Fouille de processus pour l'amélioration d'un jeu sérieux

Sébastien Amoury, Karell Bertet

Fouille de processus pour l'amélioration d'un jeu sérieux

Sébastien Amoury and Karell Bertet

Laboratoire Informatique, Image et Interaction, La Rochelle Université, La Rochelle, France

Résumé : Nous évaluons, dans cet article, l'efficacité de différents algorithmes de fouille de processus pour les traces numériques recueillies lors d'interactions Homme-Robot dans le contexte d'un jeu sérieux, ainsi que l'apport des modèles générés par ces algorithmes sur l'amélioration des activités. L'extraction de modèles à partir des logs de chaque joueur permet d'entrapercevoir des comportements communs entre différents joueurs, mais aussi des comportements moins communs, voir uniques, des "pépites" propres à chaque joueur. Cela permet aussi de mettre en lumière les différents problèmes que les joueurs ont pu rencontrer (problèmes de reconnaissance vocale avec un robot, mauvaise réponse ou réponse non-prévu) et des pistes pour améliorer l'interaction. Néanmoins l'extraction de modèles via la fouille de processus sur un jeu de données aussi large sont difficile à exploiter. Nous proposons ici quelques perspectives pour les rendre plus exploitables.

Mots clés : Fouille de processus, Fouille de séquences, Interactions Humain-Robot

1 Introduction

A l'heure du "tout numérique", la génération de données et l'exploitation de ces dernières est un enjeu de taille pour la recherche ou l'industrie. L'interaction Humain-Robot est un domaine de recherche où l'interactivité y est présente sous nombre de formes, ce qui suscite fortement l'intérêt chez le public. Chez les personnes jeunes, particulièrement friandes de ce genre de choses qui y voient une manière d'accéder à du contenu numérisé et souvent de manière ludique, de l'engouement est généré. Chez les personnes plus âgées, cela représente une curiosité à découvrir. Cet intérêt, de plus en plus croissant pour des expériences reposant sur ce type d'interaction amène des secteurs à s'adapter et proposer bien plus ce genre d'expérience afin d'attirer le public au sein de ces secteurs, comme par exemple les lieux de culture tels que les musées, peu fréquentés par une population jeune. Ce type d'expérience interactive mêlant ludique et culturel s'inscrit dans le domaine des jeux sérieux (serious game), qui consiste à mêler le côté culturel (d'un endroit, d'une exposition etc.) au côté ludique d'une interaction avec différents terminaux numériques tels que des tablettes, des smartphones ou bien des robots, et ce afin de proposer des expériences plus enrichissantes et appréciables qu'une visite simple.

Ces différentes interactions vont générer des traces numériques, aussi appelés logs, principalement sous un format textuel, comme présenté sous la forme du tableau 1. Après leur collecte sous forme de journal d'activités, les logs peuvent être exploités grâce à diverses techniques issues de la fouille de processus.

- **CaseID :** il correspond à l'identifiant de l'utilisateur qui effectue l'action ;
- **Activité :** l'événement que l'utilisateur a déclenché ;
- **Timestamp :** la date et l'heure où l'utilisateur a déclenché son événement ;

caseID	Activité	Timestamp
joueur 1	repondsQuestion1	13/10/2017 14:25:01
joueur 2	demandeIndice3	13/10/2017 14:26:54
joueur 1	repeteQuestion2	13/10/2017 14:28:30

Tableau 1. Exemple de données exploitables par un algorithme de fouille de processus

La fouille de processus permet d'extraire des graphes ou des réseaux de Petri afin de visualiser l'enchaînement des différents processus exécutés lors d'une interaction. Un analyste peut ainsi tirer des conclusions de ce qu'il perçoit, afin de faire évoluer cette interaction.

L'idée derrière la fouille de processus est de découvrir les processus (Process Discovery), vérifier la conformité ou la qualité des modèles découverts (Conformance Checking) et l'amélioration de processus réels (Enhancement) par l'extraction de connaissances à partir des données de parcours.

Malgré sa jeunesse dans le paysage informatique (le début remonte aux années 1990), la fouille de processus a obtenu rapidement un intérêt croissant dans divers domaines d'application du fait de ses finalités. On peut observer son utilisation dans le cadre d'analyse de processus dans un hôpital [5], dans l'industrie [10] ou bien encore dans le jeu-vidéo [2]. Des travaux ont également été effectués dans le cadre des jeux sérieux comme on peut le voir dans les travaux de Hernández et al. [3].

Le fonctionnement général d'un algorithme de fouille de processus est divisé en trois étapes : (1) l'analyse des données de navigation dans les fichiers de logs ; (2) l'identification des processus et de leurs relations; (3) construction d'un modèle. Les modèles obtenus permettent de décrire le processus et les interactions sous forme d'un diagramme.

Les fichiers de logs constituent une excellente source de données et peuvent être considérés comme des traces brutes qu'il faudra par la suite transformer en traces modélisées (sous forme d'un journal d'événements) afin de pouvoir les utiliser. Ces traces modélisées doivent porter les informations nécessaires représentant les activités afin de pouvoir en effectuer leur analyse.

Pour résumer, il faut sessioniser les actions utilisateurs afin de pouvoir en faire une analyse. Des champs additionnels, porteurs d'informations, peuvent être ajoutés aux trois précédemment cités (paramètres optionnels, courte description etc.). Mais que se passe t-il lorsque nous souhaitons générer un modèle à partir d'une masse de données volumineuse?

Nous allons, dans un premier temps, effectuer un état de l'art de la fouille de processus qui permettra de regrouper les différents modèles de processus existants, ainsi que les algorithmes présents dans la littérature. Nous aborderons ensuite notre cas d'étude et les perspectives qu'elle amène, puis nous conclurons.

2 État de l'art

2.1 Les différents modèles de processus

Les algorithmes de fouille de processus permettent d'obtenir différents types de modèles que nous allons présenter, chaque algorithme permettant d'obtenir en général un ou plusieurs des modèles présentés ici (il existe des équivalence entre certains modèles).

Tout d'abord nous avons les ***réseaux de Petri***, présenté pour la première fois par Murata [6], qui sont des modèles mathématiques avec représentation graphique, qui permettent d'exprimer des séquences d'évènements, avec des synchronisations et des partages de ressources. Ils sont composés de deux types de noeuds : des places et des transitions, reliés par des arcs. Si un arc relie une place à une transition il est dit "entrant", s'il relie une transition à une place, il est dit "sortant". Pour décrire l'état courant de ce type de système, on utilise un système de jeton qui sont associés aux places. L'évolution dans ce système est régie par un ensemble de règles qui décrivent les transitions (récupérer un objet dans un jeu par exemple).

Nous avons ensuite les ***Workflow Nets***, introduits par Van der Aalst [8] en 1997 et repris ensuite en 2011 [9], qui sont une sous-classe des réseaux de Petri. Ses caractéristiques les plus importantes sont d'avoir une place dédiée pour un "début" du processus et une "fin" du processus. Un tel modèle implique que pour chaque succession d'événements depuis la place initiale, il existe une séquence de transitions qui permet d'atteindre la place finale.

Il existe également les ***Directly-Follows Graphs*** (DFG) qui sont des graphes où chaque noeud représente une activité existante dans le journal d'événements et les arêtes dirigées sont présentes entre les noeuds s'il existe au moins une trace dans le journal d'événement où l'activité "source" est suivie par l'activité "cible". Il est également facile de représenter des mesures comme la fréquence (nombre de fois où l'activité source est suivie par l'activité cible) et des indicateurs de performance (moyenne de temps écoulé par exemple entre les deux activités).

2.2 Les différents algorithmes de fouille de processus

Au travers de la littérature, nous pouvons observer l'existence de plusieurs algorithmes de fouille de processus, chacun présentant ses forces et ses faiblesses. En effet, l'évolution croissante du domaine a permis d'affiner les différentes techniques et permettent de travailler sur des traces de qualités différentes (traces bruitées, incomplètes, non structurées) afin d'en obtenir tout de même une représentation du modèle de processus. Généralement, les différentes techniques de découverte de processus se basent sur l'identification des liens entre les activités dans un ensemble de traces. On peut identifier quatre relations qui permettent de définir deux activités quelconques a_1 et a_2 :

- la succession directe, indique que a_1 est immédiatement suivi de a_2 dans certaines séquences d'événements, elle est notée $a_1 \geq a_2$;
- la causalité indique que pour toutes les traces, a_1 est situé toujours avant a_2, il n'y a jamais a_2 avant a_1. Il faut la différencier de la succession directe dans le sens où a_1 et a_2 peuvent être espacés d'une où plusieurs activités. Elle est notée $a_1 \rightarrow a_2$;
- parallèle si dans une trace on obtient $a_1 \geq a_2$ et dans une autre $a_2 \geq a_1$. Elle est notée $a_1 \| a_2$;
- le choix lorsqu'il n'y a jamais $a_1 \geq a_2$ ni $a_2 \geq a_1$. Il est noté $a_1 \neq a_2$.

L'un des premiers algorithmes à avoir vu le jour pour la découverte de processus est l'algorithme α par Van der Aalst et al. [7]. Il consiste à identifier les différentes relations observées entre les activités dans les logs et se fonde sur les 4 relations énoncées précédemment. Il est en revanche incapable de découvrir des boucles ou des activités dupliquées. Certaines améliorations à cet algorithme ont été apportées avec l'apparition de α^+, puis α^{++}. Cette famille d'algorithmes souffre toujours cependant du bruitage dans le journal d'événements.

Nous avons ensuite l'Heuristic Miner, développé par Weijters et al. en 2006 [12] et amélioré en 2011 [11]. Comme son nom l'indique, il se base sur une approche heuristique afin de résoudre les différents problèmes rencontrés par l'algorithme α. La différence fondamentale entre α et Heuristic Miner vient du fait qu'Heuristic Miner se base sur des mesures statistiques afin de déterminer les relations entre activités. Les inconvénients de cet algorithme sont que les modèles produits contiennent des anomalies et ne sont pas capables de rejouer la majorité des logs.

Un autre algorithme est l'Inductive Miner, découvert en 2014 par Leemans et al. [4]. Il a été développé afin de produire des modèles de processus qui permettent de rejouer la majorité des logs et qui ne contiennent pas d'anomalies. Il permet en outre de traiter efficacement un potentiel bruit dans les logs en supprimant les traces non fréquentes et permet de traiter le problème des logs non complets et est a l'heure actuelle l'algorithme le plus utilisé en fouille de processus. En revanche il n'est pas en mesure d'identifier des motifs complexes et non locaux de contrôle de processus.
Un tableau permettant de comparer les algorithmes est disponible en annexe. (2)

3 Cas d'étude

Notre cas d'étude est un jeu sérieux (serious game) ayant eu lieu au Muséum d'Histoire Naturelle de La Rochelle en 2017. Ce jeu sérieux permettait de faire découvrir une exposition au public qui devait alors répondre à de multiples questions en rapport avec l'exposition, posées par un robot humanoïde Nao (l'automate des possibles est présent en annexe, figure 1).

Les joueurs étaient tous identifiés à l'aide de symboles reconnaissables par le robot, afin de bien associer un événement effectué au bon joueur (répondre à une question par exemple). Cette activité a entraîné la création de nombreux logs (environ 6500 traces) pour chaque action qui a été effectuée par les participants. Plusieurs questions sont soulevées à la suite de cette expérimentation :

- L'exploitation des logs peut-elle conduire à l'amélioration future de cette activité ?
- Existe t-il des catégories de joueurs qui se dégagent dans les logs ?

Ces deux questions correspondent bien à l'esprit de la fouille de processus dans son objectif. La première question est intéressante pour observer les différents problèmes qu'il y a pu avoir lors de l'activité et éviter certaines erreurs ou abandons des participants.

En effet, lors du déroulement de l'activité, des traces indiquent la présence de multiples erreurs de reconnaissance vocale ou des problèmes d'identification du joueur, ce qui a engendré, à terme, des abandons sans doute dûs à une frustration de la part du joueur. Cela a pu donner une piste de résolution pour un emploi futur de cette activité mais aussi d'autres activités employant le même mode opératoire avec le robot.

La difficulté de la deuxième question réside dans le fait que chaque activité possède des paramètres bien distincts associés aux paramètres du robot et de l'activité (mouvements qui doivent être effectués par le robot, rapidité d'élocution du texte, identification de la question et texte à prononcer, réponse du joueur, etc.). Ces paramètres pourraient être bien évidemment supprimés de la trace mais une grande partie de la sémantique serait alors perdue. Du fait de la richesse combinée de toutes ces traces, l'exploitation du graphe de tous les joueurs n'est pas possible et donne lieu à un graphe incompréhensible doté de toute sorte de noeuds et d'arêtes entrelacés.

Afin de pouvoir extraire des informations des graphes des joueurs, il faut procéder à l'extraction du graphe du joueur demandé et visualiser manuellement ce graphe. Bien que les comportements communs se reflètent assez bien entre deux joueurs lorsque l'on se penche individuellement sur ces graphiques (voir figure 2 en annexe, on remarque qu'ils ont tous les deux donnés les mêmes réponses, donnant ainsi un comportement commun. Les graphiques ont été extraits à l'aide de la bibliothèque Python *pm4py*), effectuer cette tâche pour un jeu de données important est fastidieux. Il nous faut donc proposer des améliorations afin de pouvoir exploiter ces logs sans retirer d'informations, en conservant la sémantique des données.

4 Piste exploratoire

Une piste à explorer est l'utilisation de la fouille de séquences. En effet, un journal d'événements peut être assimilé à une séquence d'événements survenue avec un ordre précis. Mais le fait est que, comme nous l'avons vu dans la section 2, ce journal d'événements est composé d'un ensemble de données hétérogènes qui influent grandement sur la sémantique des données et que d'autres champs peuvent être ajoutés aux trois principaux. Une analyse avec un outil spécialisé peut être intéressant.

Dans le cadre de ses travaux de recherche, le Laboratoire Informatique, Image et Intéraction (L3i), de La Rochelle Université, propose un outil, centré sur l'analyste de données, capable de calculer un treillis de concept à partir de données complexes (type séquences) et hétérogènes, nommé **GALACTIC** (**GA**lois **LA**ttices, **C**oncept"'e **T**heory, **I**mplicational systems and **C**losures). Cet outil utilise l'algorithme NextPriorityConcept [1] et permet de calculer des concepts pour des données hétérogènes et complexes en entrée.

On obtient alors un treillis de concept qui représente les données sous forme de clusters hiérarchiques et ce de manière non supervisée. Chaque concept est décrit par un ensemble de prédicats, chaque prédicat étant spécifique à chaque caractéristique. Par exemple, une caractéristique numérique peut être décrite sous la forme d'un prédicat de la forme "*est plus petit/grand que n*", n étant une valeur numérique. Une séquence peut, elle, être décrite sous forme de prédicat de la manière suivante : "*correspond à la sous-séquence s*".

L'utilisation de cet outil permettrait donc d'effectuer une autre approche et d'obtenir un modèle de données plus compréhensible pour la même masse de données.

5 Conclusion

La fouille de processus est un enjeu très intéressant pour l'amélioration des jeux sérieux et l'extraction de connaissances associées à une activité. Mes expérimentations à court terme vont consister à me servir de l'outil Galactic afin d'extraire la connaissance des logs et comparer les approches évoquées dans ce papier.

6 Remerciements

Nous remercions le projet ANR SmartFCA Grant ANR-21-CE23-0023 de l'Agence Nationale Française de Recherche qui permet le financement de ce papier.

References

1. C. Demko, K. Bertet, C. Faucher, J.F. Viaud, and S. O. Kuznetsov. NextPriorityConcept: A new and generic algorithm computing concepts from complex and heterogeneous data. *Theoretical Computer Science*, 845:1–20, 2020.
2. A. Godziński, A. Stroinski, W. Piątek, and A. Stroiński. Pattern recognition in games using process mining. In *2022 International Symposium on Electrical, Electronics and Information Engineering (ISEEIE)*, pages 42–45, 2022.
3. J. A. C. Hernández, M. P. Duarte, and J. M. Dodero. An architecture for skill assessment in serious games based on event sequence analysis. In *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM 2017, New York, NY, USA, 2017. Association for Computing Machinery.
4. Sander JJ Leemans, D. Fahland, and W. MP Van Der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *Business Process Management Workshops: BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers 11*, pages 66–78. Springer, 2014.
5. R. S. Mans, M. H. Schonenberg, M. Song, W. M. P. van der Aalst, and P. J. M. Bakker. Application of process mining in healthcare – a case study in a dutch hospital. In Ana Fred, Joaquim Filipe, and Hugo Gamboa, editors, *Biomedical Engineering Systems and Technologies*, pages 425–438, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
6. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
7. W. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
8. W. M. P. Van der Aalst. Verification of workflow nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, pages 407–426, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
9. W. M. P. Van Der Aalst, K. M. Van Hee, A. H. M. Ter Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, and M. T. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal aspects of computing*, 23:333–363, 2011.
10. W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713–732, 2007.
11. AJMM Weijters and J. T. S Ribeiro. Flexible heuristics miner (fhm). In *2011 IEEE symposium on computational intelligence and data mining (CIDM)*, pages 310–317. IEEE, 2011.
12. AJMM Weijters, W. MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166(July 2017):1–34, 2006.

Annexes

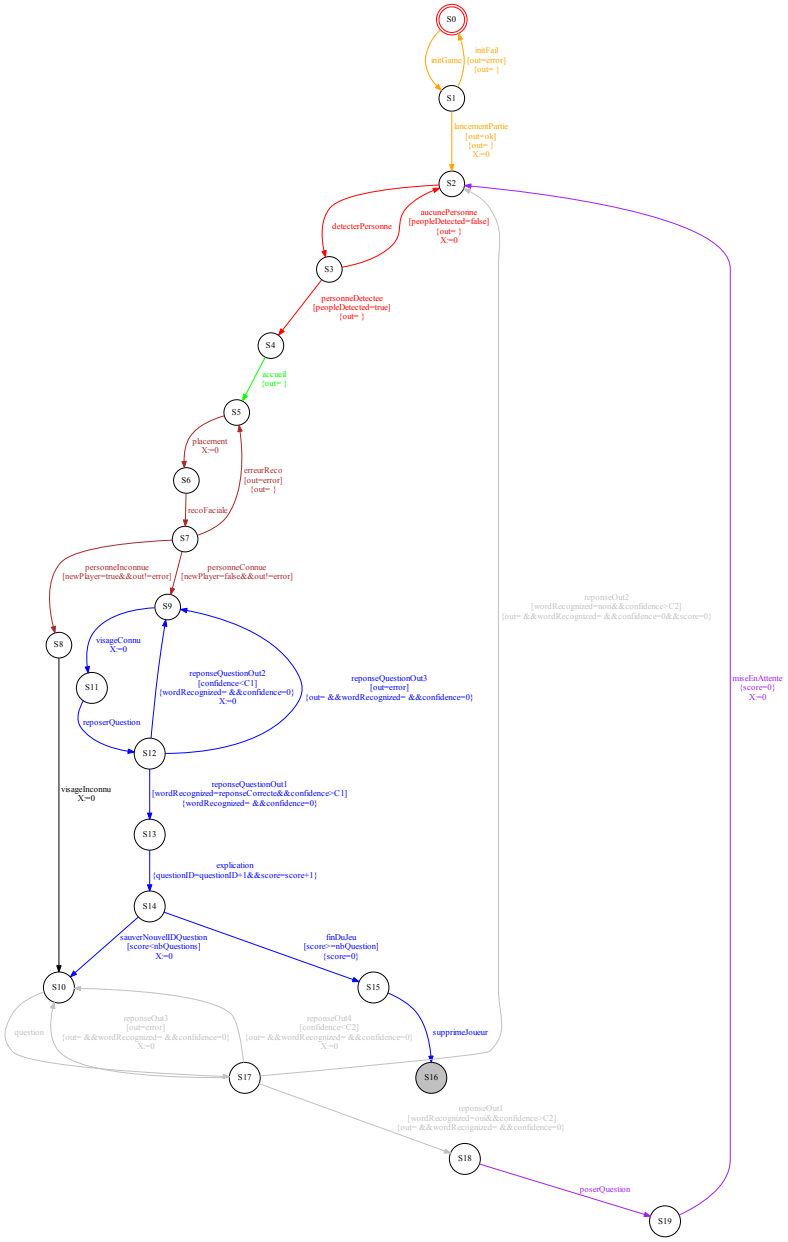

Fig. 1. Graphe des possibles de l'expérimentation du Muséum d'Histoire Naturelle

	Bruité	**Incomplet**	**Log réel**
α^+	non supporté	non supporté	non supporté
Heuristic Miner	supporté	non supporté	non supporté
Inductive Miner	supporté	supporté	supporté

Tableau 2. Tableau d'analyse des supports de logs pour les algorithmes cités

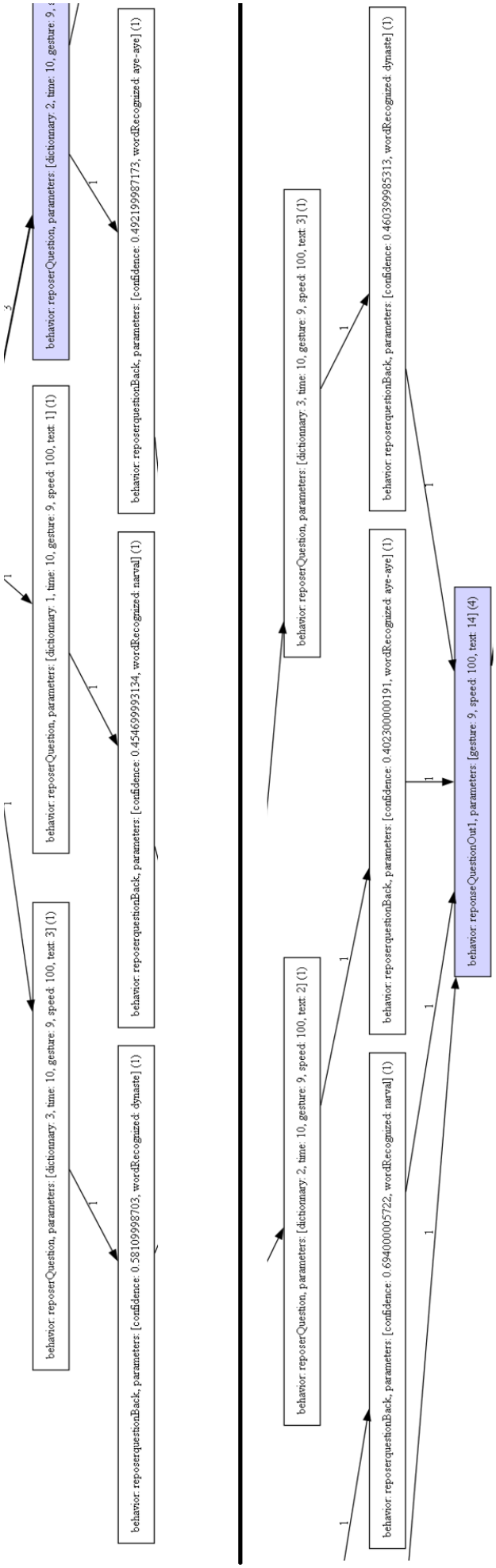

Fig. 2. Comparaison entre deux joueurs