Affectation distribuée d'individus à des activités avec des préférences additivement séparables

Maxime Morge et Antoine Nongaillard

Univ. Lille, CNRS, SMAC/CRIStAL

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Université de Lille

cnrs

centralelille

Contexte scientifique

- *2015* PEPS MoMIS CNRS
  - **Objectif=** Modéliser les dynamiques de rétroaction dans un réseau social virtuel/réel
  - **Partenaires=** CLERSÉ (Lille1), CRIStAL (Lille1)
- *2016-2018* Programme Chercheurs Citoyens NPdC
  - **Objectif=** un réseau social numérique pour la cohésion sociale et lutter contre l'isolement
  - **Partenaires=** SCALab (Lille3), LAMIH (UVHC), CRIStAL (Lille1), CCAS (Lille)

Théorie du choix social : méthodes opérationnelles

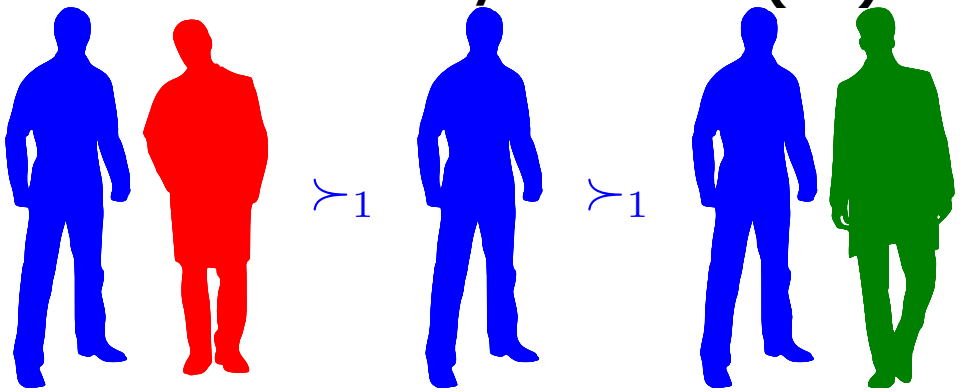
- Jeu hédonique [Dreze and Greenberg, 1980] :
  Notre problème $=$ spécialisation à deux faces
- Sélection d'activité de groupe [Igarashi et al., 2017] :
  $\neq$ jeu anonyme ou sur un réseau social
- Hôpitaux/résidents [Manlove, 2014] :
  Les préférences des individus portent sur leurs alter egos
- Quelle méthode de résolution ?
  - Optimisation quadratique (NP-difficile)
  - Recherche locale (nombreux optima locaux)
  - DCOP ($\neq$ scalable)
  - Jeu hedonique ($\neq$ modélisation)
  - Modélisation multi-niveaux [Nongaillard and Picault, 2016]

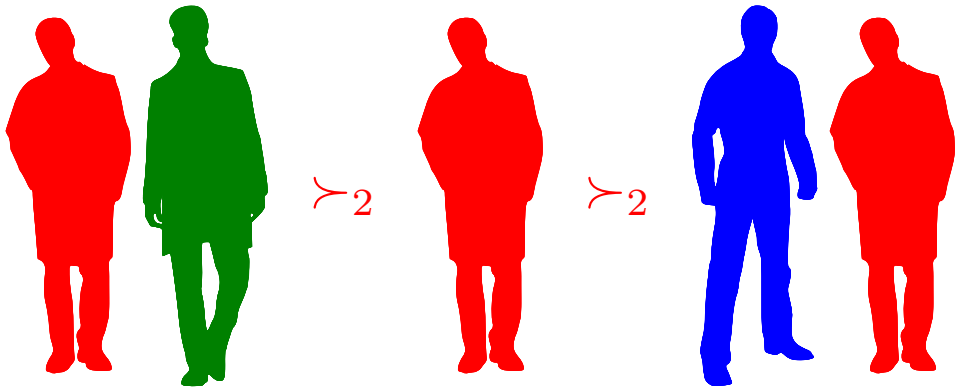
Problème Individus/Activités (IA)

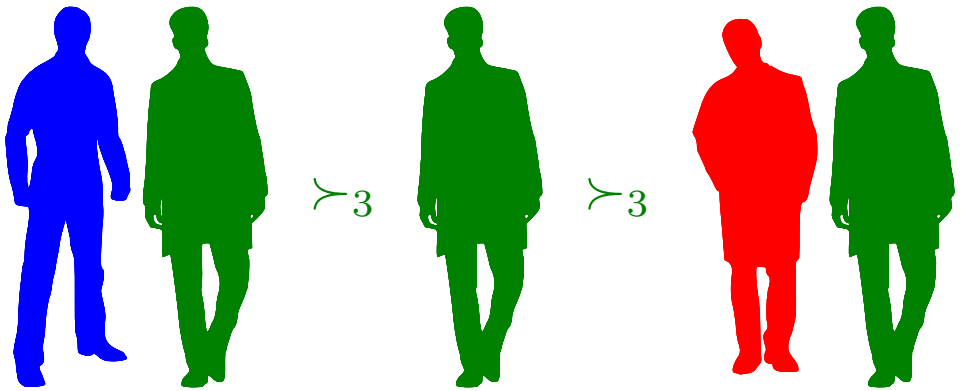
Un couple $IA = \langle I, A \rangle$ avec m individus et n activités, où :

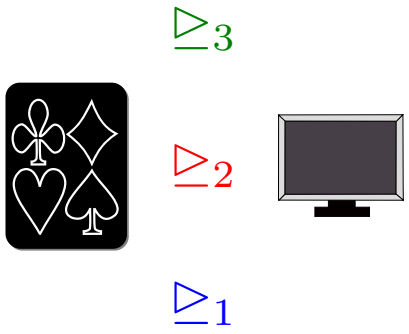
- $A = \{a_1, \ldots, a_n\}$ est un ensemble de n activités. Chaque activité a_j peut accueillir au plus c_j membres ;
- $I = \{1, \ldots, m\}$ est un ensemble de m individus. À chaque individu i correspond,
  1. une préférence sur les activités, i.e. une relation réflexive, complète et transitive $\trianglerighteq_i$ sur les activités $A \cup \{\theta\}$,
  2. une préférence hédonique, i.e. une relation de préférence réflexive, transitive et complète $\succsim_i$ sur l'ensemble des groupes auxquels il appartient.

Problème Individus/Activités (IA) : exemple









$\mathbf{c}(\clubsuit) = 2$

Appariement

- Une **coalition** est couple $C = \langle a, G \rangle$ où :
  - $a \in A \cup \{\theta\}$ est une activité ;
  - $G \subseteq I$ un groupe.

- Un **appariement** M est représenté par les fonctions $a_M : I \to A \cup \{\theta\}$ et $g_M : I \to \mathcal{P}(I)$ telles que :

$$\forall i \in I,\ a_M(i) \in A \cup \{\theta\} \tag{1}$$

$$\forall i \in I,\ i \in g_M(i) \subseteq I \tag{2}$$

$$\forall i \in I,\ a_M(i) = \theta \Rightarrow g_M(i) = \{i\} \tag{3}$$

$$\forall i \in I\ \forall j \in g_M(i),\ a_M(j) = a_M(i) \tag{4}$$

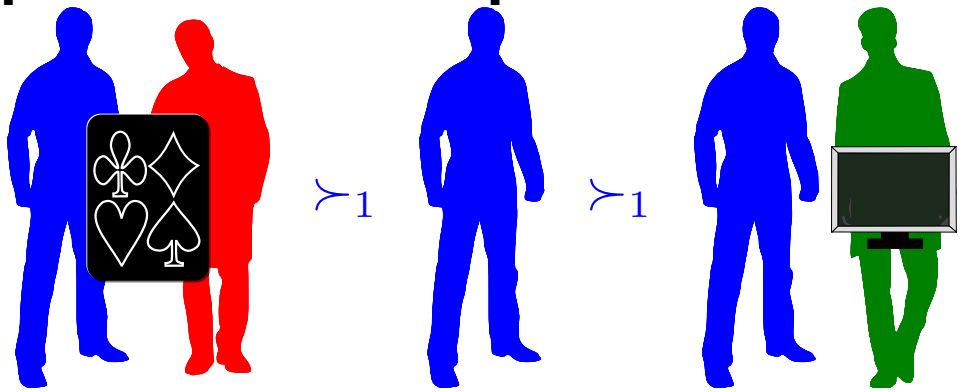
$$\forall i,j \in I,\ i \neq j \wedge a_M(i) = a_M(j) \neq \theta \Rightarrow g_M(i) = g_M(j) \tag{5}$$

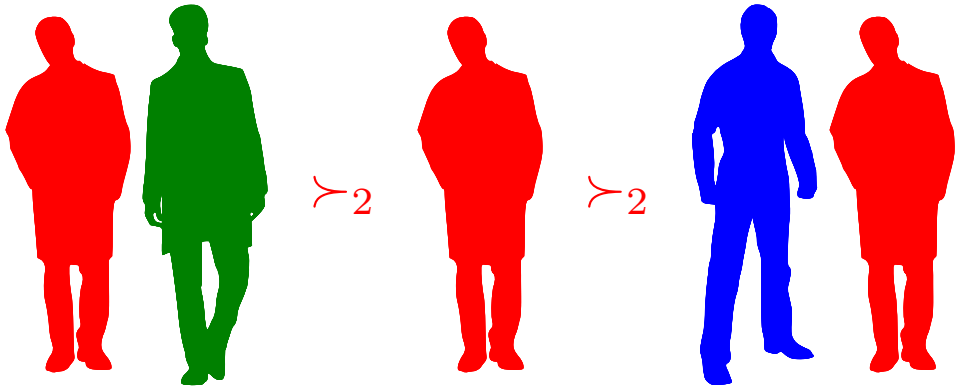
- La fonction de poste :

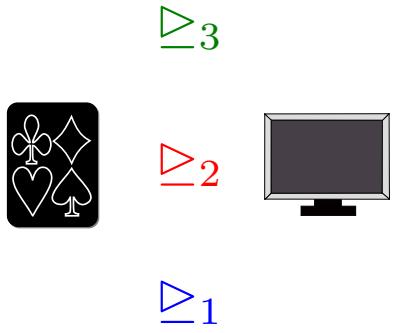
$$p_M : A \cup \{\theta\} \to \mathcal{P}(i)$$

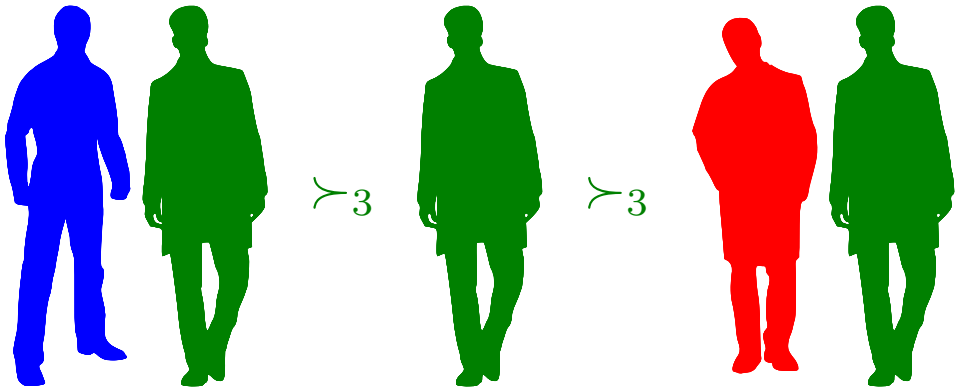
$$p_M(a) = \{i \in I; a_M(i) = a\} \tag{6}$$

Appariement : exemples



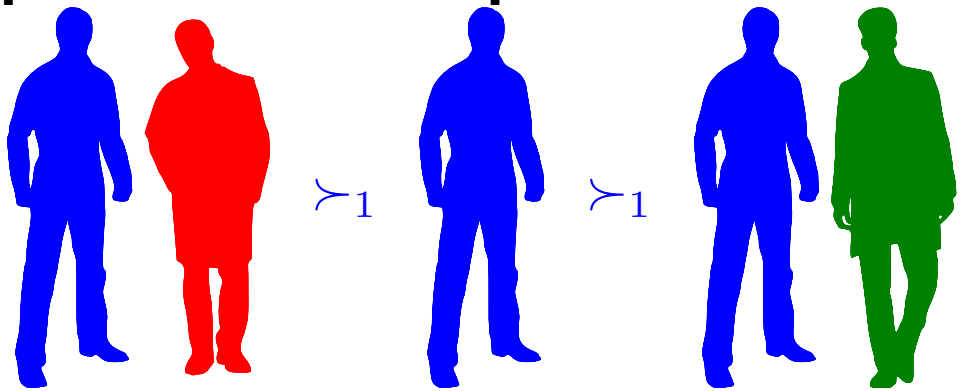


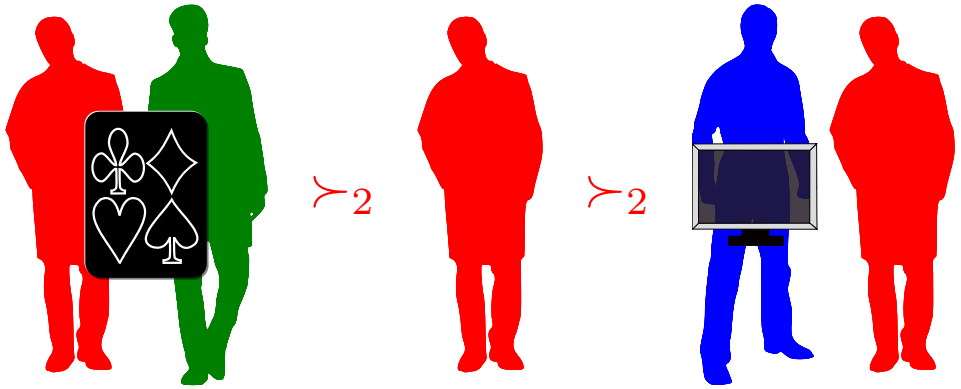


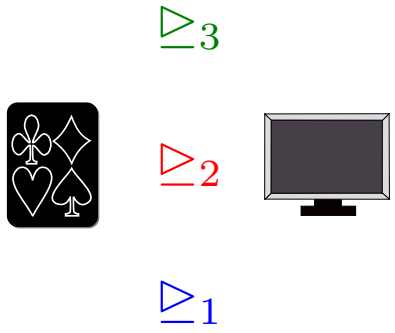


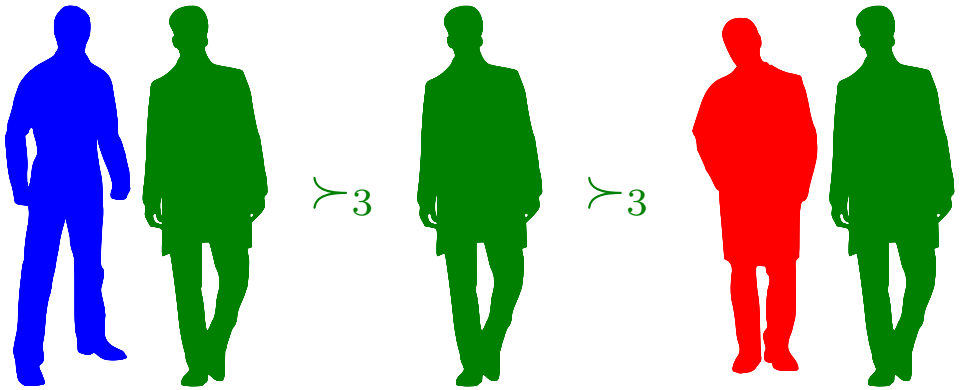
$\mathbf{c}($  $) = 2$

Appariement : exemples



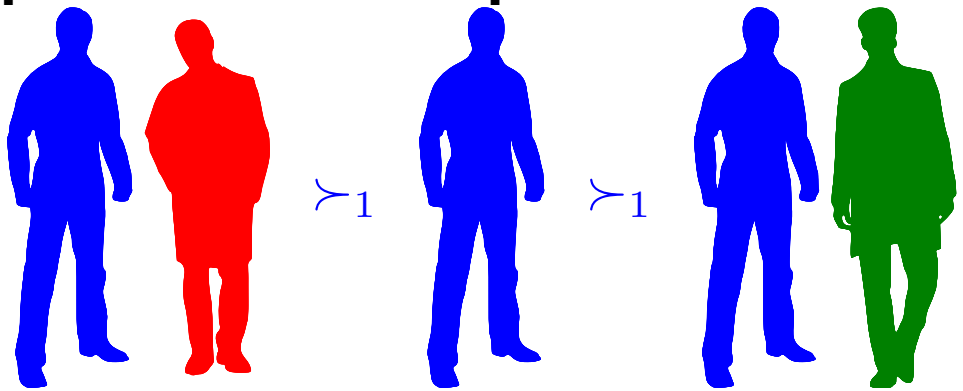


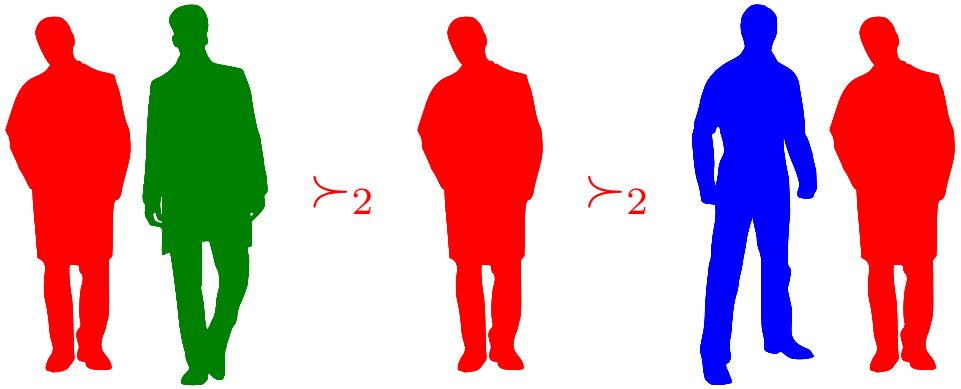


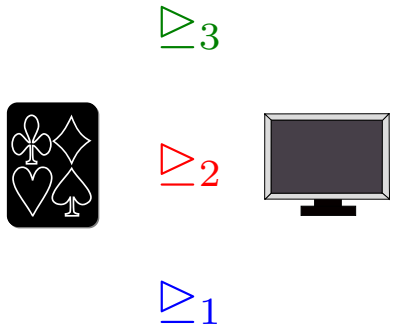


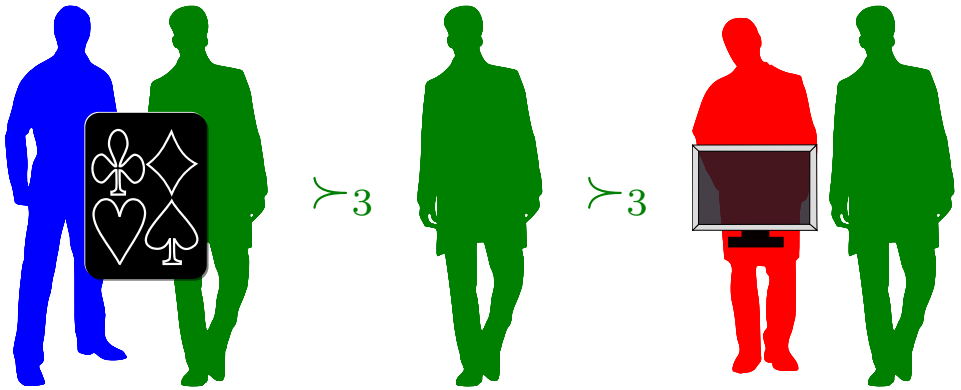
$\mathbf{c}($  $) = 2$

Appariement : exemples







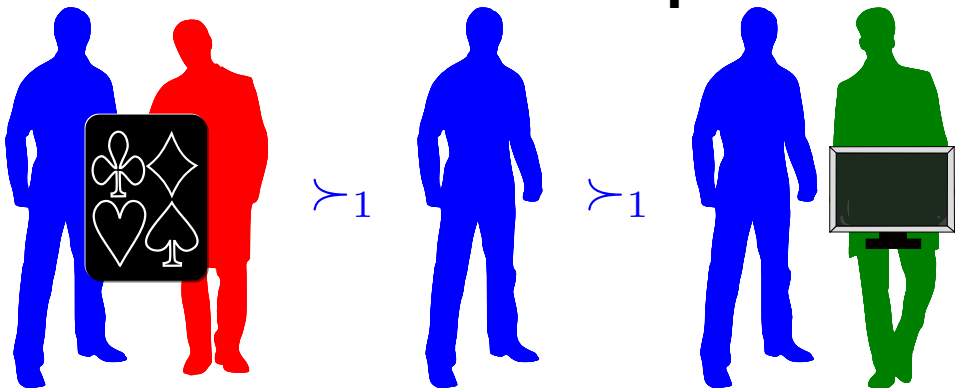


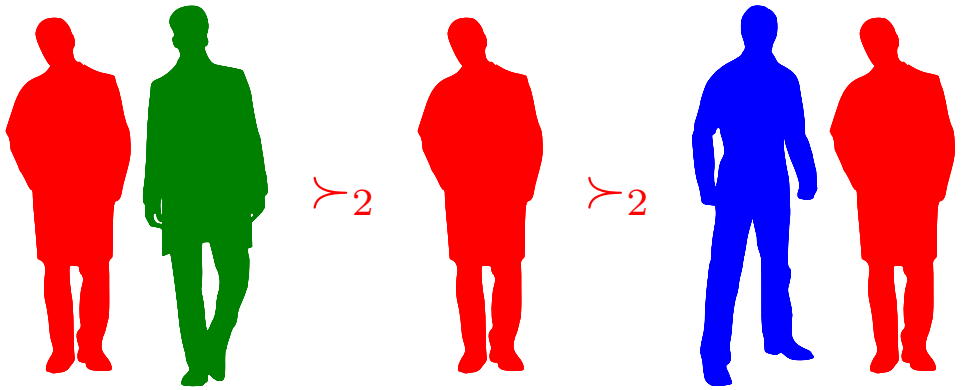
$\mathbf{c}($  $) = 2$

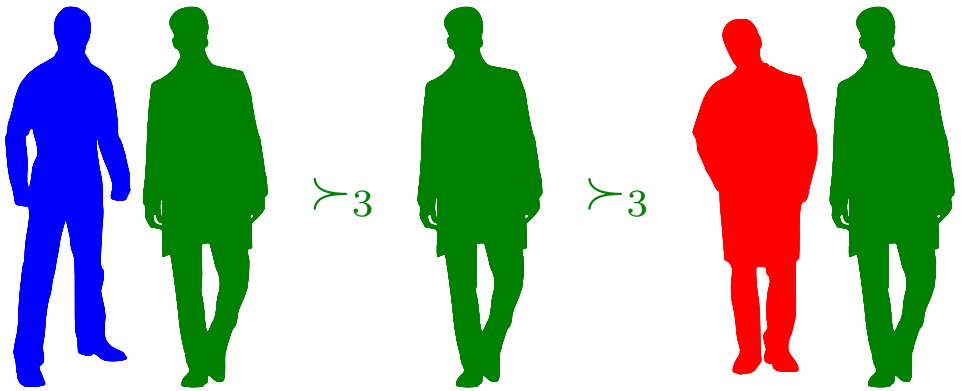
Stabilité d'un appariement *M*

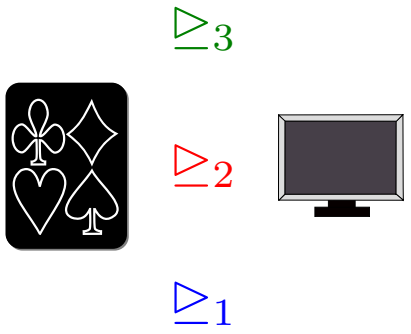
- *M* est **valide** ssi aucune activité n'est surchargée
- *M* est **individuellement rationnel** (IR) ssi chaque individu y apprécie ses partenaires et son activité
- Une coalition *C* **bloque fortement** *M* ssi tous les agents souhaitent et peuvent se séparer et former leur propre coalition.
  M est **stable de cœur** (C) s'il est valide et aucune coalition ne bloque fortement *M*.
- Une coalition *C* **bloque faiblement** *M* ssi au moins un agent souhaite et peut se séparer et former sa propre coalition.
  M est **strictement stable de cœur** (SC) s'il est valide et aucune coalition ne bloque faiblement *M*.
- *M* est **Nash stable** (NS) s'il est valide, rationnel et à l'abri des mouvements individuels.

Stabilité souhaitable mais pas nécessaire



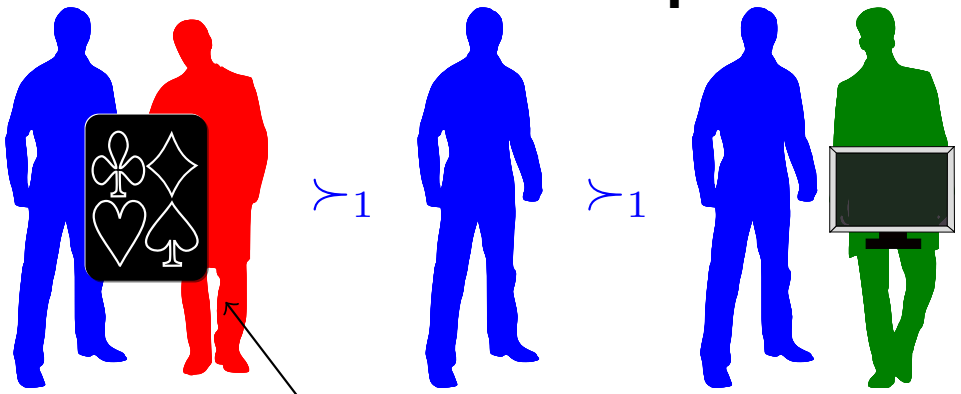


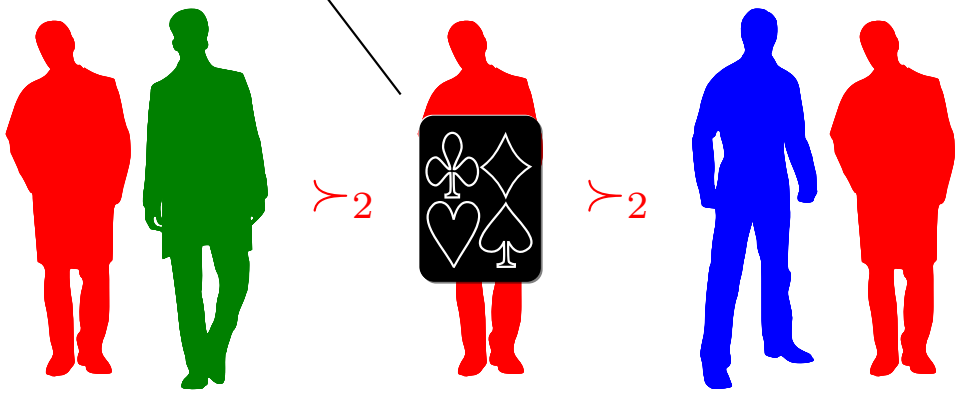


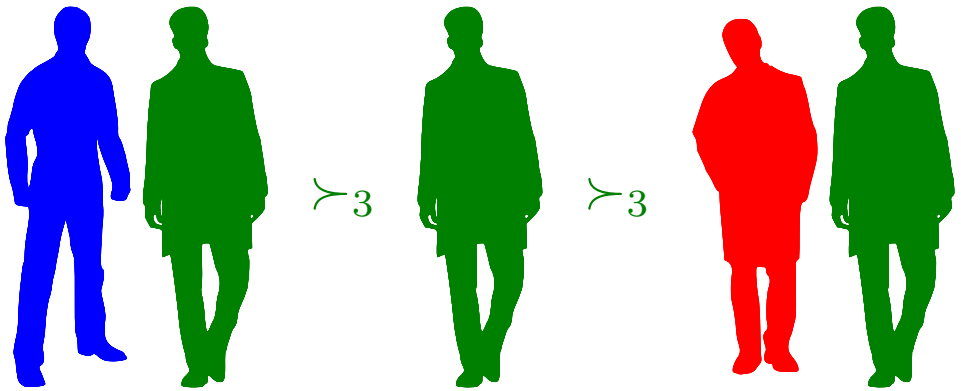


$\mathbf{c}($  $) = 2$

Stabilité souhaitable mais pas nécessaire



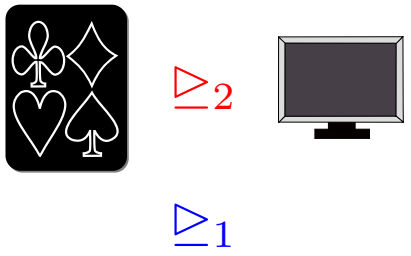




$\succ_1$ $\succ_1$

$\succ_2$ $\succ_2$

$\succ_3$ $\succ_3$



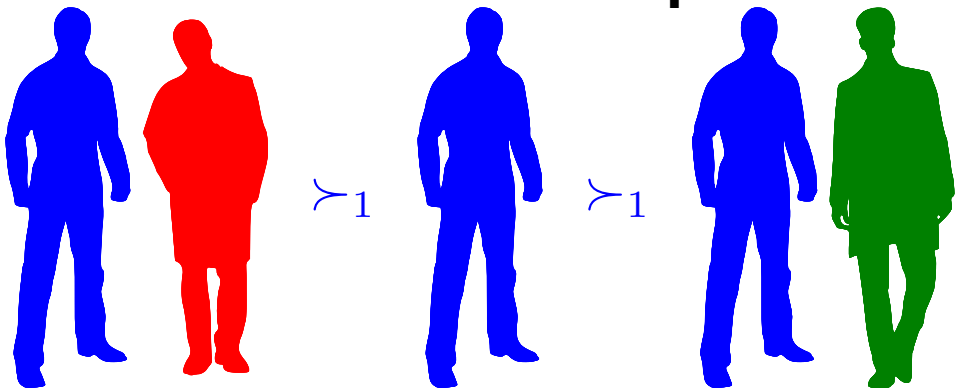
$\trianglerighteq_3$

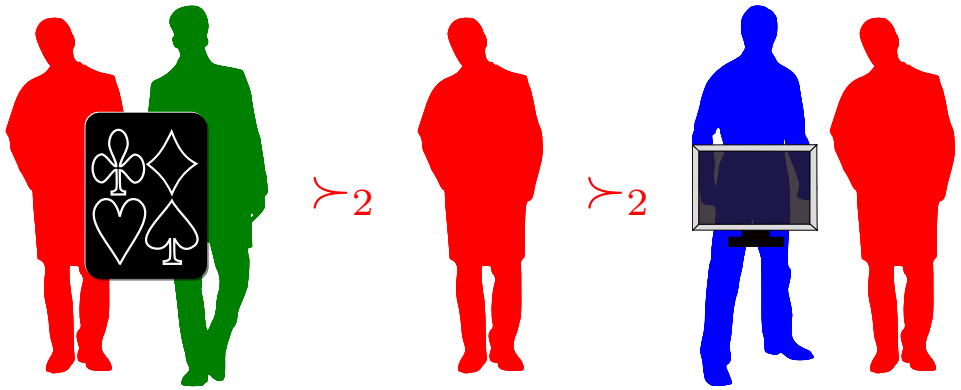
$\trianglerighteq_2$

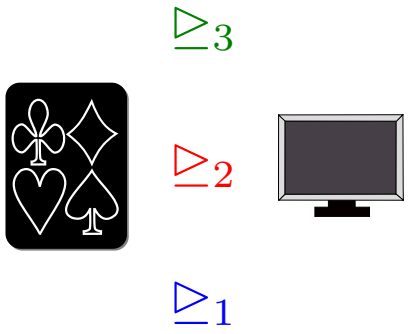
$\trianglerighteq_1$

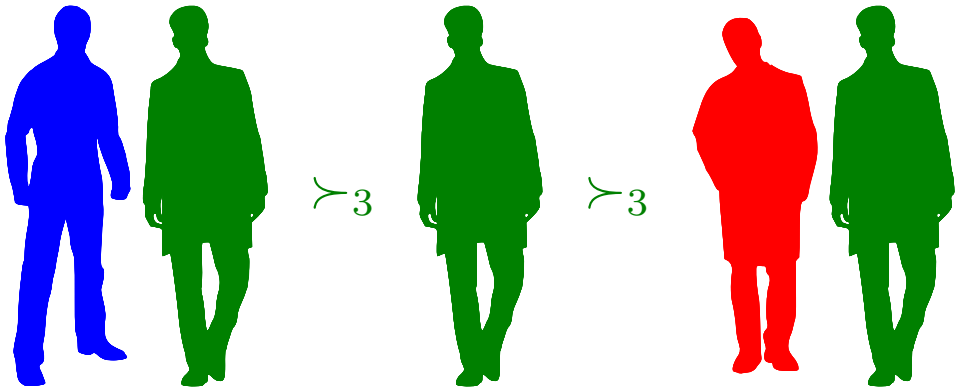
$\mathbf{c}($  $) = 2$

Stabilité souhaitable mais pas nécessaire



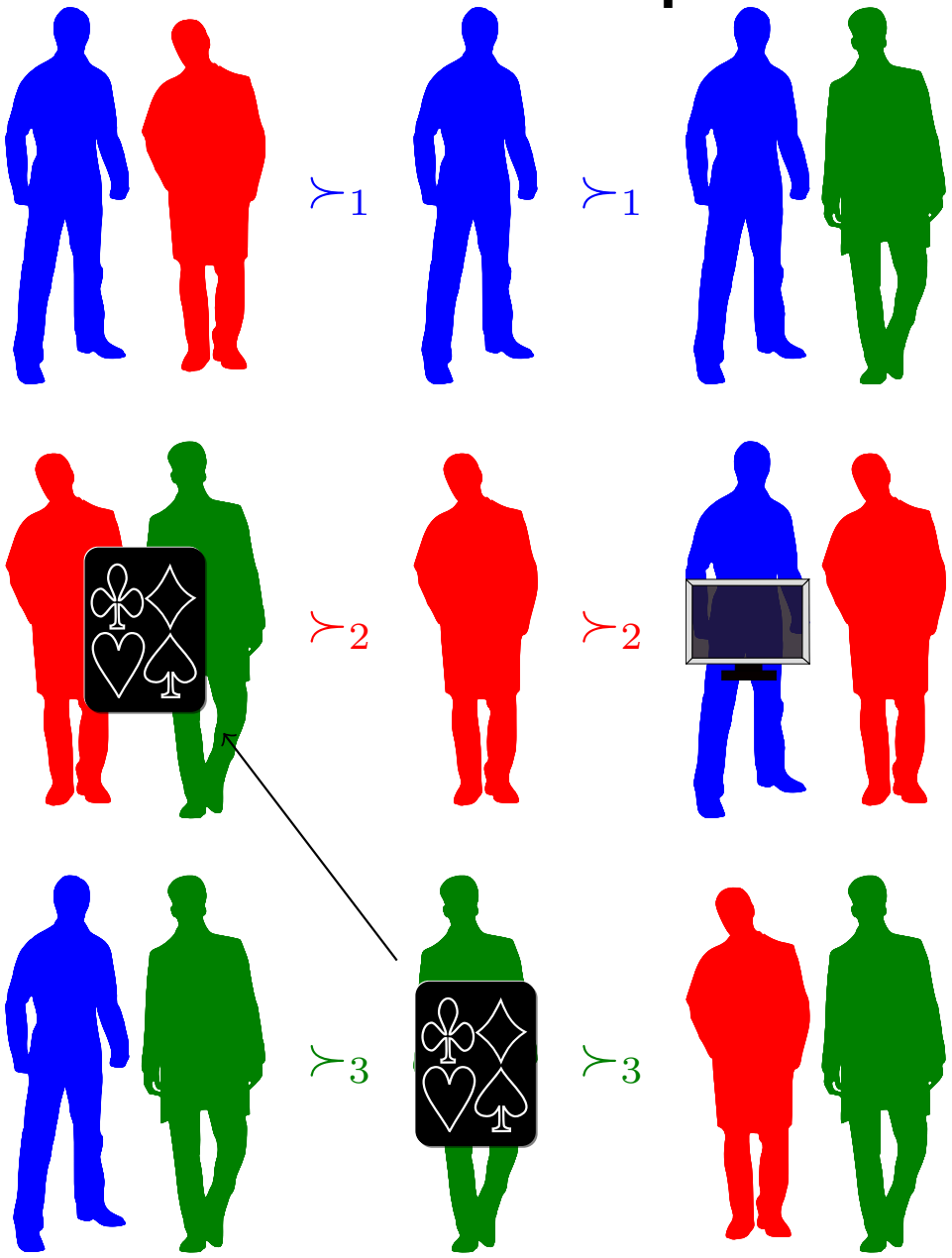




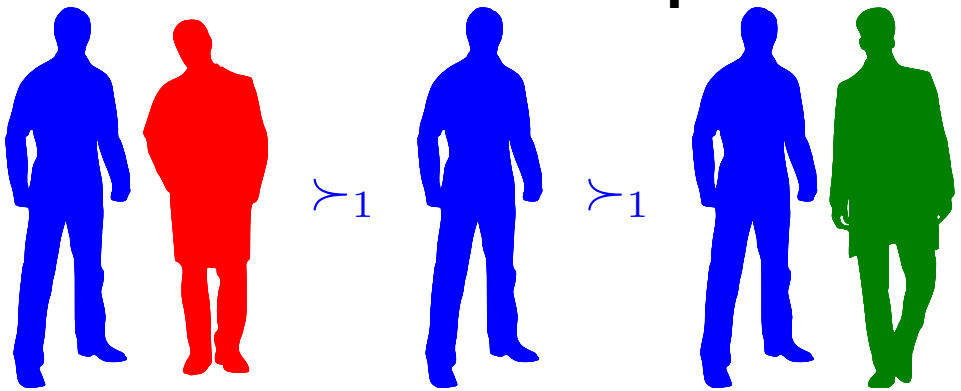


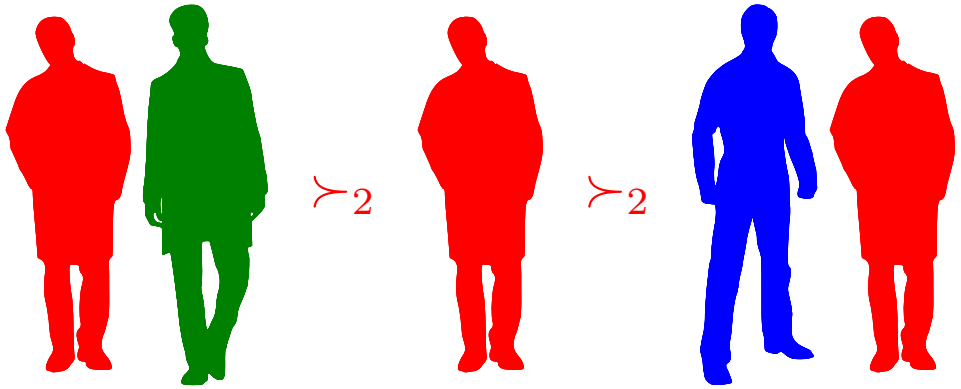
$\mathbf{c}($  $) = 2$

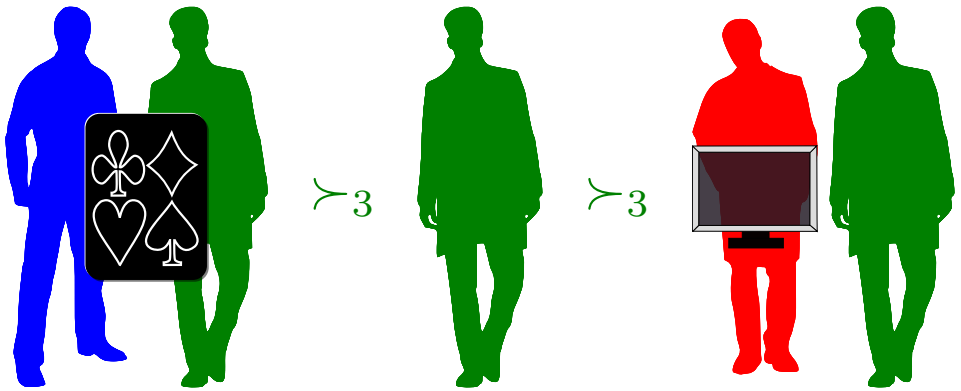
Stabilité souhaitable mais pas nécessaire

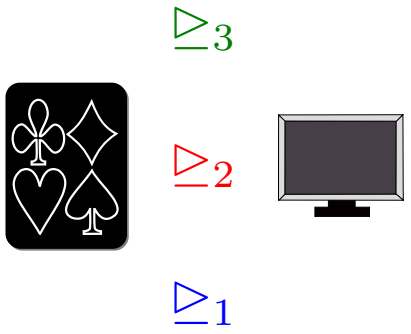


Stabilité souhaitable mais pas nécessaire



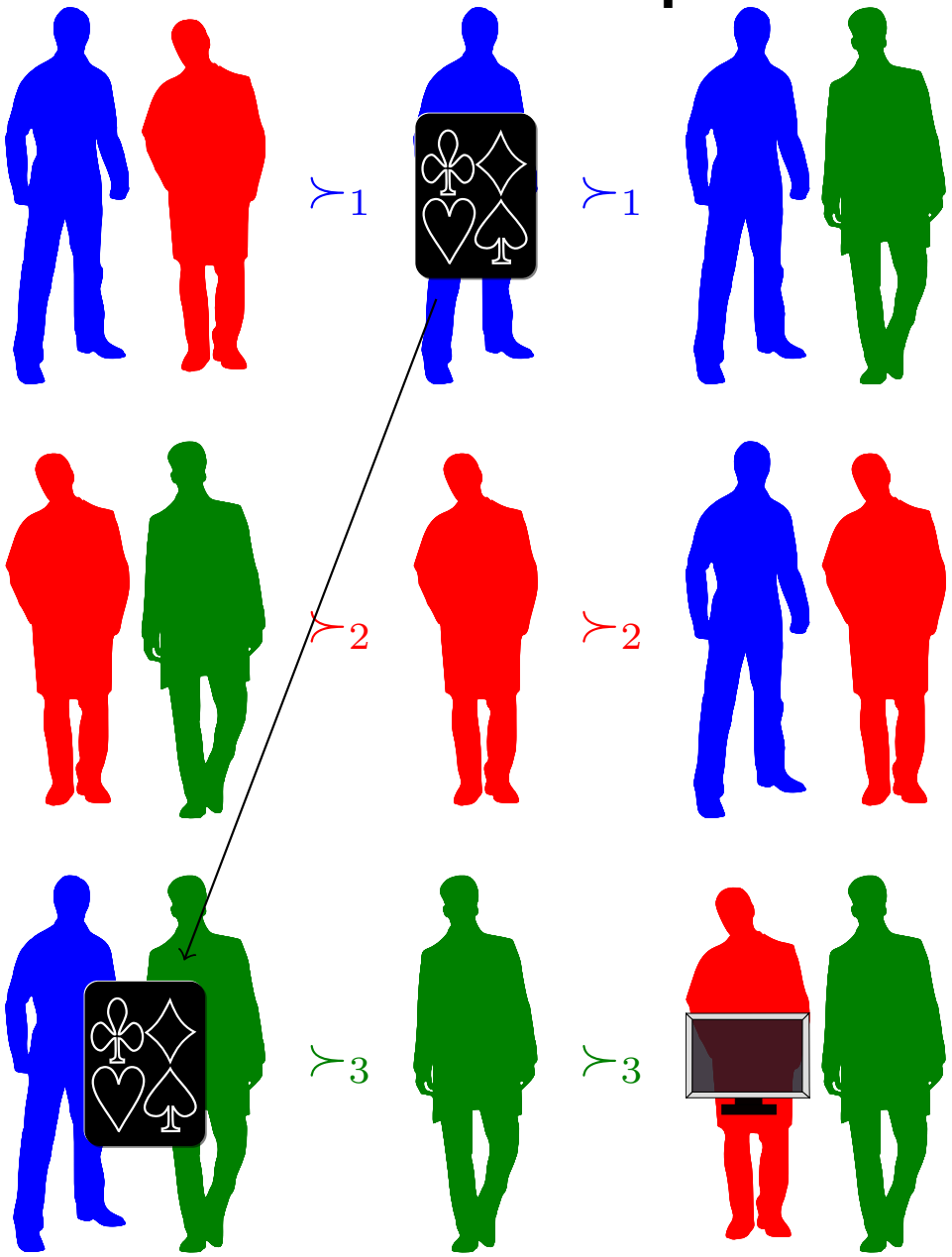


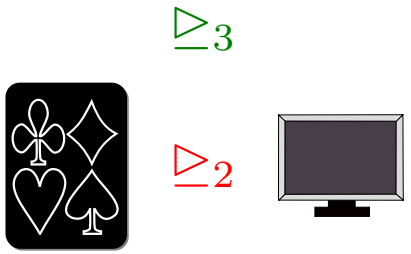




$\mathbf{c}(\heartsuit) = 2$

Stabilité souhaitable mais pas nécessaire



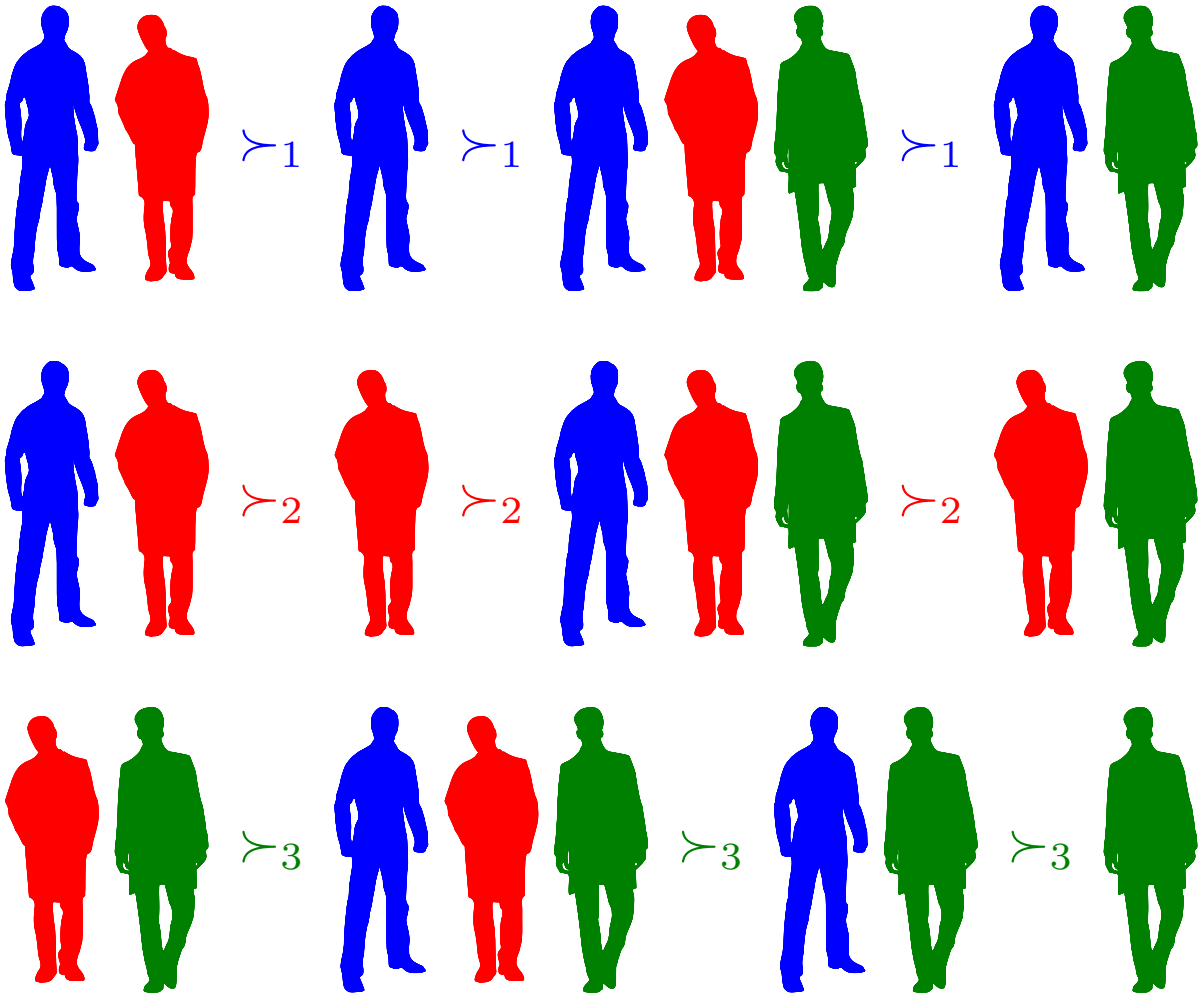


$$\mathbf{c}(\clubsuit) = 2$$

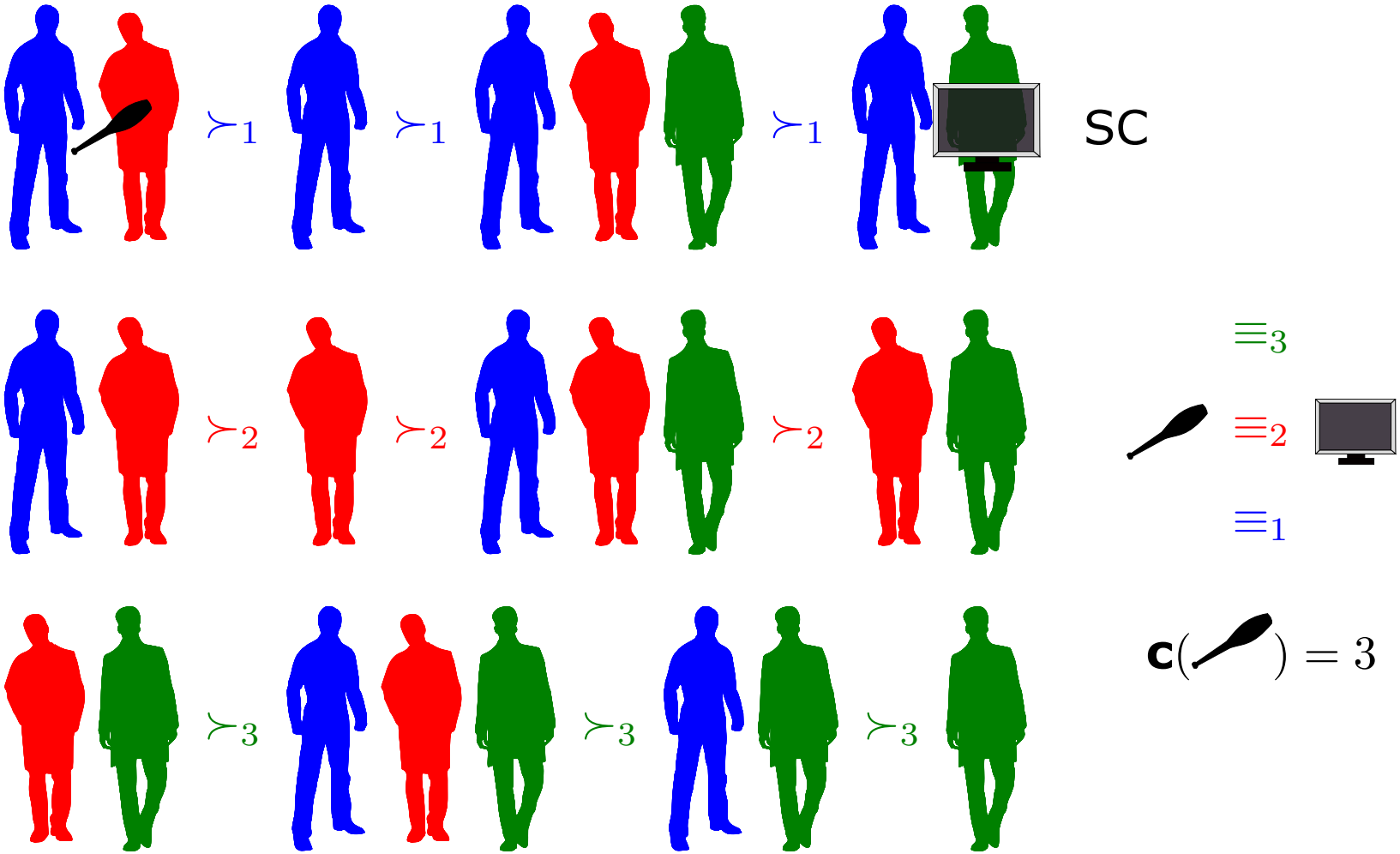
Pareto-optimalité d'un appariement *M*

- *M* est **parfait** si la coalition de n'importe quel individu est une des préférés
- *M′* **Pareto-domine** *M* s'il est strictement meilleur pour au moins un individu et pas pire pour le autres.
- *M* est **Pareto-optimal** s'il n'y a pas d'alternative dans laquelle tous les agents seraient dans une position équivalente ou meilleure.

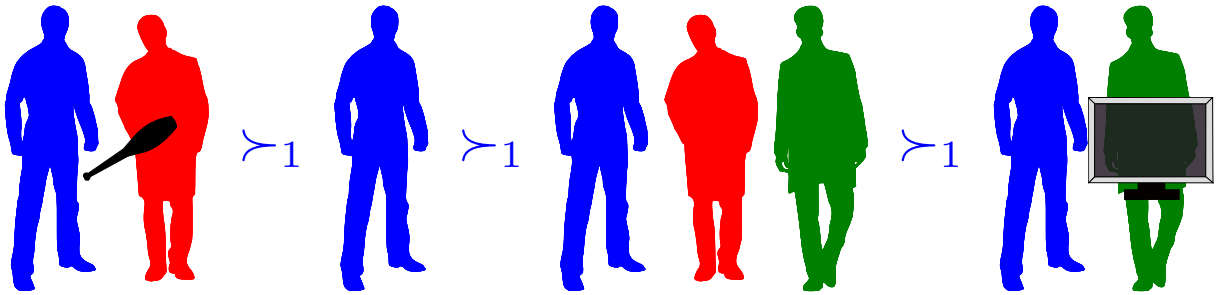
La Pareto-optimalité n'est pas discriminante



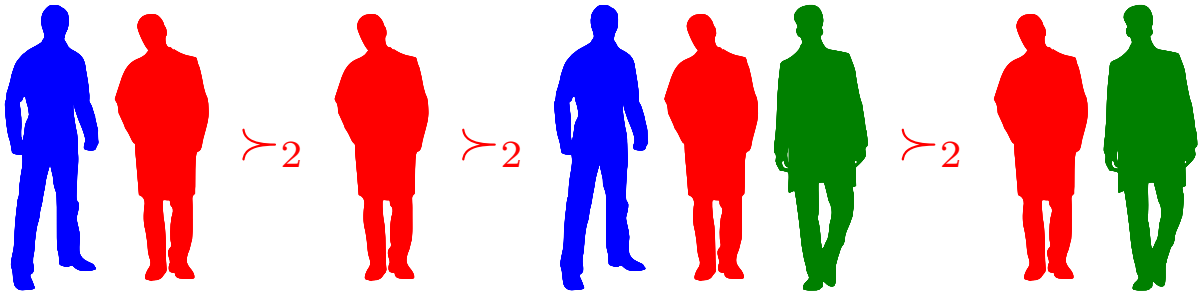
La Pareto-optimalité n'est pas discriminante



La Pareto-optimalité n'est pas discriminante



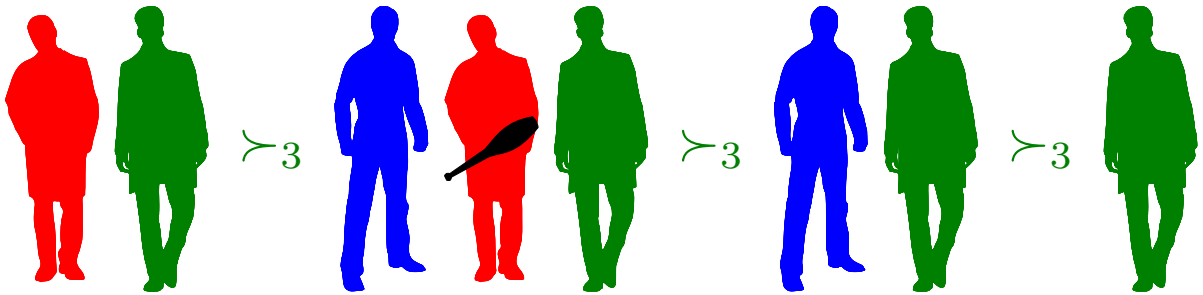
SC



$\equiv_3$

$\equiv_2$

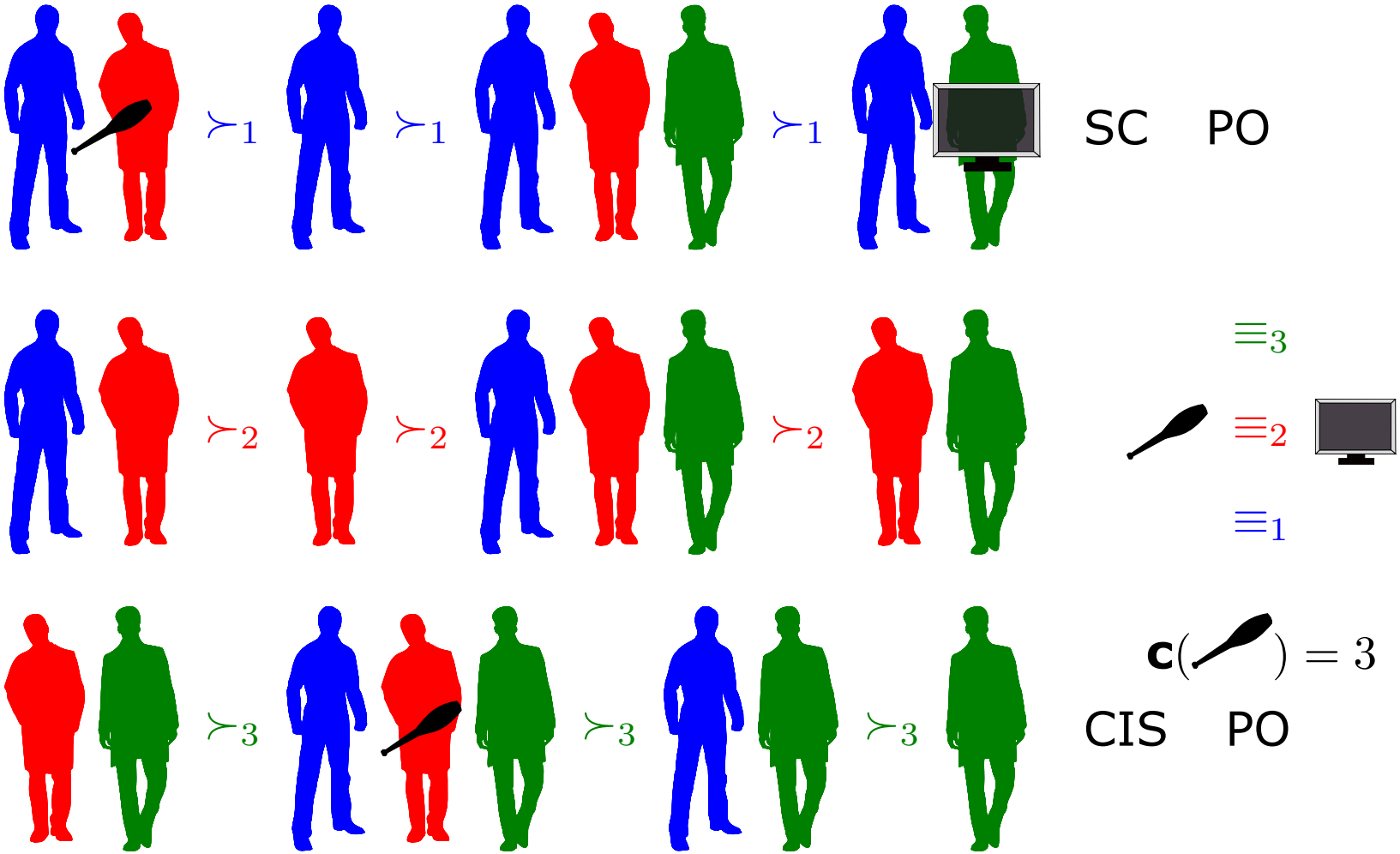
$\equiv_1$



$\mathbf{c}(\quad) = 3$

CIS

La Pareto-optimalité n'est pas discriminante



Problème IA additivement séparable (ASIA)

Chaque individu $i \in I$ est muni :

1. d'une fonction de valuation des activités
   $v_i : A \cup \{\theta\} \to [-1;1]$ $v_i(\theta) = 0$
2. d'une fonction de valuation des partenaires
   $w_i : I \setminus \{i\} \to [-1;1]$

La **fonction d'utilité** $u_i : G(i) \times A \cup \{\theta\} \to [-1;1]$ tq :

$$u_i(g,a) = \frac{[\frac{1}{m-1}\Sigma_{j \in g, j \neq i} w_i(j)] + v_i(a)}{2}$$

Bien-être social d'un appariement M

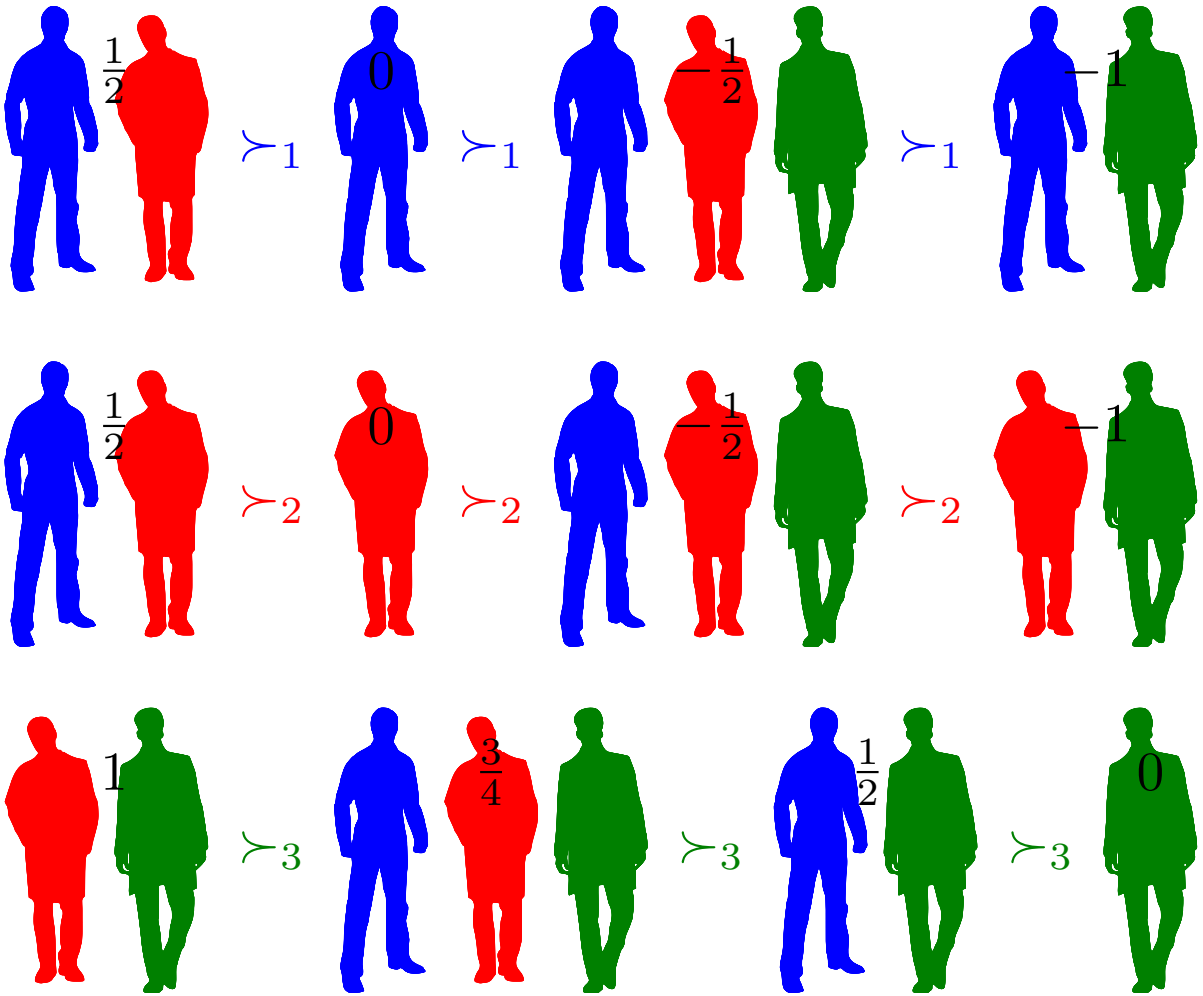
- Le **bien-être utilitaire** de M est :

$$U(M) = \frac{1}{m} \sum_{i \in I} u_i(g_M(i), a_M(i))$$

- Le **bien-être égalitaire** de M est :

$$E(M) = \min_{i \in I}(u_i(g_M(i), a_M(i)))$$

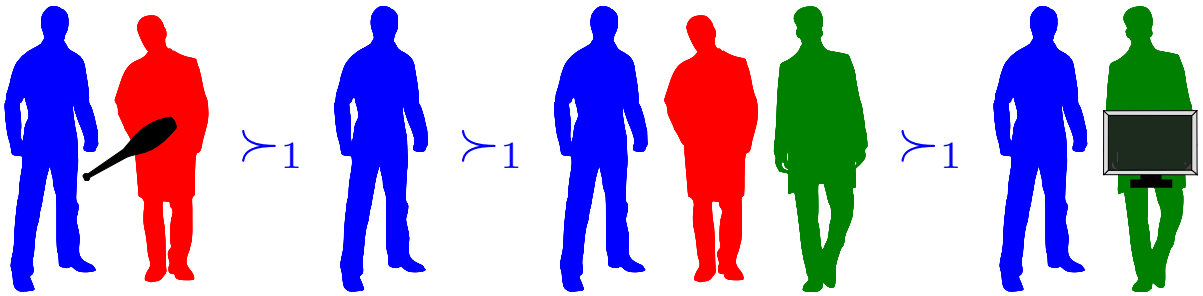
Bien-être utilitaire : exemples



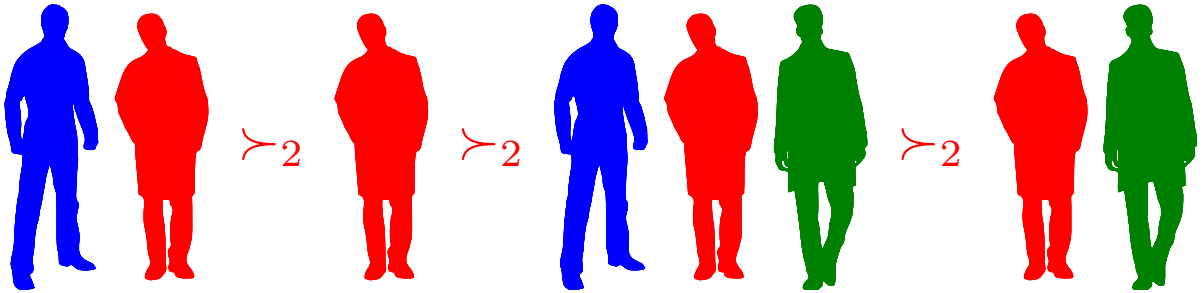
$v_{1\|2\|3}(\text{🪓}) = 0$

$\mathbf{c}(\text{🪓}) = 3$

Bien-être utilitaire : exemples

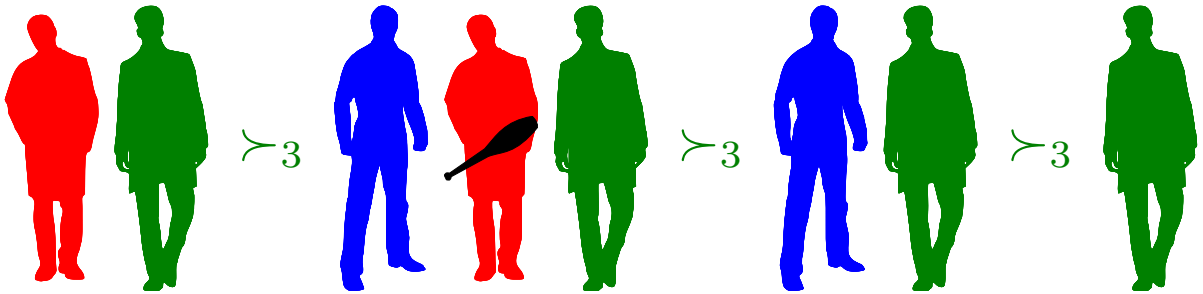


PO $\quad \mathbf{U(M_1) = \frac{1}{12}}$



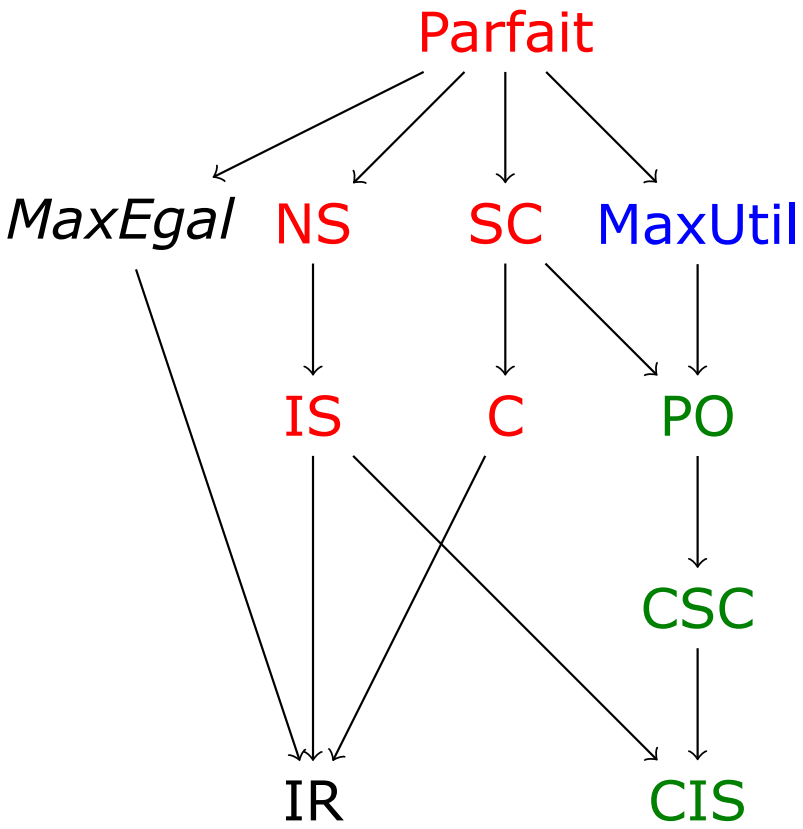
$v_{1\|2\|3}(\text{oar}) = 0$

$\mathbf{c}(\text{oar}) = 3$



PO $\quad \mathbf{U(M_2) = \frac{1}{24}}$

Relations d'inclusion entre stabilité, équité et optimalité



inspiré de [Aziz et al., 2013]

Les individus se proposent à l'activité préférée et concède

```
1   val m = new Matching(asia); var free = asia.individuals
2   while (! free.isEmpty){// Tant que les individus ne sont pas tous affectés
3     free.foreach { i: Individual =>// Pour chaque individu libre
4       if (concessions(i).isEmpty){// S'il est desespéré
5         m.a += (i -> Activity.VOID); m.g += (i -> new Group(i)); free -= i
6       } else {// Sinon il choisit l'activité préféré
7         val a = concessions(i).head; val g = m.p(a); val ng = g + i
8         if (g.isEmpty){// Si l'activité est inoccupée
9           m.a += (i -> a); m.g += (i -> new Group(i)); free -= i
10        } else {// Sinon tous les sous-groupes sont évalués
11          var subgroups = ng.subgroups().filterNot(o => o.isEmpty)
12          if (a.q = g.size) subgroups -= ng
13          var umax = Double.MinValue; var bestG = new Group()
14          subgroups.foreach{ g2 =>
15            val u= g2.u(a)
16            if (umax < u) {
17              umax = u; bestG = g2
18            }
19          }// Le meilleur sous-groupe est sélectionné
20          bestG.foreach { j => m.g += (j -> bestG) }
21          (g.diff(bestG)).foreach {j : Individual =>// Des individus peuvent être désaffectés
22            m.a += (j -> Activity.VOID); m.g += (j -> new Group(j)); free += j
23            concessions += (j -> concessions(j).tail)//et concéder
24          }
25          if (bestG.contains(i)){//L'individu peut être affecté
26            m.a += (i -> a); free -= i
27          } else concessions += (i -> concessions(i).tail)//ou concéder
28        }
29      }
30    }
31  }
```

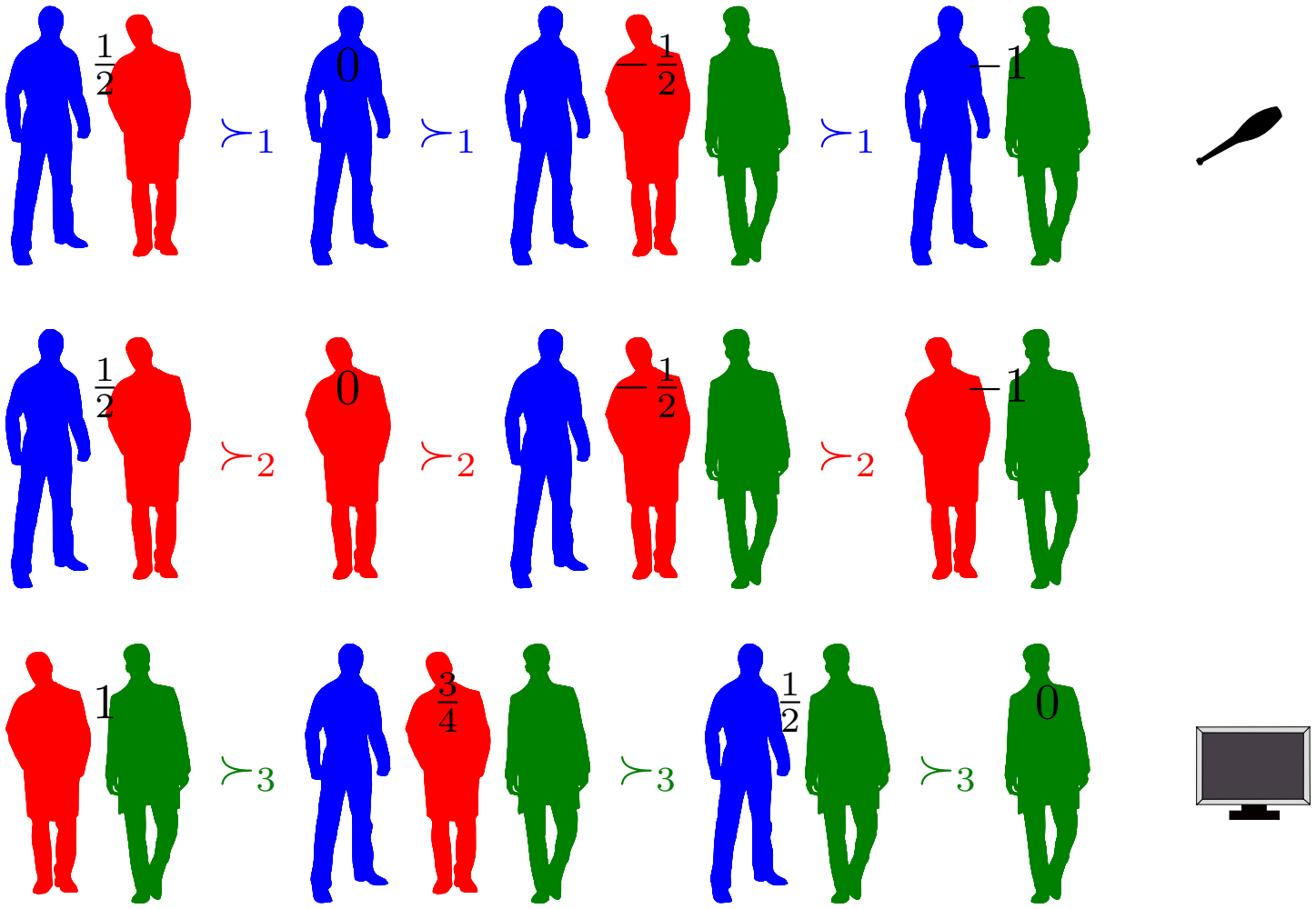
Algorithme d'approximation

```
1   val m = new Matching(asia); var free = asia.individuals
2   while (! free.isEmpty){// Tant que les individus ne sont pas tous affectés
3     free.foreach { i: Individual =>// Pour chaque individu libre
4       if (concessions(i).isEmpty){// S'il est desespéré
5         m.a += (i -> Activity.VOID); m.g += (i -> new Group(i)); free -= i
6       } else {// Sinon il choisit l'activité préféré
7         val a = concessions(i).head; val g = m.p(a); val ng = g + i
8         if (g.isEmpty){// Si l'activité est inoccupée
9           m.a += (i -> a); m.g += (i -> new Group(i)); free -= i
10        } else {// Sinon tous les sous-groupes sont évalués
11          var subgroups = Set[Group]()
12          if ( approximation ) {// Un seul individu est exclu
13            ng.foreach { j => subgroups += ng.filterNot(_.equals(j)) }
14            subgroups+=ng
15          } else subgroups = ng.subgroups().filterNot(o => o.equals(Group()))
16          if (a.q = g.size) subgroups -= ng
17          var umax = Double.MinValue; var bestG = new Group()
18          subgroups.foreach{ g2 =>
19            val u= g2.u(a)
20            if (umax < u) {
21              umax = u; bestG = g2
22            }
23          }// Le meilleur sous-groupe est sélectionné
24          bestG.foreach { j => m.g += (j -> bestG) }
25          (g.diff(bestG)).foreach {j : Individual =>// Des individus peuvent être désaffectés
26            m.a += (j -> Activity.VOID); m.g += (j -> new Group(j)); free += j
27            concessions += (j -> concessions(j).tail)//et concéder
28          }
29          if (bestG.contains(i)){//L'individu peut être affecté
30            m.a += (i -> a); free -= i
31          } else concessions += (i -> concessions(i).tail)//ou concéder
32        }
33      }
34    }
35  }
```

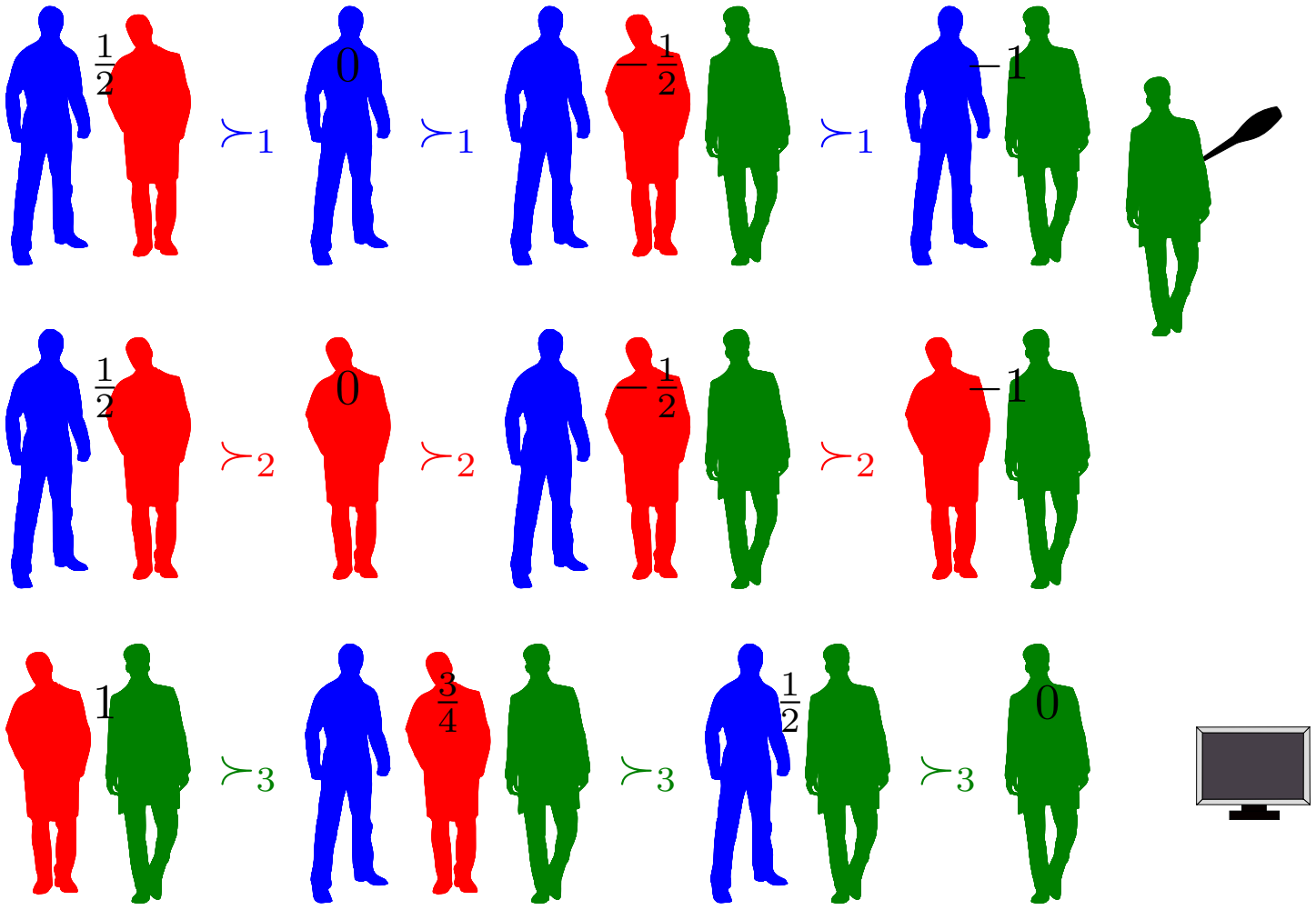
Résultats théoriques

- L'algorithme (d'approximation) se termine, i.e. l'invariant de boucle $\Sigma_{i \in I} |\mathit{concessions}(i)| + |\mathit{Free}|$
- L'algorithme (d'approximation) retourne un appariement valide, i.e. aucune activité n'est surchargée
- L'appariement retourné par l'algorithme exact est Pareto-optimal, i.e. un individu concède si sa présence importune le groupe
- Le bien-être utilitaire n'est pas nécessairement maximal l'algorithme est orienté activité

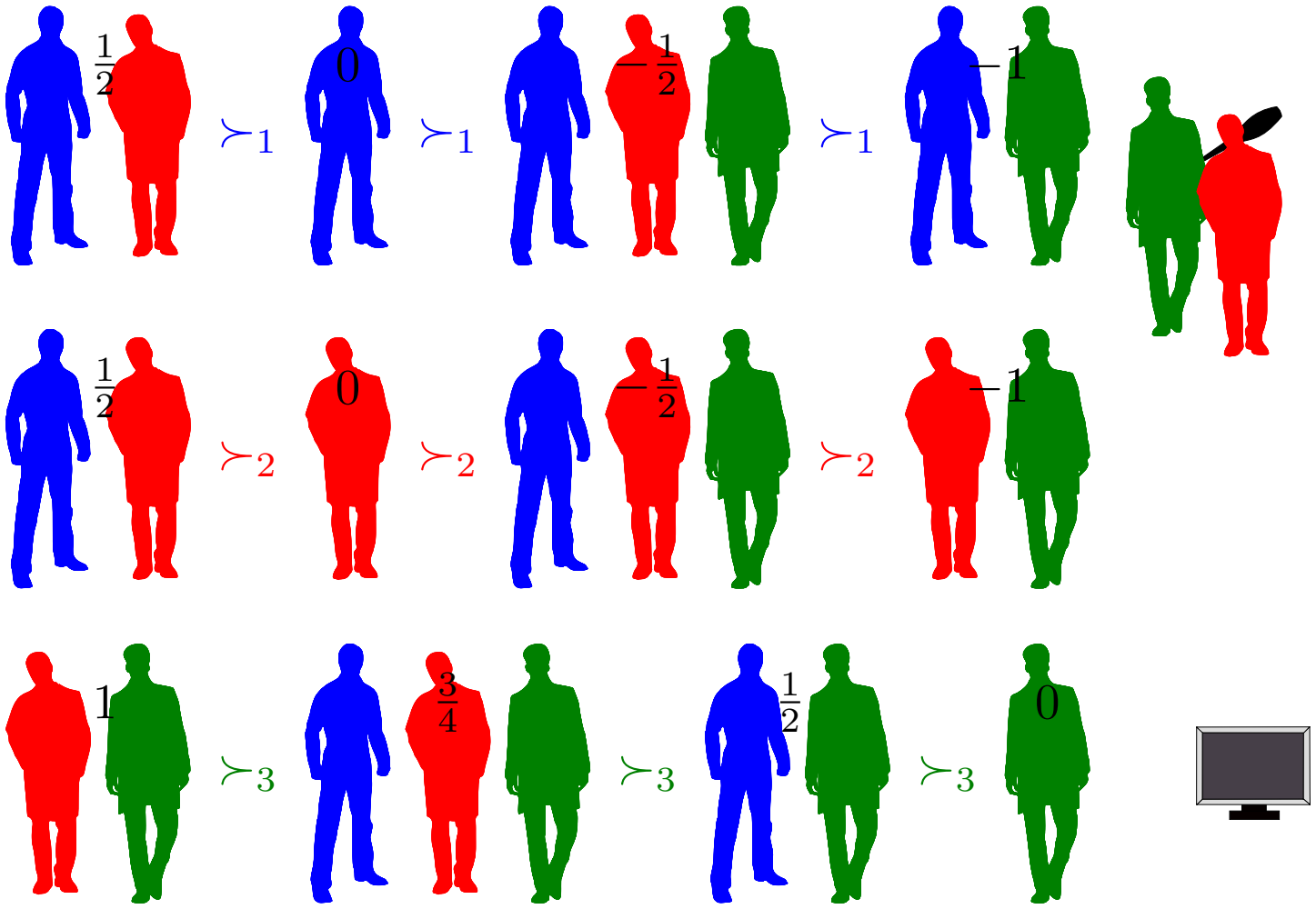
Algorithme : exemple



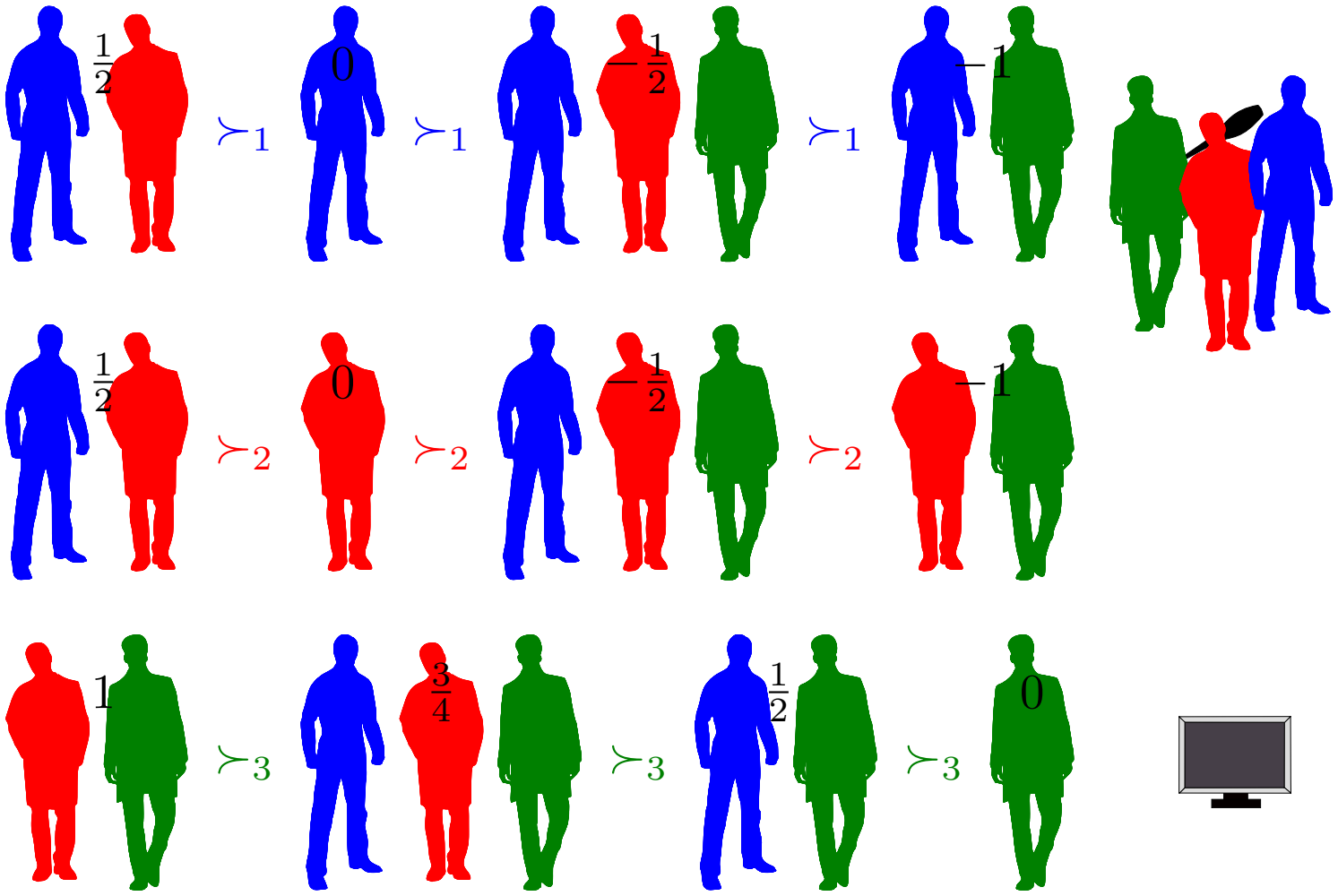
Algorithme : exemple



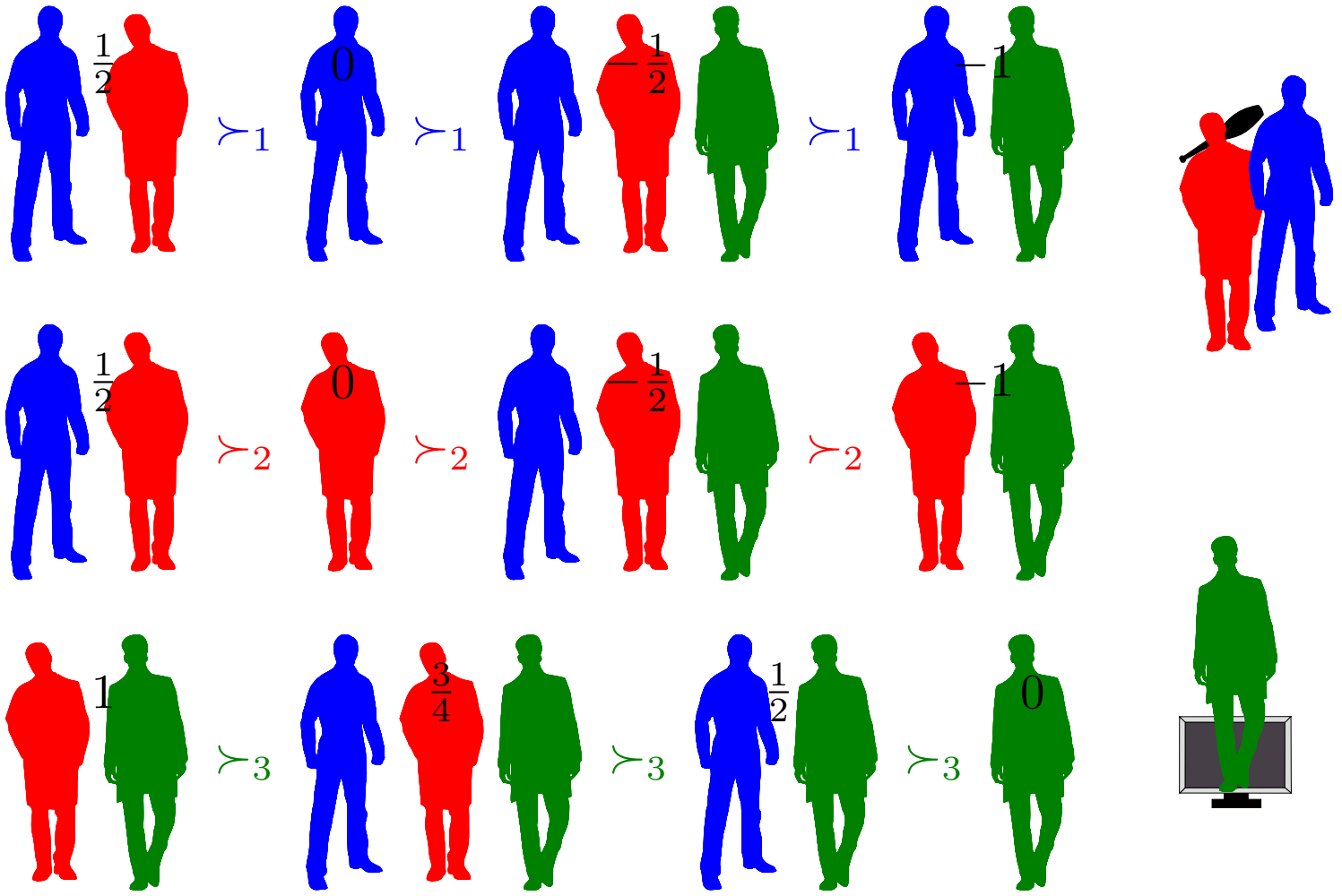
Algorithme : exemple



Algorithme : exemple



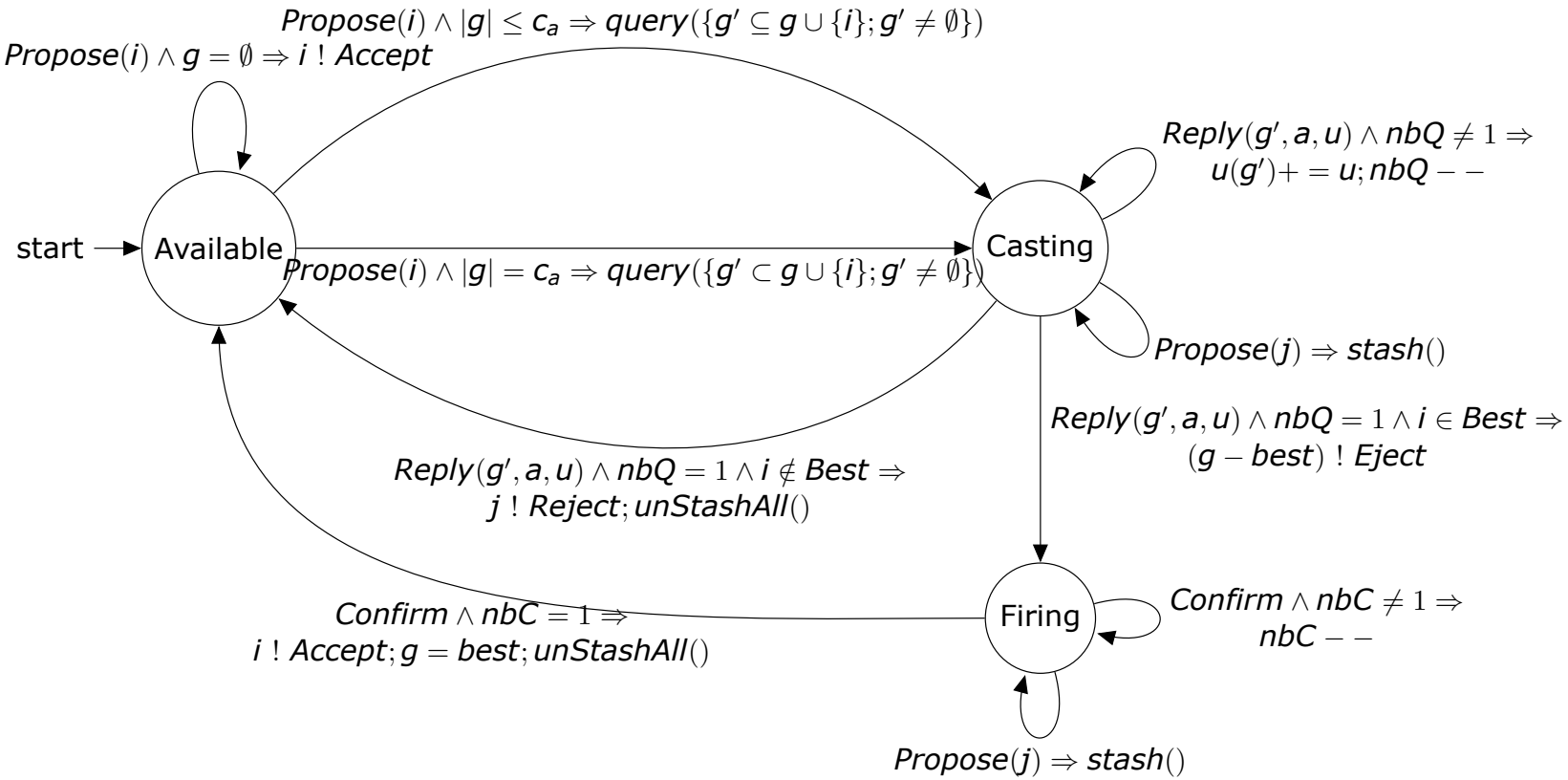
Algorithme : exemple



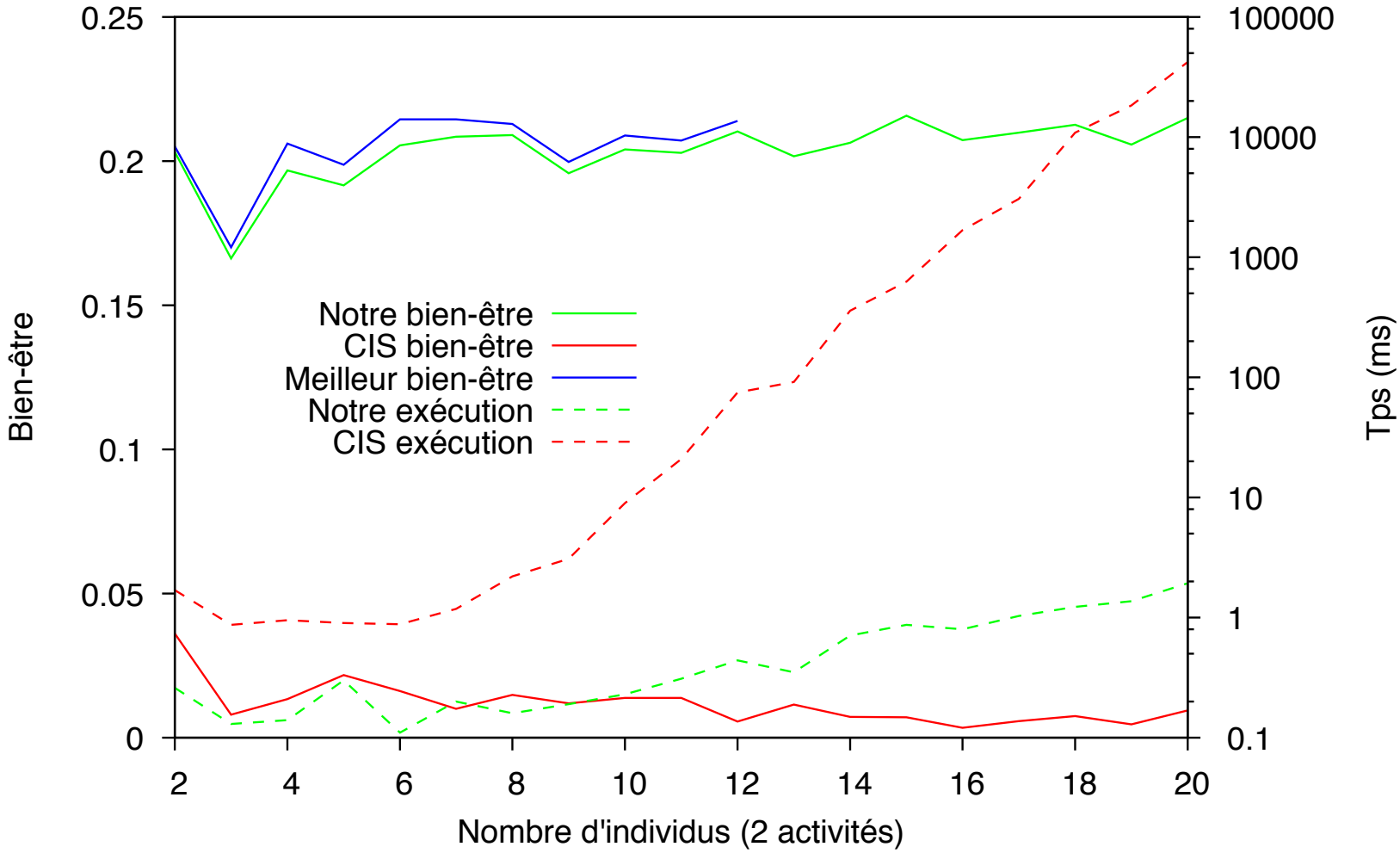
Comportement d’un agent « individu »

```
1    def receive = {
2      case Start => {
3        supervisor = sender
4        if (! concessions.isEmpty) adr(concessions.head) ! Propose(name) // Se propose à l'activité préférée
5        else supervisor ! Assignement(name, Activity.VOID) // ou reste définitivement seul
6      }
7      case Accept => { // L'affectation est confirmée
8        supervisor ! Assignement(name, concessions.head)
9      }
10     case Reject => {// L'affectation est infirmée
11       concessions = concessions.tail
12       if (concessions.isEmpty) supervisor ! Assignement(name, Activity.VOID) // Soit il est désépéré
13       else adr(concessions.head) ! Propose(name) // Soit il concède
14     }
15     case Withdraw => {// Désaffectation
16       supervisor ! Disassignement(name)
17     }
18     case Confirm => {//  Désaffectation confirmée par le solveur
19       adr(concessions.head) ! Confirm
20       concessions = concessions.tail
21       if (concessions.isEmpty) supervisor ! Assignement(name, Activity.VOID) // Soit il est desépéré
22       else adr(concessions.head) ! Propose(name) // Soit il concède
23     }
24     case Query(group,activity) => {// L'utilité est demandé
25       sender ! Reply(group,activity,i.u(group,activity))
26     }
27   }
```

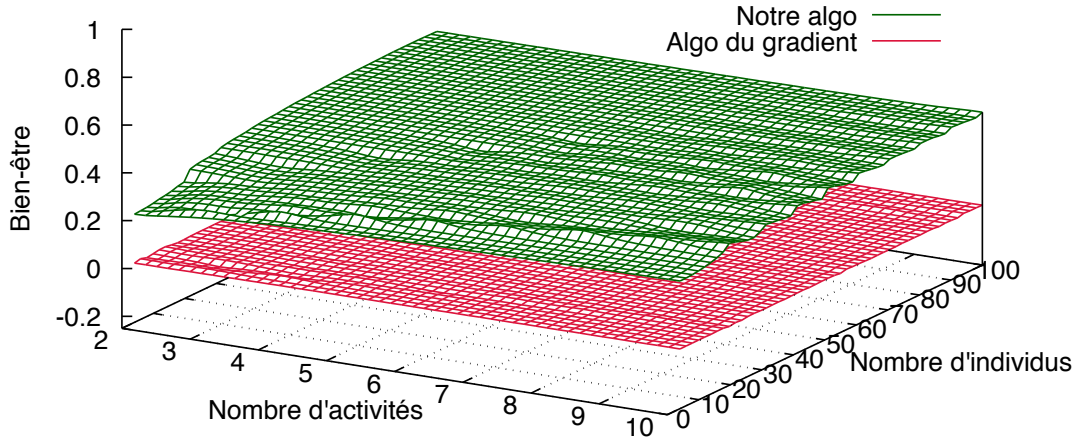
Comportement d'un agent « activité »

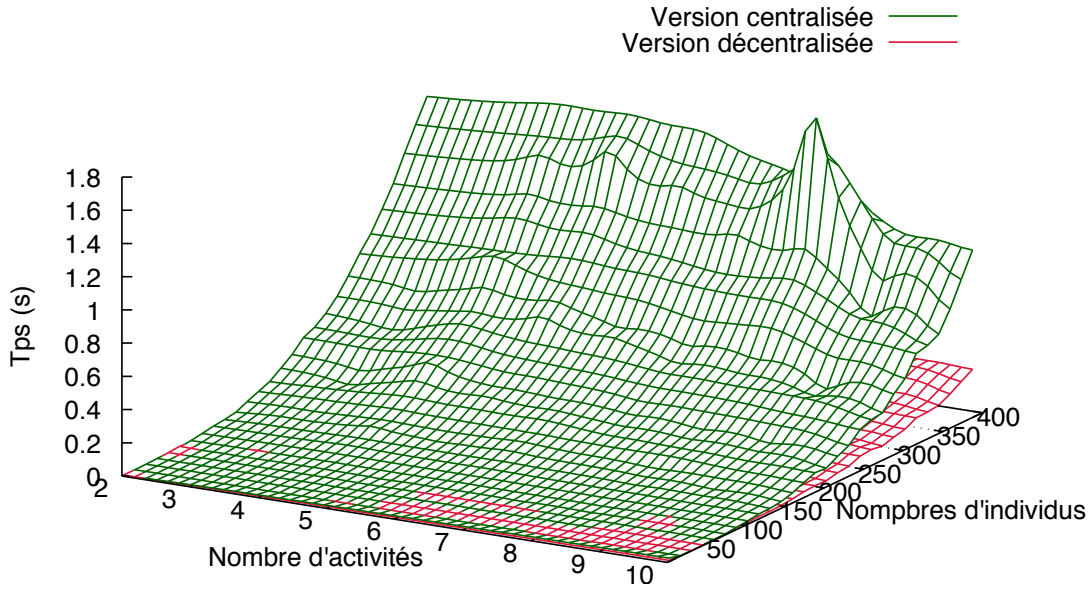


Expérimentations : notre algorithme vs. [Ballester, 2004]



Expérimentations : algo. du gradient vs. notre algo.

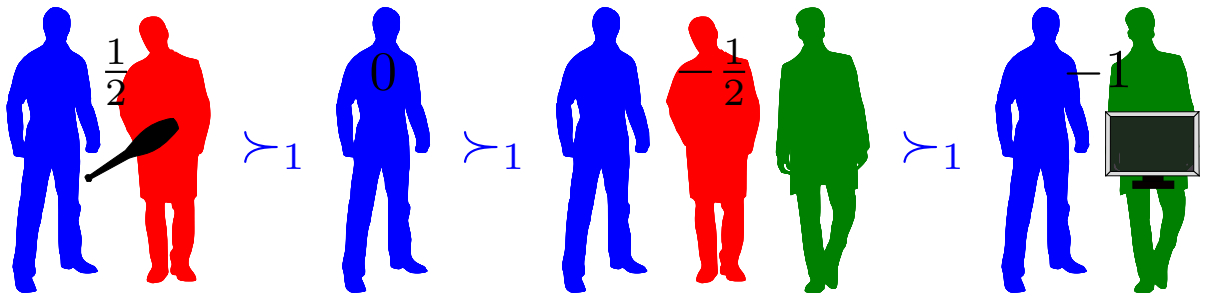




Affectation distribuée d'individus à des activités

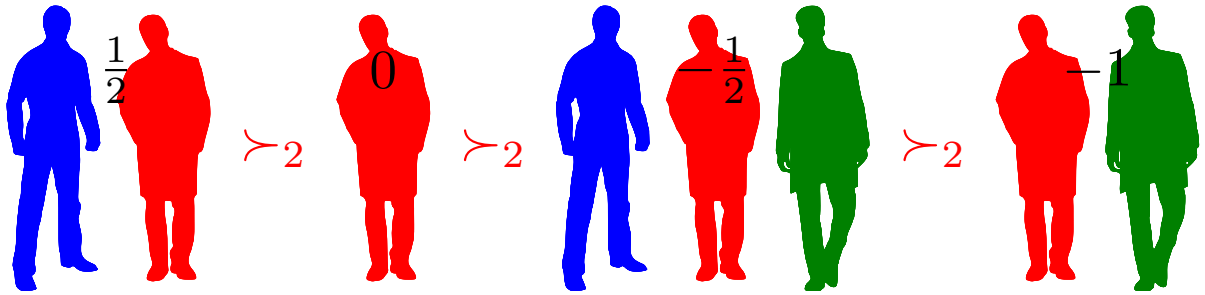
- **Motivation :** un réseau social numérique pour la cohésion sociale et lutter contre l'isolement
- **Contribution :**
  1. Formalisation d'une problème Individus/Activités avec des préférences additivement séparable
  2. Algorithme distribué pour des solutions Pareto-optimales
- **Difficulté :** synchronisation et détection d'arrêt
- **Perspectives :**
  1. bien-être égalitaire
  2. données réelles

Bien-être égalitaire : exemples

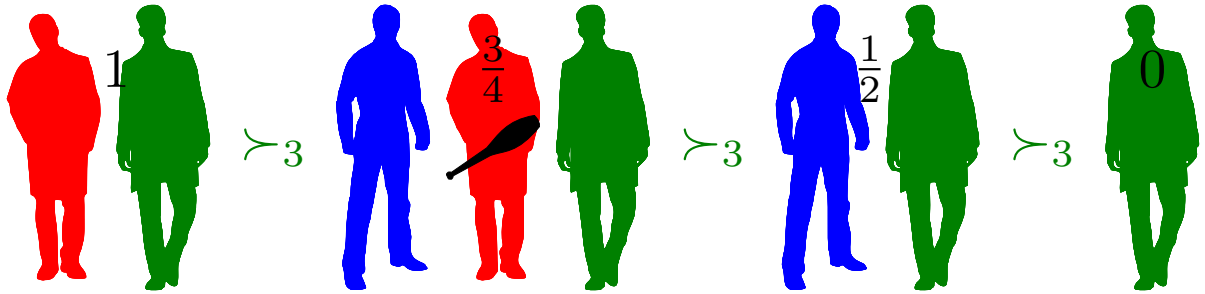


$\mathbf{U(M_1) = \frac{1}{12}}$

$\mathbf{E(M_1) = 0}$



$v_{1\|2\|3}(\text{oar}) = 0$



$\mathbf{c}(\text{oar}) = 3$

$\mathbf{U(M_2) = \frac{1}{24}}$

$\mathbf{E(M_2) = -\frac{1}{8}}$

References I


Aziz, H., Brandt, F., and Seedig, H. G. (2013).
Computing desirable partitions in additively separable hedonic games.
AIJ, 195 :316–334.


Ballester, C. (2004).
NP-completeness in hedonic games.
GEB, 49(1) :1–30.


Dreze, J. and Greenberg, J. (1980).
Hedonic coalitions : Optimality and stability.
Econometrica, 48 :987–1003.


Igarashi, A., Peters, D., and Elkind, E. (2017).
Group activity selection on social networks.
In *Proc. of AAAI*, pages 565–571.


Manlove, D. F. (2014).
Algorithmics of Matching Under Preferences.
World Scientific.

References II


Nongaillard, A. and Picault, S. (2016).
Modélisation multi-niveaux des problèmes d'affectation et d'appariement.
In *24e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'16)*, pages 75–84, Rouen, France. Cépaduès.