'appliquer nos algorithmes aux campagnes précédentes pour infirmer ou confirmer ces premiers résultats.

La section suivante présente l'implémentation de l'application réalisée.

4 Implémentation

Notre implémentation a été réalisée à l'aide du langage Java et du langage SQL. Le développement est composé de quatre paquetages :

- le paquetage `org.hr` issu de la librairie DSMP permet de modéliser un problème HR et le résoudre ;
- le paquetage `fr.lifl.smac.pe.bd` implémente la connexion et les méthodes d'accès à la base de données ;
- le paquetage `fr.lifl.smac.pe.metier` permet de représenter un problème PE ;
- le paquetage `fr.lifl.smac.pe.transformation` permet de transformer un problème PE en problème HR.

La figure 2 représente les différentes relations entre ces paquets. Le paquetage `fr.lifl.smac.pe.metier` dépend de `fr.lifl.smac.pe.bd` qui permet d'instancier l'ensemble des objets représentant un problème PE. Le paquetage `fr.lifl.smac.pe.transformation` permet d'effectuer la transformation entre un problème PE et un problème HR. Ainsi, chaque instance de `PeSolver` est composée de trois instances de `PeToHr` correspondant à chacun des trimestres. Cette classe intermédiaire hérite de la classe `HR` et utilise les classes `TitulaireToHospital` et `StagiaireToResident` afin de résoudre PE par trimestre. Ces deux dernières classes héritent respectivement de `Hospital` et `Resident` et permettent de générer les listes de préférences des hôpitaux et des résidents.

La figure 3 représente les différentes classes du paquetage `org.hr` modélisant un problème HR. `HR` est composée de m `Hospital` et n `Resident`. Ces derniers héritent de la classe `HrEntity` regroupant l'ensemble des attributs identiques entre `Hospital` et `Resident`. Il est important de noter que les listes de préferences entre hôpital et résident sont modélisées dans la classe `HrEntity`. Un problème HR est résolu par la classe abstraite `HRSolver`. `RGS` et `HGS` héritent de `HRSolver` et implémentent l'interface `Solver`. Cette interface permet de s'assurer qu'un solver de problème HR est munie d'une méthode `run()`. Une solution à un problème HR est un objet de type `Assignment` représentant un appariement. Elle est composée de m instances de `Position`. Chaque position représente l'ensemble des p postes occupés par des résidents pour chaque hôpital.

La figure 4 représente les classes du paquetage `fr.lifl.smac.pe.metier`. Les relations entre ces classes sont équivalentes aux associations entre entités dans le MCD représentées dans la figure 1. Ainsi les classes `Stagiaire` et `Titulaire` héritent de la classe `Individu` et partagent un ensemble d'attributs (`id`, `nom`, `prenom`). De même, un problème PE est composé de n stagiaires et m titulaires et peut être résolu par `PESolver`. Une solution à un problème PE est un objet du type `Affectation` composée de $3m$ `Stage` correspondant à une liste de 0 à 2 stagiaires affectés par trimestre à un titulaire.

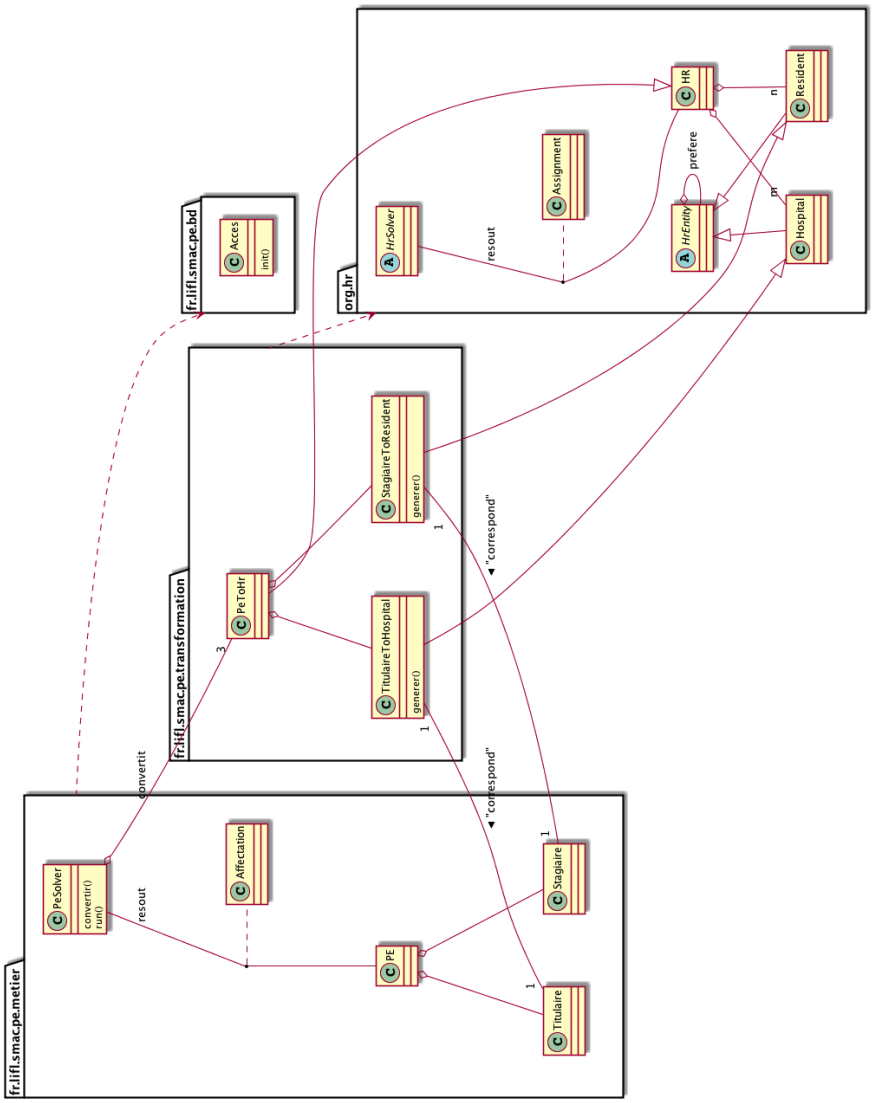

FIGURE 2 – Architecture de l'application

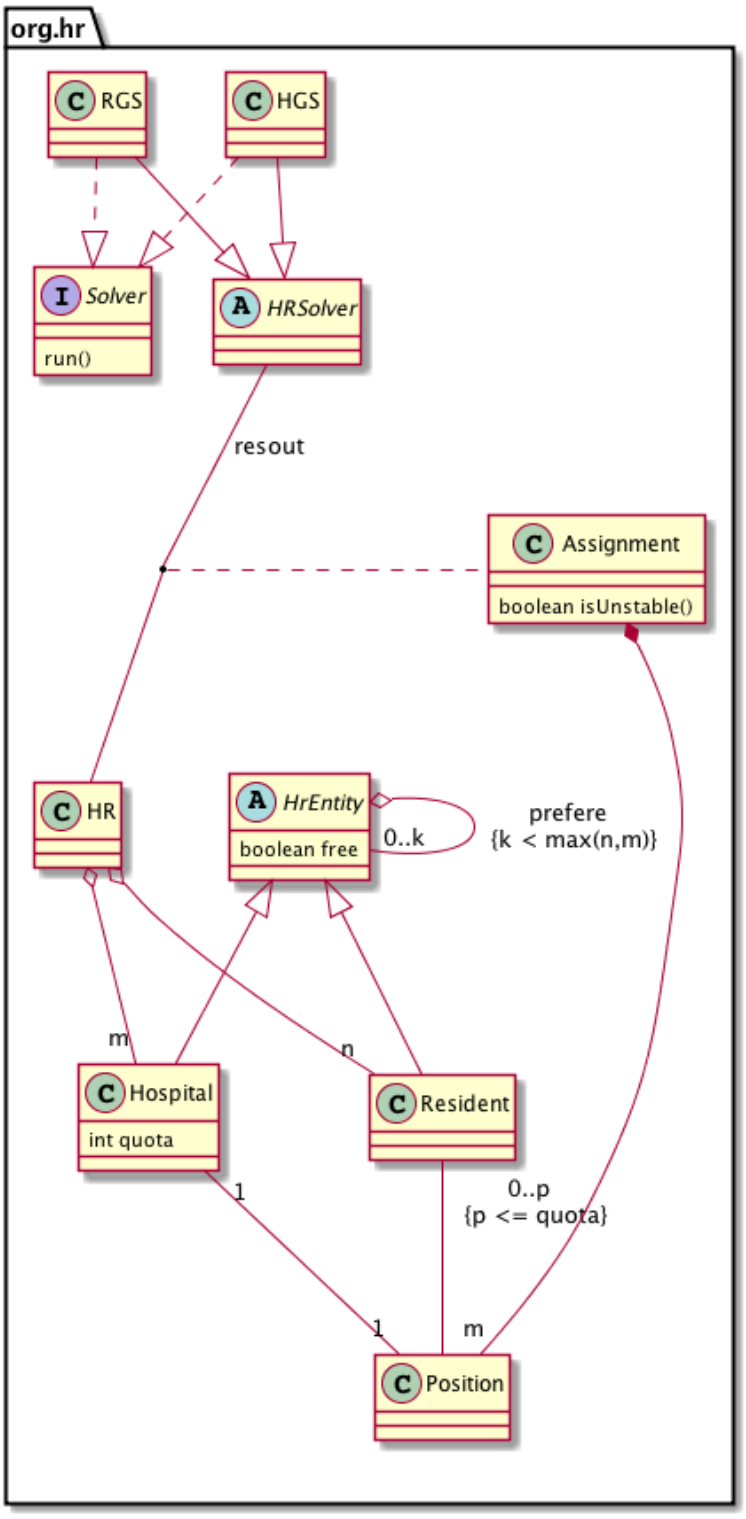

FIGURE 3 – Diagramme de classe de `org.hr`

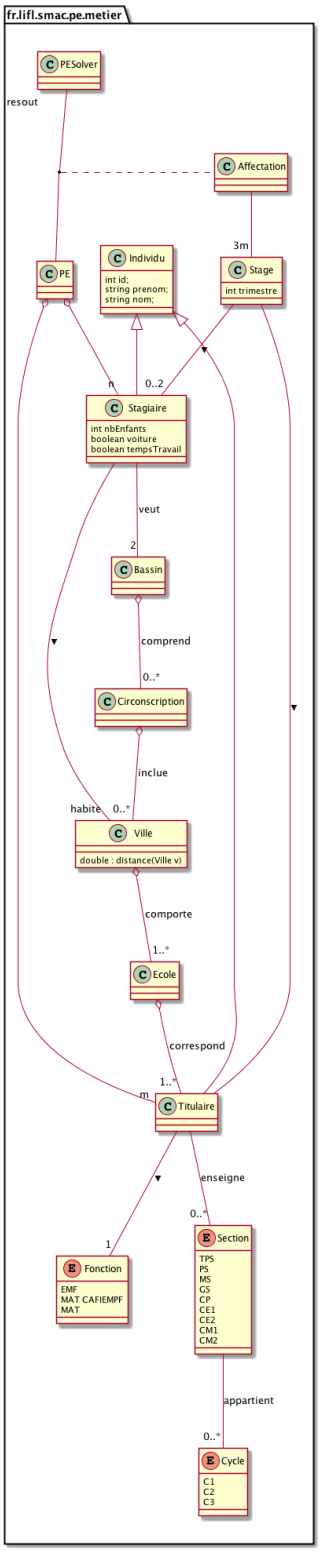

FIGURE 4 – Diagramme de classe de `lifl.fr.smac.pe.metier`

5 Conclusion

Nous nous sommes attaqués ici à une application pratique : l'affectation de stages pour les Professeurs des écoles en formation dans l'académie de Lille. Nous avons montré que ce problème concret peut être capturé par un problème bien connu : celui des hôpitaux/résidents. Nous avons présenté deux algorithmes qui permettent d'aboutir à des solutions stables pour ce problème abstrait où l'on suppose que les individus classent leurs partenaires potentiels : RGS (*Resident Gale-Shapley*) et HGS (*Hospital Gale-Shapley*). Toutefois, obtenir de telles listes de préférences n'est pas direct. D'après nos expérimentations, RGS semble améliorer la qualité de l'affectation, alors que HGS est plus adapté pour réduire les coûts de gestion. Il serait intéressant de confirmer ces premiers résultats à l'aide d'autres campagnes. Dans de futurs travaux, nous souhaitons utiliser l'algorithme Swing++ pour trouver un compromis entre ces deux objectifs.

Analyser les données et contraintes du problème afin d'établir un algorithme de résolution répondant aux demandes de l'`IUFM` a été l'un des aspects les plus complexes de ce stage. Une fois l'analyse effectuée, j'ai réalisé la documentation de l'application pour finalement la concevoir. Ainsi, respecter cette démarche scientifique a renforcé mes compétences de gestion de projet en me permettant chaque semaine de planifier avec mon encadrant les prochaines étapes. De plus, en début de stage j'ai eu l'occasion de travailler en collaboration avec l'`EMSE` (l'École Nationale Supérieure des Mines de St-Étienne) pour la conception de la méthode Swing++, ce travail d'équipe a été publié dans la conférence Modèles Formels de l'Interaction 2013 (MFI 13, cf annexe 1). En effet, la rédaction scientifique, qui est au coeur du métier de chercheur, a été la compétence que j'ai le plus développée. Ce stage m'a permis d'intégrer la vie d'une équipe de recherche en participant à différents séminaires, réunions d'équipes et aux Colloquium Pollaris. De plus, j'ai été membre de l'équipe organisatrice de la `PFIA` (Plate-Forme d'Intelligence Artificielle) qui s'est déroulée à Lille du 01 au 05 juillet. Cet événement a également été l'occasion de présenter mes travaux de recherches lors de la conférence MFI.

6 Annexe

6.1 MFI 2013

Les résultats obtenus à l'issue de mon projet de recherche ont été valorisés et diffusés dans l'article suivant [6] :

Swing++ : méthode multi-agents pour la résolution du problème des mariages stables

É. Piette⋆ M. Morge⋆
eric.piette@etudiant.univ-lille1.fr maxime.morge@lifl.fr

G. Picard†
gauthier.picard@emse.fr

⋆Laboratoire d'Informatique Fondamentale de Lille
Université de Lille 1
59655 Villeneuve d'ascq cédex

†École Nationale Supérieure des Mines de Saint-Étienne
158 cours Fauriel
42023 Saint-Étienne cédex 2

Résumé :
Nous nous intéressons ici au problème classique des mariages stables. De récents algorithmes ont été proposés pour résoudre ce problème en prônant l'équité des solutions. C'est le cas de la méthode Swing. Malheureusement, cette méthode peut ne pas se terminer pour certaines instances de problémes. Dans cette article, nous étendons cette méthode pour détecter les éventuels dilemmes à l'origine des cycles d'exécution et pour les résoudre. Notre implémentation est distribuable et, comme Swing, elle prône l'équité.

Mots-clés : Négociation, Théorie du choix social, Marchés artificiels

Abstract:
We are interested in the well-known stable marriage problem. Recently, some methods have been proposed which promote the fairness of the outcome, in particular Swing. Unfortunately, the latter does not terminate for all problem instances. In this paper, we propose Swing++ which detects the dilemmas from where the non-termination come from. Additionally, Swing++ solve them. Finally, our implementation can be distributed and it promotes the fairness as Swing.

Keywords: Negotiation, Social choice theory, Artificial markets

1 Introduction

Nous nous intéressons ici à un problème bien connu en informatique [6] et en économie [9], celui des mariages stables – en anglais, *Stable Marriage Problem* (SMP). Dans ce problème, l'objectif est de former des couples en prenant en compte les préférences que possède chaque individu vis-à-vis de leurs partenaires potentiels. L'algorithme de Gale-Shapley (GS) [4] est une preuve constructive de l'existence d'une solution pour toute instance du problème. Les solutions de GS ne sont pas nécessairement optimales du point de vue utilitaire. Toutefois, de récents travaux ont abordé cette problématique en se basant sur des techniques qui prônent l'équité (SML2 [5], ZigZag [10] et Swing [3]).

Dans cet article, nous nous intéressons à Swing [3] car c'est, à notre connaissance, la seule méthode distribuable [2] permettant d'atteindre une solution stable. Malheureusement, Swing, peut ne pas se terminer pour certaines instances du problème. En effet, cette méthode peut mener à des situations de dilemme. Dans ce papier, nous étendons Swing pour : i) détecter les éventuels dilemmes ; ii) et les résoudre. Comme Swing, notre méthode est distribuable et elle prône l'équité.

Nous commençons par une formalisation du problème des mariages stables dans la section 2. Afin d'évaluer les solutions, nous introduisons la notion de bien être issue de la théorie du choix social (cf. section 3). Dans la section 4, nous présentons la méthode Swing et nous identifions les situations de dilemme qui provoquent la non-terminaison. Notre algorithme Swing++ est présenté dans la section 5. Nous exposons nos expérimentations en section 6. Enfin, nous terminons par une discussion.

2 Problème du mariage stable

Nous étudions ici le problème des mariages stables, en anglais *Stable Marriage Problem*, qui a été introduit par Gale et Shapley [4] en 1962 . Ce problème s'énonce simplement. On considère n individus d'une communauté (e.g la gent féminine) et n individus d'une autre communauté (e.g. la gent masculine). Chaque individu a des penchants pour les individus de l'autre partition avec lesquels il souhaite être apparié.

Définition 1 (SM) *Un **problème de mariage stable** de taille n (avec $n \geq 1$) est un couple SM $= \langle X, Y \rangle$ avec $|X| = |Y| = n$, où :*
- *$X = \{x_1, \ldots, x_n\}$ est un ensemble de n hommes, i.e. relations de préférence sur Y, représentant les préférences des hommes sur l'ensemble des femmes ;*
- *$Y = \{y_1, \ldots, y_n\}$ est un ensemble de n femmes, i.e. relations de préférence sur X, représentant les préférences des femmes sur l'ensemble des hommes.*

On note $y_2 \succ_{x_1} y_3$ le fait que l'homme x_1 préfère strictement la femme y_2 à la femme y_3. Dans cet article, on suppose que les relations de préférences sont des ordres stricts, ce qui signifie que chaque individu est en mesure de choisir entre deux partenaires (il n'y a ni ex æquo, ni indifférence).

Résoudre un problème SM consiste à assortir ces communautés. On appelle appariement, en anglais *matching*, un ensemble de n mariages monogames et hétérosexuels.

Définition 2 (Appariement) *Un **appariement** M pour le problème SM $= \langle X, Y \rangle$ de taille n est une application $\mu_M : X \cup Y \to X \cup Y \cup \{\theta\}$ [1] telle que :*
- *$\forall x \in X, \mu_M(x) \in Y \cup \theta$ et $\forall y \in Y, \mu_M(y) \in X \cup \theta$*
- *$\forall z \in X \cup Y$, si $\mu_M(z) \neq \theta$ alors $\mu_M(\mu_M(z)) = z$*

$\mu_M(z)$ représente le partenaire de z selon l'appariement M. Dans un appariement, chaque individu est marié et l'application μ est involutive et donc bijective. $\mu_M(z) \succ_z \mu_{M'}(z)$ signifie que z préfère (son partenaire dans) l'appariement M à (son partenaire dans) M'. $\mu_M(z) \succeq_z \mu_{M'}(z)$ si $\mu_M(z) \succ_z \mu_{M'}(z)$ ou $\mu_M(z) = \mu_{M'}(z)$. Nous supposons ici que les relations de préférences sont complètes et que les individus préfèrent être mal accompagnés que seuls ($\mu_M(z) \succ_z \theta$ si $\mu_M(z) \neq \theta$).

Un appariement est stable s'il n'existe pas de couple qui préférerait être ensemble plutôt qu'avec leur partenaire actuel. En d'autres termes, il n'existe pas de relation extra-conjugale menaçant l'appariement. Ces relations prennent la forme de couples hypothétiques bloquants.

1. θ dénote l'individu fantôme.

Définition 3 (Stabilité) *Soient un problème SM $= \langle X, Y \rangle$ de taille n et M un appariement pour SMP. Un couple $(x_i, y_i) \in X \times Y$ est **bloquant** si $y_i \succ_{x_i} \mu_M(x_i)$ et $x_i \succ_{y_i} \mu_M(y_i)$. M est **instable** s'il existe un couple bloquant.*

Une solution à un problème SM de taille n est un appariement stable comportant n couples. L'algorithme GS [4] est une preuve constructive de l'existence systématique d'une solution.

Algorithme 1: GS orienté homme

Données : un problème SM = (X,Y)
Résultat : un appariement M

```
Tous les individus sont libres ;
tant que il existe un homme libre x faire
    y ← la première femme de la liste de x ;
    //x se propose à y
    x2 ← est le partenaire courant de y
    éventuellement θ ;
    si x est dans la liste de y alors
        Marier x et y ;
        //y dispose
        si x2 ≠ θ alors
            x2 ← libre ;
        pour chaque successeur x3 de x dans la
        liste de y faire
            Supprimer x3 de la liste de y ;
            //y ne concède plus
            Supprimer y de la liste de x3 ;
            //y informe les concernés
```

Dans l'algorithme GS (cf. algorithme 1) tel que nous le présentons, les hommes proposent et les femmes disposent. Une exécution consiste en une séquence de propositions des hommes envers les femmes. En d'autres termes, les hommes jouent le rôle de proposant et les femmes jouent le rôle de répondant. On dit que GS est orienté homme. Il peut être facilement adapté pour être orienté femme, en inversant leur rôle respectif. Dans ce cas, les femmes proposent et les hommes disposent.

Un appariement stable est le résultat d'un jeu entre deux communautés. Il peut être évalué du point de vue d'une communauté ou de l'autre.

Définition 4 (Préférence communautaire) *Soit un problème SM $= \langle X, Y \rangle$ de taille n. Soient M et M' deux appariements stables pour SM et $E \subseteq (X \cup Y)$. M **Pareto-domine***

*M' sur E ssi $\forall z \in E \ \ \mu_M(z) \succeq_z \mu_{M'}(z)$ et $\exists z \in E$ tq $\mu_M(z) \succ_z \mu_{M'}(z)$. Un appariement est **Pareto-optimal** si c'est un appariement stable qui n'est pas Pareto dominé sur $X \cup Y$. Un appariement est Pareto **mâle-optimal** (respectivement Pareto **femelle-optimal**) si c'est un appariement stable qui n'est pas Pareto-dominé sur X (respectivement Y).*

Donald Knuth a démontré dans [6] que l'exécution de l'algorithme GS orienté homme (resp. orienté femme) aboutit à une solution – un appariement stable – qui est mâle-optimale (resp. femelle-optimale) et que cette solution est Pareto-dominée sur les femmes (resp. les hommes) par les autres appariement stables.

Exemple 1 *Soient les relations de préférence suivantes $SM = \langle\{x_1, x_2, x_3\}, \{y_1, y_2, y_3\}\rangle$ avec :*

$y_2 \succ_{x_1} y_1 \succ_{x_1} y_3 \quad x_2 \succ_{y_1} x_1 \succ_{y_1} x_3$
$y_3 \succ_{x_2} y_2 \succ_{x_2} y_1 \quad x_3 \succ_{y_2} x_2 \succ_{y_2} x_1$
$y_1 \succ_{x_3} y_3 \succ_{x_3} y_2 \quad x_1 \succ_{y_3} x_3 \succ_{y_3} x_2$

Pour cette instance, l'appariement M ($\mu_M(x_1) = y_2$, $\mu_M(x_2) = y_1$, $\mu_M(x_3) = y_3$) est instable car (x_2, y_2) est un couple bloquant. À l'inverse, les trois appariements M_1, M_2 et M_3 sont stables :

– $\mu_{M_1}(x_1) = y_1$, $\mu_{M_1}(x_2) = y_2$, $\mu_{M_1}(x_3) = y_3$
– $\mu_{M_2}(x_3) = y_1$, $\mu_{M_2}(x_1) = y_2$, $\mu_{M_2}(x_2) = y_3$
– $\mu_{M_3}(x_2) = y_1$, $\mu_{M_3}(x_3) = y_2$, $\mu_{M_3}(x_1) = y_3$

On peut noter que M_2 est atteint par GS orienté homme et donc est mâle-optimal tandis que M_3, atteint par GS orienté femme, est femme-optimale. M_1 est Pareto-optimal mais il ne peut être atteint par aucun de ces algorithmes.

3 Critères sociaux

Afin d'évaluer la qualité d'une solution du point de vue collectif, nous utilisons la théorie du choix social [7]. Cette évaluation repose sur une mesure individuelle de la satisfaction des individus, qui est ensuite agrégée afin d'estimer la satisfaction de toute (ou partie de) la société.

Pour le problème SM, chaque individu (quelle que soit sa communauté) évalue sa satisfaction en fonction de ses préférences via son regret qui est : 0 s'il est marié avec le premier individu de sa liste ; 1 s'il est marié avec le deuxième individu de sa liste, etc. Moins un individu a de regret, plus sa satisfaction est grande.

Définition 5 (Satisfaction individuelle) *Soient un problème $SM = \langle X, Y\rangle$ de taille n et un individu z. Considérons l'appariement M. Si le rang de $\mu_M(z)$ dans z est k (avec $0 \leq k < n$), alors le **regret de** z **dans l'appariement** M (noté $regret(\mu_M(z))$) est k. La **fonction d'utilité** de l'individu z est la fonction $u_z : X \cup Y \rightarrow [0, 1]$ définie telle que :*

$$u_z(\mu_M(z)) = \begin{cases} \frac{(n-1)-k}{n-1} & \text{si } \mu_M(z) \neq \theta \\ 0 & \text{sinon} \end{cases}$$

La théorie du choix social propose plusieurs fonctions pour agréger les utilités des individus, comme par exemple le bien-être utilitaire. Nous adaptons cette mesure au problème SM afin d'évaluer les solutions du point de vue global. Nous introduisons la mesure de bien-être équitable pour évaluer les disparités entre communautés.

Définition 6 (Satisfaction collective) *Soit un problème SM de taille n. Considérons un appariement M. On appelle bien-être social une fonction qui associe à tous les individus (ou une partie) un niveau de satisfaction dans l'intervalle [0,1].*

– *Le **bien-être utilitaire** considère le bien-être de toute la société :*
$sw_u(X \cup Y) = \frac{1}{2n} \sum_{z \in X \cup Y} u_z(\mu_M(z))$
– *Le **bien-être masculin** considère la satisfaction des hommes :*
$sw_u(X) = \frac{1}{n} \sum_{x \in X} u_x(\mu_M(x))$
– *Le **bien-être féminin** considère la satisfaction des femmes :*
$sw_u(Y) = \frac{1}{n} \sum_{y \in Y} u_y(\mu_M(y))$
– *Le **bien-être équitable** considère l'équité entre les hommes et les femmes :*
$$sw_e(X \cup Y) = 1 - \frac{1}{n} \left| \sum_{x \in X} u_x(\mu_M(x)) - \sum_{y \in Y} u_y(\mu_M(y)) \right|$$

Les notions de bien-être présentées ici ont été normalisées (entre 0 et 1). Pour le bien-être utilitaire, masculin et féminin, plus le bien-être est grand, plus la solution est optimale du point de vue de la communauté envisagée. Pour le bien-être équitable, plus le bien-être est grand, plus la solution est probe. Nous avons choisi d'envisager l'équité entre les communautés plutôt que le bien-être égalitaire qui considère le bien-être de l'individu le plus insatisfait car cette notion est indépendante de la communauté. Pour une même instance de problème SM, la personne la

moins satisfaite par une solution peut être un homme et la personne la moins satisfaite par une autre solution peut être une femme. La comparaison des solutions à l'aide de cette métrique n'est donc ni simple ni directe.

Exemple 2 *Si nous considérons de nouveau l'exemple 1, les bien-être pour les différents appariements stables sont les suivants :*

Bien-être	M_1	M_2	M_3
$sw_u(X \cup Y)$	0.5	0.5	0.5
$sw_u(X)$	0.5	1	0
$sw_u(Y)$	0.5	0	1
$sw_e(X \cup Y)$	1	0	0

L'appariement M_1, qui ne peut pas être atteint par GS, est la solution la plus équitable.

4 Swing

Dans l'algorithme GS, une communauté joue le rôle de proposant, l'autre de disposant. Ainsi les proposants sont favorisés. Swing permet à chaque communauté d'adopter alternativement ces deux rôles afin d'atteindre un résultat plus équitable [3].

Chaque individu est représenté par un agent dont l'état interne est défini comme suit.

Définition 7 (Agent) *Soient $SM = \langle X, Y \rangle$ un problème de taille n et un agent $a \in \mathcal{A}$ représentant l'individu $z \in X \cup Y$. À chaque instant, l'**agent** a est représenté par un tuple $a = \langle \sigma_a, \pi_a, \kappa_a, \mu_a \rangle$ où :*
- *$\sigma_a \in \{\top, \bot\}$ est le **statut marital** ($\top$ si marié, $\bot$ si célibataire) ;*
- *π_a est la **liste de préférence** basée sur la relation de préférence $\succ_z$ (cf. Def 2) ;*
- *$\kappa_a \in [0, n]$ est le **niveau de concession** ;*
- *$\mu_a \in X \cup Y \cup \{\theta\}$ est le **partenaire** courant.*

Quand l'agent est célibataire $\sigma_a = \bot$, son partenaire est fantomatique $\mu_a = \theta$. Par souci de concision, on considère $z \equiv a$ et $X \cup Y \equiv \mathcal{A}$. Nous notons $\pi_a(1)$ le premier individu de la liste de préférence, $\pi_a(2)$ le second, ainsi de suite. Pour tous individu λ, $\pi_a(k) = \lambda$ équivaut à $regret_a(\lambda) = k$. Ainsi le niveau de concession est le rang maximal de la liste de préférence qu'un individu considère comme acceptable à un instant donné. $\kappa_a = 1$ signifie qu'un individu ne considère acceptable que l'individu qu'il préfère. $\kappa_a = 0$ signifie qu'un individu est marié avec son partenaire préféré et qu'il n'est prêt à faire aucune concession. Initialement, $\sigma_a = \bot$, $\kappa_a = 1$, $\mu_a = \theta$ pour l'ensemble des individus. La liste de préférence π_a est spécifique à chaque individu.

Dans la méthode Swing, les hommes et les femmes se proposent alternativement (cf algorithme. 2).

Chaque proposant envoie une demande à l'ensemble des individus acceptables. Ainsi quand une proposition est acceptée par un disposant :

1. si le proposant est marié, il divorce ;
2. si le disposant est marié, il divorce ;
3. le proposant et le disposant se marient ;
4. le niveau de concession du proposant et celui du disposant sont modifiés de sorte qu'ils ne peuvent divorcer que pour de meilleurs partenaires.

Quand tous les individus sont mariés, Swing se termine [2]. Lorsqu'un agent divorce, son ancien partenaire devient célibataire et concède, c.-à-d. qu'il considère le partenaire suivant dans sa liste comme acceptable (cf algorithme 3).

Quand Swing s'arrête, il atteint une solution pour le problème : un appariement stable [3].

Malheureusement, la terminaison de Swing n'est pas garantie.

Exemple 3 *Considérons le problème $SM = \langle \{x_1, x_2, x_3\}, \{y_1, y_2, y_3\} \rangle$ avec les préférences suivantes :*

$y_3 \succ_{x_1} y_2 \succ_{x_1} y_1 \quad x_2 \succ_{y_1} x_1 \succ_{y_1} x_3$
$y_3 \succ_{x_2} y_2 \succ_{x_2} y_1 \quad x_1 \succ_{y_2} x_3 \succ_{y_2} x_2$
$y_1 \succ_{x_3} y_2 \succ_{x_3} y_3 \quad x_3 \succ_{y_3} x_2 \succ_{y_3} x_1$

Pour cet instance, les appariements stables sont :
- *M_1 avec $\mu_{M_1}(x_1) = y_2$, $\mu_{M_1}(x_2) = y_3$ et $\mu_{M_1}(x_3) = y_1$;*
- *M_2 avec $\mu_{M_2}(x_1) = y_2$, $\mu_{M_2}(x_2) = y_1$ et $\mu_{M_2}(x_3) = y_3$.*

Alors que M_1 est male-optimal (le résultat de l'algorithme GS orienté homme), M_2 est femelle-optimal (le résultat de l'algorithme GS orienté femme). Ces solutions peuvent être représentées dans un graphe (cf figure 1) où

2. Le problème du mariage stable avec listes incomplètes est hors de portée de cet article. Toutefois, on peut remarquer que Swing peut être adapté à ce type de problème en modifiant le test d'arrêt (cf ligne 2 dans l'algorithme 2)

Algorithme 2: Swing

Données : un problème SM = (X,Y)
Résultat : un appariement M

```
1  etape ← 0 ;
2  tant que il y a un individu libre faire
3      si etape est paire alors
4          proposants ← X ;
5      sinon
6          proposants ← Y ;
7      pour tous les p ∈ proposants faire
8          pour (i = 1 ; i ≤ p.κ ; i++) faire
9              d ← p.π(i) ;
10             //p se propose à d
11             si d.regret(p) ≤ d.κ alors
12                 //si d accepte
13                 si d.σ = ⊤ alors
14                     divorce(d) ;
15                 si p.σ = ⊤ alors
16                     divorce(p) ;
17                 p.μ ← d ;
18                 p.κ ← p.regret(d) − 1 ;
19                 d.μ ← p ;
20                 d.κ ← d.regret(p) − 1 ;
21                 break;
22             sinon
23                 //d rejette p
24         si p.σ = ⊥ alors
25             p.κ ← min(p.κ + 1, n) ;
26     etape ++ ;
```

Algorithme 3: divorce

Données : un agent a

```
1  divorcé ← a.μ ;
2  divorcé.κ ← min(regret_a(divorcé) + 1, n) ;
3  divorcé.σ ← ⊥ ;
```

chaque nœud correspond à un couple et un arc signifie que l'un des partenaires du couple origine menace le couple destination. Par exemple x_3 préfère y_1 et y_3 préfère être avec x_3. Ce graphe contient le circuit [x_3, y_3, x_2, y_1].

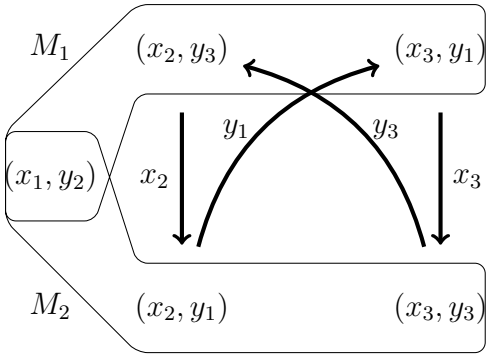

FIGURE 1 – Exemple de circuit

Les cycles d'exécution de Swing sont provoqués par le souhait des hommes de se diriger vers les solutions mâle-optimales et les femmes vers les solutions femelle-optimales. Ces situations de dilemme prennent la forme de circuit dans le graphe associé. On peut remarquer que dans un tel circuit le nombre d'hommes menaçants et le nombre de femmes menaçantes est identique et donc la taille d'un circuit est toujours paire.

Afin de modéliser la dynamique d'exécution de Swing, nous représentons ici l'état du système à chaque instant.

Définition 8 (État du système) *Soient $SM = \langle X, Y \rangle$ un problème de taille n et $\mathcal{A}$ l'ensemble des agents représentant les individus. À chaque instant, l'état du système est représenté par le couple $E = \langle \phi, \psi \rangle$ où :*
– *$\phi = \Pi_{a \in \mathcal{A}}(\sigma_a, \mu_a, \kappa_a)$ représente les différents agents ;*
– *$\psi = M$ si les hommes proposent ou $\psi = F$ sinon.*

Nous ne considérons ici que la partie dynamique des agents. Les préférences π_a restent inchangés. De plus, nous distinguons les étapes où les proposants sont les hommes et les étapes où les femmes proposent. Deux états E_1 et E_2 sont identiques si et seulement si les agents sont tous égaux ($\phi_1 = \phi_2$) et si ils correspondent à la même phase ($\psi_1 = \psi_2$). Une exécution peut être

représentée par un automate fini déterministe où un état correspond à un état du système et une seule transition relie chaque couple d'état. Dans cet automate, une boucle correspond à un cycle d'exécution.

Exemple 4 *Reprenons l'exemple 3. Comme représenté dans la figure 2, la résolution mêne à un dilemme.*

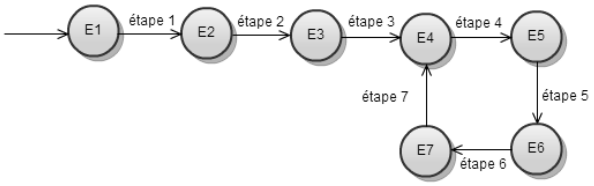

FIGURE 2 – cycle d'exécution de Swing

Ici, pour tous les états pairs $\psi = F$ et $\psi = M$ sinon. Dans la table 1, nous représentons les états du système pour notre exemple.

état	a	$x1$	$x2$	$x3$	$y1$	$y2$	$y3$
E_1	σ_a	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
	μ_a	θ	θ	θ	θ	θ	θ
	κ_a	1	1	1	1	1	1
E_2	σ_a	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
	μ_a	θ	θ	θ	θ	θ	θ
	κ_a	2	2	2	1	1	1
E_3	σ_a	$\top$	$\perp$	$\perp$	$\perp$	$\top$	$\perp$
	μ_a	y_2	θ	θ	θ	x_1	θ
	κ_a	1	2	2	2	0	2
E_4	σ_a		$\top$	$\perp$	$\perp$		$\top$
	μ_a	//	y_3	θ	θ	//	x_2
	κ_a		0	3	2		1
E_5	σ_a		$\perp$	$\top$	$\perp$		$\top$
	μ_a	//	θ	y_3	θ	//	x_3
	κ_a		2	2	3		0
E_6	σ_a		$\perp$	$\top$	$\top$		$\perp$
	μ_a	//	θ	y_1	x_3	//	θ
	κ_a		3	0	2		2
E_7	σ_a		$\top$	$\perp$	$\perp$		$\top$
	μ_a	//	y_3	θ	θ	//	x_2
	κ_a		0	2	2		1

TABLE 1 – Exécution de Swing

5 Swing++

Pour éviter les cycles d'exécution présentés dans la section précédente, nous proposons d'étendre Swing afin de détecter les dilemmes et de les traiter.
A chaque divorce, un agent mémorise son nouveau partenaire, c.-à-d son amant.

Définition 9 (Agent) *Soient $SM = \langle X, Y \rangle$ un problème de taille n et un agent $a \in \mathcal{A}$ représentant l'individu $z \in X \cup Y$. Nous définissons l'état de l'agent $a' = \langle \sigma_a, \pi_a, \kappa_a, \mu_a, \eta_a \rangle$ tel que :*
– *$a = \langle \sigma_a, \pi_a, \kappa_a, \mu_a \rangle$ (cf définition 7) ;*
– *$\eta_a \in X \cup Y \cup \{\theta\}$ représente l'amant.*

Initialement, $\eta_a = \theta$. Il n'est pas difficile de voir qu'il y a un cycle d'exécution de Swing s'il existe un état du système pour lequel on a un agent a avec $\eta^{2k}(a) = a$, avec $k \geq 2$. C'est le cas dans l'exemple 4 (cf table 1). Le circuit [x_3, y_3, x_2, y_1] est détecté quand l'étape E_4 est exécutée une seconde fois.

Dans Swing++ (cf algorithme 4), les agents qui sont impliqués dans un dilemme vont soit se sacrifier soit abandonner. Un sacrifice consiste à briser le circuit en n'envoyant pas de demande au partenaire potentiel mais à l'individu suivant dans la liste de préférences. Un abandon consiste à passer son tour de parole.

L'algorithme 5 permet de détecter les circuits quelque soit le nombre d'agents impliqués. Il permet de déterminer si le couple (o, d) est impliqué dans un circuit. Si le désir est réciproque ($o.\eta = d$), alors ce n'est pas le cas. Dans le cas contraire, soit o n'a pas d'amant et il n'y a pas de circuit. Sinon on ajoute le nouvel amant au $circuit$. Finalement, on vérifie que l'arc (o,d) est compris dans cette liste.

Pour résoudre un dilemme, les agents responsables du circuit doivent ne pas envoyer de proposition, on parle de sacrifice (cf algorithme 6). Ainsi, d'autres solutions soient explorées. Si le nombre d'agent ayant détecté le circuit est maximum, alors l'agent se sacrifie (et donc brise le circuit), le nombre de détection maximum augmente de 1 afin qu'un autre individu se sacrifie dans le cas où le même circuit se reforme. Sinon, il abandonne et incrémente de 1 le nombre de détection du circuit.

On peut noter qu'il est possible, au cours de l'exécution de Swing++, que deux individus célibataires se sacrifient jusqu'à se marier en rendant l'appariement instable. Afin de s'assurer que l'appariement calculé soit stable, nous vérifions lorsqu'on ajoute un mariage à l'appariement, que celui-ci reste stable (cf ligne 24 dans l'algorithme 4). Ainsi, ce test garantit la stabilité de la solution.

Algorithme 4: Swing++

Données : un problème SM
Résultat : un appariement M

```
1  etape ← 0 ;
2  nbDetect ← 0 ;
3  nbDetectMax ← 0 ;
4  tant que il y a un individu libre faire
5      si etape est paire alors
6          proposants ← X ;
7      sinon
8          proposants ← Y ;
9      pour tous les p ∈ proposants faire
10         pour (i = 1 ; i ≤ p.κ ; i++) faire
11             d ← p.π(i) ;
12             si detectionCircuit(p, d) alors
13                 //p et d dans un
                   circuit
14                 si
                   sacrifice(nbDetect,nbDetectMax)
                   alors
15                     //p concède
16                     p.η ← θ;
17                 sinon
18                     //p abandonne
19                     break ;
20             sinon
21                 //p propose à d
22                 si d.regret(p) ≤ d.κ alors
23                     //d accepte
24                     si appariementStable(p,d)
                       alors
25                         si d.σ = ⊤ alors
26                             divorce(d) ;
27                             d.η = p;
28                         si p.σ = ⊤ alors
29                             divorce(p) ;
30                             p.η = d;
31                         p.μ ← d ;
32                         p.κ ← p.regret(d) − 1 ;
33                         d.μ ← p ;
34                         d.κ ← d.regret(p) − 1 ;
35                     break ;
36                 sinon
37                     //d rejette p
38         si p.σ = ⊥ alors
39             p.κ ← min(p.κ + 1, n) ;
40     etape ++ ;
```

Algorithme 5: détectionCircuit

Données : individu o et un individu d.
Résultat : vrai si l'arc (o, d) est impliqué dans un circuit.

```
1  circuit ← [ ];
2  si o.η = d alors
3      // désir réciproque
4      retourner faux ;
5  origine ← o;
6  tant que o.η! = θ faire
7      si o.η = origine alors
8          // il y a un circuit
9          si d ∈ circuit alors
10             // contenant o et d
11             retourner vrai ;
12         sinon
13             retourner faux ;
14     si o.η ∈ circuit alors
15         // circuit sans o
16         retourner faux ;
17     circuit.add(o.η) ;
18     o ← o.η ;
19 retourner faux ;
```

Algorithme 6: sacrifice

Données : un nombre de détection $nbDetect$ et un maximum $nbDetectMax$

```
1  si nbDetect = nbDetectMax alors
2      //abandonne
3      nbDetectMax ++ ;
4      nbDetect ← 0 ;
5      retourner vrai ;
6  sinon
7      //concède
8      nbDetect ++ ;
9      retourner faux ;
```

État	a'	$x1$	$x2$	$x3$	$y1$	$y2$	$y3$
	σ_a	$\top$	$\top$	$\bot$	$\bot$	$\top$	$\top$
E_7	μ_a	y_2	y_3	θ	θ	x_1	x_2
	κ_a	1	0	2	2	0	1
	η_a	θ	y_3	y_1	x_2	θ	x_3
	σ_a		$\top$	$\bot$	$\bot$		$\top$
E_8	μ_a	//	y_3	θ	θ	//	x_3
	κ_a		0	3	2		1
	η_a		y_3	θ	x_2		x_3
	σ_a		$\bot$	$\top$	$\bot$		$\top$
E_9	μ_a	//	θ	y_3	θ	//	x_3
	κ_a		2	2	3		0
	η_a		y_3	θ	x_2		x_3
	σ_a		$\bot$	$\top$	$\top$		$\bot$
E_{10}	μ_a	//	θ	y_1	x_3	//	θ
	κ_a		2	0	1		2
	η_a		y_3	y_1	x_2		x_3
	σ_a		$\top$	$\top$	$\top$		$\top$
E_{11}	μ_a	//	y_3	y_1	x_3	//	x_2
	κ_a		0	0	2		1
	η_a		y_3	y_1	x_2		θ

TABLE 2 – Exécution de Swing++

Exemple 5 *Nous considérons ici le SM de l'exemple 3. Pour cette instance, l'exécution de Swing++ est identique à l'exécution de Swing pour les 7 premières étapes (cf table 1). Dans le tableau 2, nous représentons les états de E_7 à E_{10} avec les amants. Au cours de l'étape entre E_7 et E_8, x_3 détecte le circuit [x_3, y_3, x_2, y_1] et donc il se sacrifie pour le briser. Jusqu'à l'état E_{10} les agents adoptent la stratégie de concession minimale. Au cours de l'étape entre E_{10} et E_{11}, y_1 ne se propose pas à x_2 car il détecte un circuit. Comme c'est la seconde détection, y_1 abandonne. Ensuite, y_3 ne se propose pas à x_3 car il détecte de nouveau ce circuit. étant le second agent à faire cette détection pour la seconde fois, y_3 concède et elle se propose à x_2 qui accepte. L'appariement obtenu est M_1 qui est stable.*

6 Evaluation

L'ensemble des algorithmes évalués ici ont été implémenté en Java dans la librairie DSMP [3] (*Distributed resolution of Stable Marriage Problems*).

Dans nos expérimentations, nous générons de manière pseudo-aléatoire des instances du problème SM. Les listes de préférences sont des permutations indépendantes de la liste des partenaires potentiels. Pour chaque taille de problème, nous présentons pour chaque métrique et pour chaque méthode la moyenne obtenue en fonction de la taille du problème.

3. http://forge.lifl.fr/DSMP

6.1 Terminaison

Pour évaluer le taux de terminaison de Swing et de Swing++, nous traitons des instances de taille n où $n \in [2; 200]$. Pour chaque taille de problème, nous considérons $2 \times n$ instances et un nombre maximum de 1500 étapes pour Swing. Ce nombre d'étape a été fixé expérimentalement afin de nous assurer de la non-terminaison de Swing si celui-ci est atteint. Cette expérience a été réalisé sur 2 cœurs de 2.4 GHz utilisant 4 Go de mémoire vive. La durée de l'expérience a été de 156 heures. La figure 3 présente les taux de terminaison de Swing et de Swing++ pour différentes tailles de problèmes. On observe que la probabilité d'arrêt de la méthode Swing décroît avec la taille du problème. Alors que le taux de terminaison de Swing est élevé pour des problèmes de taille inférieure à 60 agents ($> 80\%$), le taux de terminaison est très faible pour des problèmes de grande taille. La probabilité qu'une instance fasse l'objet d'un dilemme ne semble pas linéaire par rapport à la taille du problème. Swing++ se termine pour l'ensemble des $40\ 000$ instances engendrées de manière (pseudo)-aléatoire mais ce résultat ne constitue pas une preuve de terminaison.

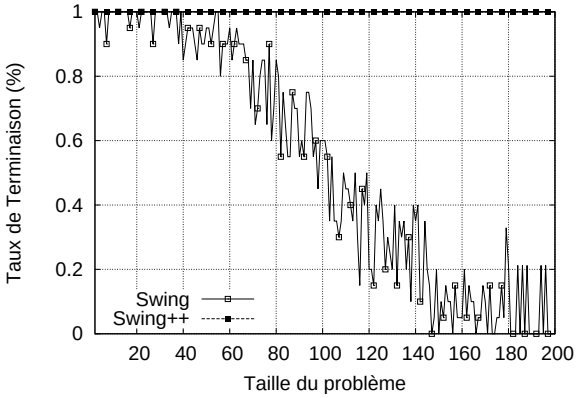

FIGURE 3 – Terminaison

6.2 Comparaison

Nous comparons ici Swing++ aux algorithmes existants en considérant comme métrique d'évaluation les différentes notions de bien-être social

introduites dans la section 3. Les algorithmes envisagés sont :

– ZigZag [10] qui consiste à parcourir une table dont les lignes (respectivement les colonnes) représentent les ordres de préférences des hommes (respectivement des femmes). Comme cet algorithme s'appuie sur cette table des mariages qui nécessite la connaissance complète du problème, cette méthode n'est pas distribuable ;
– SML2 [5] qui est un algorithme d'optimisation par recherche locale qui vise à minimiser le nombre de mariages non stables. Cet algorithme consiste, à partir d'un appariement aléatoire, à améliorer la stabilité de l'appariement en supprimant des mariages non stables. De part la notion de voisinage utilisée, cette méthode est centralisée. Dans nos expériences, nous avons configuré SML2 avec une probabilité de marche aléatoire $p = 0,2$ et un nombre maximum d'étapes $s = k \times n$, avec n la taille du problème et la constante $k = 10$.

Ici, nous traitons des problèmes de taille 2 à 100. Pour chaque taille de problème, 20 instances sont considérées.

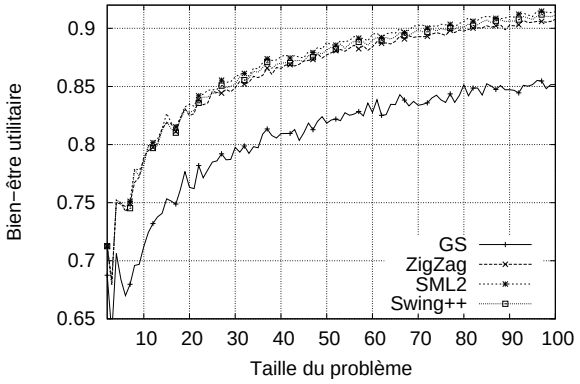

FIGURE 4 – Bien-être utilitaire

Les figures 4 et 5 représentent pour GS (orienté homme), SML2, ZigZag et Swing++ la moyenne du bien-être utilitaire et équitable respectivement. Nos expérimentations indiquent que, comme SML2 et ZigZag, Swing++ est équitable et contrairement à GS. De plus, les méthodes équitables sont plus optimales (du point de vue du bien être utilitaire). En effet, les solutions optimales ne sont pas forcément explorées par GS. Comme l'écart-type moyen pour chaque méthode est proche de 0,02 pour ces deux métriques, la différence entre les méthodes équitables et l'algorithme GS est significatif. Par contre, nous ne pouvons pas discriminer les méthodes équitables.

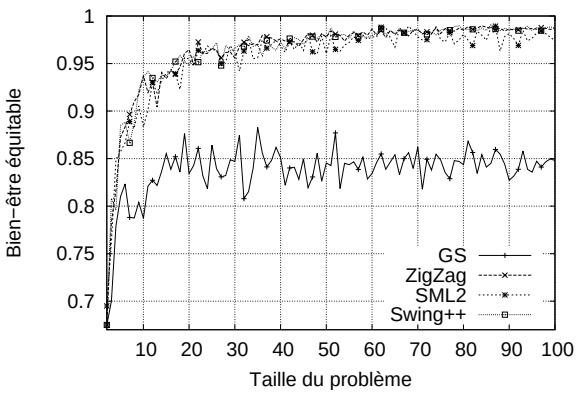

FIGURE 5 – Bien-être équitable

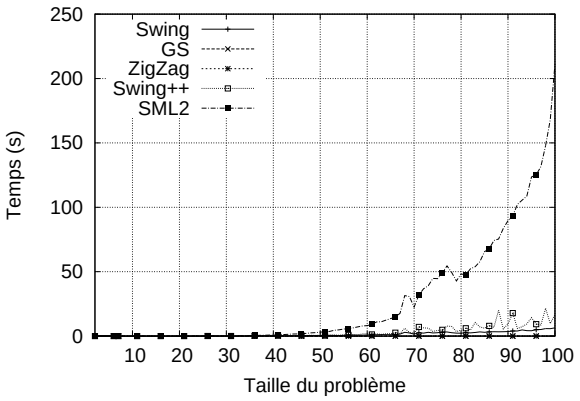

FIGURE 6 – Le temps d'exécution

La figure 6 représente le temps d'exécution moyen des différentes méthodes en fonction de la taille du problème. Contrairement à Swing++, le temps d'exécution de SML2 explose avec la taille du problème. Toutefois, Swing++ est moins rapide que ZigZag ou GS. C'est le prix de l'équité (GS n'est pas équitable) et de la stabilité (les solutions de ZigZag ne sont pas stables). Swing++ est légèrement plus lent que Swing à cause de la résolution des dilemmes.

6.3 Synthèse

Comme le montre le tableau 3, Swing++ est la seule méthode permettant d'obtenir systématiquement un appariement équitable et optimal. De plus, cette méthode est, comme GS, décentralisable, c.-à-d. qu'elle peut être implémentée à l'aide d'une plateforme multi-agents. Cette

implémentation est hors de portée de cet article.

Critère	GS	SML2	ZigZag	Swing	Swing++
Stabilité	✓	X	X	✓	✓
Terminaison	✓	X	✓	X	✓
Equité	X	✓	✓	✓	✓
Optimalité	X	✓	✓	✓	✓
Distribution	✓	X	X	✓	✓

TABLE 3 – Comparaison des méthodes résolvant SM

7 Conclusion

Dans ce papier, nous avons adopté une approche centrée individu pour la résolution du problème des mariages stables. L'algorithme distribuable de Gale-Shapley apporte une preuve de l'existence d'une solution stable pour chaque instance et il est distribuable. Toutefois, son asymétrie ne garantit pas d'atteindre de solutions équitables et optimales pour l'ensemble de la société. La méthode Swing, que nous avons étudié ici, prône l'équité en échangeant les rôles des agents à chaque étape. De plus, les individus négocient à l'aide de la stratégie de concession minimale afin d'obtenir un partenaire qui correspond à leurs préférences. Comme nous l'avons souligné ici, ces changements de rôles peuvent donner lieu à des cycles d'exécution de Swing. Alors que les hommes privilégient les solutions mâle-optimales lorsqu'ils sont proposants, les femmes privilégient les solutions femelle-optimales lorsque c'est à elles de proposer. Dans cet article, nous avons étendu Swing pour détecter ces dilemmes et réintroduire de l'asymétrie pour atteindre systématiquement une solution sans nuire à l'équité. Pour l'ensemble des $40\ 000$ expériences réalisées, Swing++ se termine.

Comme le remarque [8], les résultats de simulation sont considérés par la plupart des chercheurs comme une justification moins convaincante que les preuves formelles. Toutefois, l'étude analytique des systèmes complexes est parfois inaccessible. De même, la mise en oeuvre de la stratégie de concession minimale avec des préférences partielles est un problème difficile [1]. Si on considère que les membres d'une des communautés ne sont plus appariés avec un mais plusieurs partenaires, alors nous serons en mesure d'aborder un problème pratique comme celui de l'affectation des stages aux professeurs des écoles.

Références

[1] F. Delecroix, M. Morge, and J-C. Routier. Négociation bilatérale pour la recherche d'un compromis. In Journées Francophones sur les Systèmes Multi-Agents (JFSMA), 2013. (à paraître).

[2] P. Everaere, M. Morge, and G. Picard. Casanova : un comportement d'agent respectant la privacité pour des mariages stables et équitables. Revue d'intelligence artificielle (RIA), 26(5) :471–494, 2012.

[3] P. Everaere, M. Morge, and G. Picard. Minimal concession strategy for reaching fair, optimal and stable marriages. In Proc. of International Conference on Agents and Multiagent Systems (AAMAS), Saint Paul, Minnesota, USA, 2013. IFAMAS. (to appear).

[4] D. Gale and L. S. Shapley. College admissions and the stability of marriage. The American Mathematical Monthly, 69 :9–14, 1962.

[5] M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Local search algorithms on the stable marriage problem : Experimental studies. In Proc. of European Conference on Artificial Intelligence (ECAI), pages 1085–1086, Amsterdam, NL, 2010. IOS Press.

[6] D. Knuth. Mariages stables. Les Presses de l'Université de Montréal, Montréal, Canada, 1971.

[7] H. Moulin. Fair Division and Collective Welfare. MIT Press, Cambridge, Massachusetts, USA, 2003.

[8] T. Moyaux and P. McBurney. Centralised vs. market-based and decentralised decision-making : a review. Cybernetics & Systems, 43(7) :567–622, 2012.

[9] A. E. Roth and M. A. Oliveira Sotomayor. Two-sided Matching ; A Study in Game-theoretic Modeling and Analysis. Cambridge University Press, Cambridge, England, 1990.

[10] B. Zavidovique, N. Suvonvorn, and G. S. Seetharaman. A novel representation and algorithms for (quasi) stable marriages. In Proc. of International Conference on Informatics in Control, Automation and Robotics (ICINCO), pages 63–70, Barcelona, Spain, 2005. INSTICC Press.

6.2 SwingI++

L'algorithme proposée dans [6] a été adapté au problème du mariage stable avec listes incomplètes (SMI) décrit dans la section 2.2. Comme l'illustre l'algorithme 11, nous avons modifié le test d'arrêt.

Algorithme 11 : SwingI++

Données : un problème SMI
Résultat : un appariement M

```
1  etapeSansChangement ← 0 ;
2  etape ← 0 ;
3  nbDetect ← 0 ;
4  nbDetectMax ← 0 ;
5  tant que il y a un individu libre ou etapeSansChangement ≠ 2 faire
6      si etape est paire alors
7          proposants ← X ;
8      sinon
9          proposants ← Y ;
10     pour tous les p ∈ proposants faire
11         pour (i = 1 ; i ≤ p.κ ; i++) faire
12             d ← p.π(i) ;
13             si detectionCircuit(p, d) alors
14                 //p et d dans un circuit
15                 si sacrifice(nbDetect,nbDetectMax) alors
16                     //p concède
17                     p.η ← θ;
18                 sinon
19                     //p abandonne
20                     break ;
21             sinon
22                 //p propose à d
23                 si d.regret(p) ≤ d.κ alors
24                     //d accepte
25                     si appariementStable(p,d) alors
26                         si d.σ = ⊤ alors
27                             divorce(d) ;
28                             d.η = p;
29                         si p.σ = ⊤ alors
30                             divorce(p) ;
31                             p.η = d;
32                         p.μ ← d ;
33                         p.κ ← p.regret(d) − 1 ;
34                         d.μ ← p ;
35                         d.κ ← d.regret(p) − 1 ;
36                         etapeSansChangement ← −1;
37                     break ;
38                 sinon
39                     //d rejette p
40         si p.σ = ⊥ ∧ p.κ < p.π.taille alors
41             p.κ ← p.κ + 1 ;
42             etapeSansChangement ← −1;
43     etape ++ ; etapeSansChangement + +;
```

Références

[1] Atila Abdulkadiroglu and Tayfun Sonmez. School choice : A mechanism design approach. The American Economic Review, 93 :729–747, 2003.

[2] Gusfield D. and R.W. Irving. The stable marriage problem : structure and algorithms. MIT Press, Cambridge, MA, USA, 1989.

[3] D. Gale and L. S. Shapley. College admissions and the stability of marriage. The American Mathematical Monthly, 69 :9–14, 1962.

[4] D. Gale and M. Sotomayor. Some remarks on the stable matching problem. Discrete Applied Mathematics, 1 :223–232, 1983.

[5] D. Knuth. Mariages stables. Les Presses de l'Université de Montréal, Montréal, Canada, 1971.

[6] Éric Piette, Maxime Morge, and Gauthier Picard. Swing++ : méthode multi-agents pour la résolution du probléme des mariages stables. In Actes des Septiémes Journées Francophones sur les Modéles Formels de l'Interaction (MFI'13), 2013.

[7] A. E. Roth and M. A. Oliveira Sotomayor. Two-sided Matching ; A Study in Game-theoretic Modeling and Analysis. Cambridge University Press, Cambridge, England, 1990.