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

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**
  $y \leftarrow$ la première femme de la liste de x ;
  `//`x `se propose à` y
  $x_2 \leftarrow$ est le partenaire courant de y éventuellement θ ;
  si x est dans la liste de y **alors**
    Marier x et y ;
    `//`y `dispose`
    si $x_2 \neq \theta$ **alors**
      $x_2 \leftarrow$ libre ;
    pour chaque successeur x_3 de x dans la liste de y **faire**
      Supprimer x_3 de la liste de y ;
      `//`y `ne concède plus`
      Supprimer y de la liste de x_3 ;
      `//`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 :*
  $$sw_e(X \cup Y) = 1 - \frac{1}{n} |\sum_{x \in X} u_x(\mu_M(x)) - \sum_{y \in Y} u_y(\mu_M(y))|$$

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

```
etape ← 0 ;
tant que il y a un individu libre faire
    si etape est paire alors
        proposants ← X ;
    sinon
        proposants ← Y ;
    pour tous les p ∈ proposants faire
        pour (i = 1 ; i ≤ p.κ ; i++) faire
            d ← p.π(i) ;
            //p se propose à d
            si d.regret(p) ≤ d.κ alors
                //si d accepte
                si d.σ = ⊤ alors
                    divorce(d) ;
                si p.σ = ⊤ alors
                    divorce(p) ;
                p.μ ← d ;
                p.κ ← p.regret(d) − 1 ;
                d.μ ← p ;
                d.κ ← d.regret(p) − 1 ;
                break;
            sinon
                //d rejette p
        si p.σ = ⊥ alors
            p.κ ← min(p.κ + 1, n) ;
    etape ++ ;
```

Algorithme 3: divorce

Données : un agent a

```
divorcé ← a.μ ;
divorcé.κ ← min(regret_a(divorcé) + 1, n) ;
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].

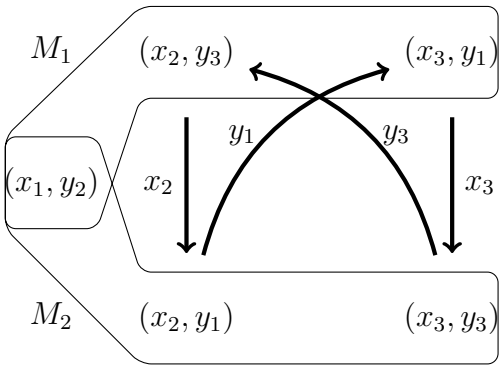

FIGURE 1 – Exemple de circuit

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 mène à un dilemme.*

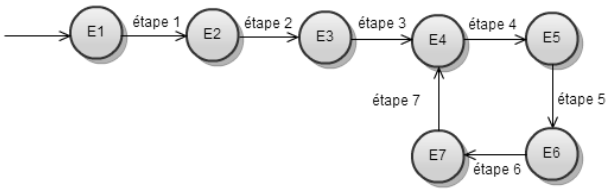

FIGURE 2 – cycle d'exécution de Swing

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.

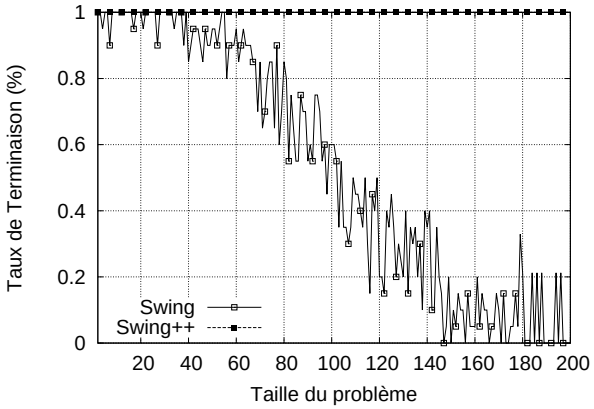

FIGURE 3 – Terminaison

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.

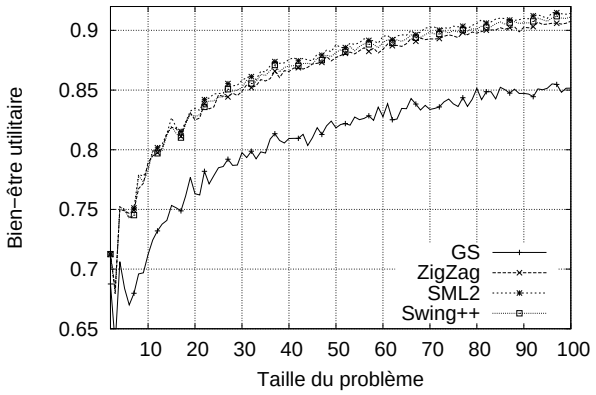

FIGURE 4 – Bien-être utilitaire

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.

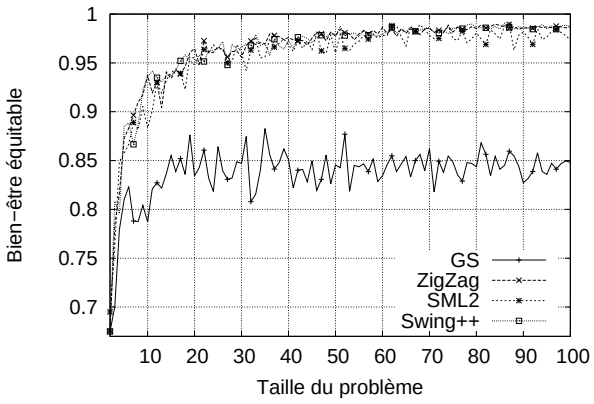

FIGURE 5 – Bien-être équitable

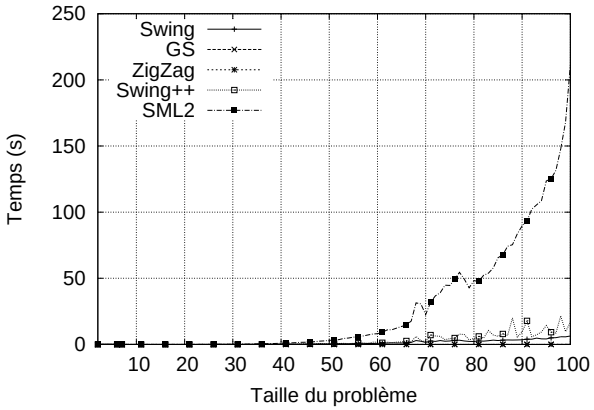

FIGURE 6 – Le temps d'exécution

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.