Optimisation du temps de réponse du système de recommandation pour l'évaluation MIMICRY

Damien Follet, Nicolas Delestre, Nicolas Malandain, Laurent Vercouter

Optimisation du temps de réponse du système de recommandation pour l'évaluation MIMICRY

Damien Follet, Nicolas Delestre, Nicolas Malandain, Laurent Vercouter[1]
[1] LITIS, INSA Rouen, 76800 St Etienne du Rouvray, France
prénom.nom@insa-rouen.fr

Résumé. L'évaluation par grille d'évaluation est de plus en plus répandue. Elle facilite l'apprentissage des apprenants mais augmente la charge d'évaluation des enseignants. Nous proposons un système de recommandation pour l'évaluation par grille de compétences, utile, utilisable, thématiquement indépendant et basé sur l'apprentissage artificiel. Dans cet article nous testons ce système sur des données réelles pour des compétences non triviales. Nous identifions deux limites fonctionnelles et proposons une méthode d'énumération des hypothèses qui repousse l'une de ces limites: le temps de réponse du système.

Mots-clés. Système de recommandation, grille d'évaluation, apprentissage artificiel, indépendant du domaine, optimisation

Abstract. Assessment through criteria grids is increasingly widespread. It eases student's learning but also makes assessment harder for teachers. We present a useful, usable and domain-independent criteria-grid assessment recommender system based on machine learning methods. In this article we tested our system on real data for non-trivial skills and found two functional limits. Then we describe a new hypotheses enumeration method that extends one of these: the responding delay.

Keywords. Recommender system, criteria grid, machine learning, domain-independent, optimization

1 Introduction

De nos jours, les entreprises et le système éducatif utilisent de plus en plus l'évaluation diagnostique. Elle permet aux entreprises d'identifier quelle est la personne la plus adaptée à un poste donné. Elle permet également aux enseignants d'identifier précisément les lacunes d'un apprenant afin de lui conseiller de réviser les points précis du cours dont la maîtrise lui fait défaut.

Les grilles d'évaluation [1] permettent de réaliser une évaluation diagnostique. Une grille d'évaluation définit un ensemble de descripteurs qualitatifs en langue naturelle pour chaque niveau de maîtrise associé à chaque compétence. Le tableau 1 présente un extrait d'une grille d'évaluation pour une seule compétence.

Tableau 1. Exemple de grille d'évaluation

niveau \ compétence	Identifier les instructions du schéma conditionnel
acquis	toutes les instructions conditionnelles évaluées correctement identifiées
en cours d'acquisition	quelques instructions conditionnelles évaluées correctement identifiées
non acquis	aucune instruction conditionnelle évaluée correctement identifiée

Cependant la tâche d'évaluation s'alourdit pour l'enseignant puisque le nombre de critères à évaluer augmente significativement. Un système d'aide qui proposerait une évaluation des copies de la classe pourrait alléger le travail de l'enseignant et lui permettre de consacrer plus de temps à aider ses apprenants à combler les lacunes ainsi identifiées. Cela faciliterait donc indirectement l'apprentissage de ces apprenants. Les qualités recherchées pour notre système de recommandation sont : indépendant de la thématique, c'est-à-dire qu'il peut être utilisé pour corriger des examens portant sur n'importe quelle thématique, utilité, c'est-à-dire que les suggestions que le système fait à l'enseignant concordent au maximum avec l'évaluation que l'enseignant aurait fait lui-même, et utilisable, c'est-à-dire à la fois assez rapide : notre objectif est que le système réponde à l'enseignant « en temps réel », autrement dit que le temps de réponse du système, c'est-à-dire le temps de calcul nécessaire au système pour intégrer les informations d'une copie, soit de moins d'une seconde, et requérant très peu de copies évaluées manuellement par l'enseignant.

Nous avons présenté dans [2] le fonctionnement d'un système d'aide à l'évaluation conçu pour les grilles d'évaluation, intitulé MIMICRY (pour *MIMetic Interactive Criteria grid Recommender for You*). Nous avons testé ce système sur les données réelles pour des compétences triviales, c'est-à-dire des compétences qui ne dépendent que de la réponse de l'élève à une seule question.

Les résultats sur les compétences triviales sont prometteurs : le système MIMICRY a été conçu pour être indépendant de la thématique, s'est avéré très utile avec un taux d'erreur de 0 % et utilisable avec un temps de réponse moyen du système de 4,00ms et moins de 36 % des copies évaluées manuellement. Il reste cependant à confirmer ces résultats sur des données réelles avec des compétences non triviales, c'est-à-dire dépendant de la réponse à plus d'une question.

L'objectif de cet article est ainsi de tester notre système MIMICRY sur des données réelles avec des compétences non triviales et d'identifier expérimentalement certaines des limites fonctionnelles de notre système. La structure de l'article en découle : nous commencerons par résumer le fonctionnement de notre système MIMICRY, puis nous le testerons sur des compétences non triviales. Nous finirons par proposer et tester expérimentalement une nouvelle méthode d'énumération des hypothèses consistantes qui permet de réduire le temps de réponse moyen du système et de pouvoir utiliser efficacement notre système dans davantage de situations.

2 Résumé du fonctionnement du système MIMICRY

Nous commencerons par résumer le fonctionnement actuel du système MIMICRY. Plus de détails sont disponibles dans les articles [2] et [3].

2.1 Objectifs et principes de fonctionnement de MIMICRY

L'objectif du système MIMICRY est de suggérer les niveaux de compétences appropriés pour chacune des copies de la classe GS (pour *Global Set*), tels que l'enseignant lui-même les aurait évaluées. Pour ce faire, le système va tenter d'imiter l'enseignant en s'appuyant sur les informations contenues dans les copies de l'ensemble TS (pour *Training Set*) des exemples manuels dont il demandera au fur et à mesure l'évaluation à l'enseignant.

2.2 Usage prévu du système

Plus précisément, le système va initialement choisir un certain nombre k de copies dont il demandera l'évaluation à l'enseignant.

En se basant sur les informations qu'il en tirera, il va essayer de comprendre quelles sont les questions qui sont pertinentes pour l'enseignant pour évaluer une compétence fixée, en faisant un certain nombre de suppositions. Il pourra ensuite en déduire quelle évaluation est appropriée pour les copies restantes de GS. Dans certains cas il ne disposera pas d'assez d'informations pour suggérer l'évaluation qui lui semble appropriée pour un certain nombre de copies : il va alors sélectionner l'une de ces copies et demander à l'enseignant comment il aurait évalué cette copie, afin de compléter les informations dont il dispose. Le système considère avoir assez d'informations lorsqu'il est capable de proposer une évaluation pour chaque copie de GS. Bien entendu, chacune de ces évaluations peut être correcte ou erronée : cela dépendra des suppositions que le système a fait ainsi que de l'homogénéité des copies dont il aura demandé l'évaluation à l'enseignant. Dans le cas d'évaluations erronées, l'enseignant pourra fournir un nouvel exemple au système afin que ce dernier rectifie son raisonnement et propose une nouvelle évaluation de toutes les copies.

Les sections suivantes précisent comment le système fait ces suppositions durant la phase d'apprentissage de l'algorithme, puis comment les copies sont sélectionnées initialement et à chaque itération lorsqu'il reste certaines copies que le système n'est pas encore capable d'évaluer.

2.3 Algorithme d'apprentissage

L'objectif de l'algorithme d'apprentissage est d'identifier l'hypothèse explicative de TS la plus probable, afin de s'en servir plus tard pour suggérer les niveaux de compétences appropriés associés aux copies de DS=GS\TS (pour *Decision Set*).

Plus en détail, comme décrit graphiquement dans la figure Fig. 1, il s'agit d'énumérer toutes les hypothèses explicatives possibles, puis de calculer la probabilité de chacune d'elles d'être au cœur du raisonnement de l'enseignant, d'identifier celle qui est la plus probable et d'en déduire la fonction d'évaluation partielle qui nous permettra d'attribuer un niveau de compétence pour chaque copie. Nous allons ici détailler chacune des notions mentionnées.

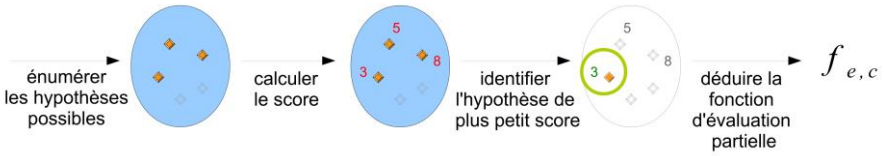


Fig. 1. Décomposition de l'algorithme d'apprentissage.

Dans notre contexte, les hypothèses explicatives représentent tous les sous-ensembles de questions qui peuvent être pertinentes d'après l'enseignant afin d'évaluer la compétence c. L'ensemble des hypothèses explicatives est donc l'ensemble des parties des questions privé de l'ensemble vide : on peut donc le représenter par un treillis. Certaines hypothèses sont absurdes vis-à-vis des informations contenues dans TS : on dira alors qu'elles sont inconsistantes au sens de la logique déductive.

Formellement, soit QS l'ensemble des questions (pour *Question Set*), une hypothèse explicative s est dite consistante par rapport à TS pour la compétence c si et seulement si tous les couples de copies de TS dont les réponses aux questions de s sont identiques, sont associées aux mêmes niveaux de compétence pour c.

En nous basant sur une approche proche de [4] et [5], nous estimerons que plus une hypothèse consistante s est associée à un faible score h(s), plus il est probable que les questions de s soient les seules questions pertinentes pour l'enseignant. Ce score, qui permet de choisir la meilleure hypothèse, se calcule grâce à la formule :

$$h(s)=\#s \times \#TS|_s$$

Où #s représente la taille de l'hypothèse s c'est-à-dire le nombre de questions auxquelles elle fait référence, et où $\#TS|_s$ représente le nombre de combinaisons de réponses différentes si l'on ne prend en compte que les questions de s et les copies de TS.

L'algorithme d'apprentissage se résume donc à la formule :

$$s^* = amin_{s \in C} h(s)$$

Où $s \in C$ représente l'ensemble des hypothèses consistantes, s^* représente l'hypothèse la plus probable que l'on cherche à identifier, C représente l'ensemble des hypothèses consistantes et h représente le score précédemment décrit. Une fois identifiée, l'hypothèse la plus probable s permettra au système de déduire la fonction d'évaluation partielle f_e la plus probable pour la compétence c à partir de TS :

$$f_e = TS|_s$$

En pratique, le nombre d'hypothèses consistantes augmente très rapidement avec le nombre de questions comme le montre la figure Fig.2 avec un exemple à 5 questions (hypothèses consistantes en bleu et inconsistantes en rouge hachuré).

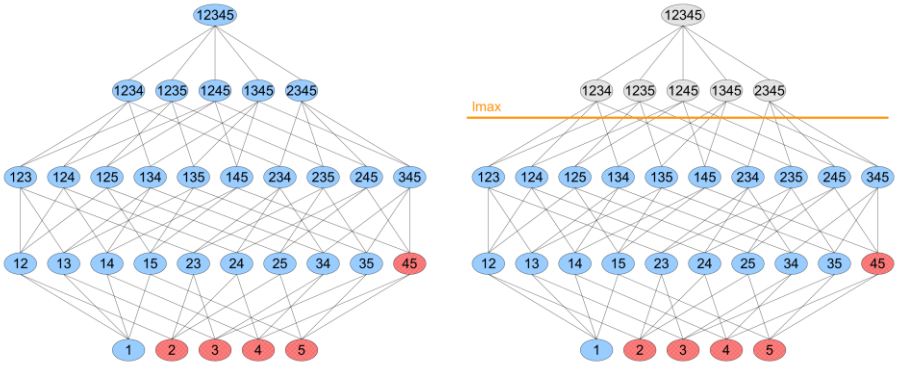


Fig.2. Hypothèses consistantes : C (gauche) et C limités par $l_{\max}$(droite)

En étudiant les propriétés du score h, nous nous sommes aperçus qu'il était possible de limiter notre recherche. Il suffit de comparer les hypothèses de taille de plus en plus grande, en se limitant aux hypothèses de taille inférieures à

$$l_{max} = \lfloor h\,(s_0)/D_{c,TS} \rfloor$$

Où s_0 représente la meilleure hypothèses consistante de taille inférieure déjà trouvée, où $D_{\mathrm{c,TS}}$est le nombre de niveaux différents pour la compétence c présents dans TS et $\lfloor x \rfloor$ la partie entière par défaut de x. La figure 2 montre combien d'hypothèses sont écartées par $l_{\max}$ (hypothèses écartées en gris barré).

L'algorithme d'apprentissage se résume finalement à

$$s^* = amin_{s \in C \text{ tel } s < l_{max}} h(s)$$

2.4 Sélection de copies

Le système sélectionne initialement k copies afin d'en tirer suffisamment d'informations pour aiguiller ses suppositions. Puis plus tard, lorsqu'il reste encore des copies que le système ne peut évaluer, il doit en sélectionner l'une d'entre elles.

Dans les deux cas, notre système MIMICRY sélectionne actuellement les copies de manière aléatoire parmi les copies qu'il ne sait pas encore évaluer. Cette stratégie est sciemment naïve et pourra être améliorée prochainement grâce aux apports du domaine de l'apprentissage actif.

3 Expérience sur des compétences non triviales

Après avoir résumé le fonctionnement de notre système et après avoir testé ses performances sur des compétences triviales dans [2], nous allons tester ses performances sur des compétences non triviales. Nous commencerons par décrire le protocole de test utilisé, puis nous présenterons les résultats de ce test.

3.1 Protocole

Nous avons rassemblé nos données réelles en faisant passer à 89 apprenants de première année de spécialité d'une école d'ingénieur un QCM de 29 questions sur Moodle portant sur les connaissances en programmation qu'ils ont acquis pendant les années précédentes. L'enseignant du cours a évalué a posteriori leurs réponses à l'aide d'une grille d'évaluation afin de leur attribuer un niveau pour chacune des 7 compétences différentes, et on s'est assuré qu'il n'y avait pas d'incohérence de notation entre les copies des apprenants. On présente ici les résultats pour deux compétences : une compétence triviale c_1 : « Connaître et utiliser les instructions du schéma conditionnel » dépendant d'une seule question, et une compétence non triviale c_3 : « Connaître et utiliser les principes et outils de l'architecture n-tiers » dépendant de trois questions. Ces deux compétences peuvent être associées à l'un des 3 niveaux : « Acquis », « En cours d'acquisition » et « Non acquis ». On dira ainsi que $c_1$est de complexité 1 et $c_3$est de complexité 3.

Nous avons extrait les réponses à 15 questions sur 29. Ces 15 questions ont été sélectionnées aléatoirement et nous nous sommes assurés qu'elles contiennent les questions pertinentes pour l'enseignant. Nous avons fourni initialement k=10 copies sélectionnées aléatoirement par le système. A chaque itération du système nous avons noté le nombre de copies manuellement évaluées jusque-là, le taux d'erreur de l'algorithme en comparant les prédictions de l'algorithme avec les évaluations réelles de l'enseignant ainsi que le temps de réponse moyen du système. Par souci de fiabilité nous avons répété chaque expérience 1000 fois et calculé la moyenne des résultats obtenus.

3.2 Résultats

Le tableau 2 présente les résultats de l'expérience : moyenne et intervalle de confiance à 95 %.

Tableau 2. Résultats de l'expérience sur c1 et c3

compétence	complexité	$\%err_f$ (%)	durée apprentissage (ms)	durée décision (ms)	nombre itérations
c_1	1	$0,28\% \pm 0,19\%$	$0,795 \pm 0,00956$	$0,096 \pm 0,00246$	$5,09 \pm 0,08$
c_3	3	$9,72\% \pm 0,5\%$	$357,4 \pm 20,9$	$0,121 \pm 0,00202$	$8,94 \pm 0,43$

On constate que le temps de réponse moyen du système, et plus précisément la durée moyenne de la phase d'apprentissage, augmente très fortement avec la complexité de la compétence évaluée. Nous avons effectué des régressions avec plusieurs modèles afin d'estimer la vitesse de croissance de la durée d'apprentissage : celle-ci dépasse 1 seconde à partir d'une complexité de 6 avec le modèle linéaire ($y = a * x + b$) et à partir d'une complexité de 4 avec les modèles puissance ($y = a * x^b$) et exponentiel ($y = a * e^{b*x}$). Cela s'avère rapidement incompatible avec les contraintes d'un système de recommandation en temps réelles telles que décrites dans l'introduction de cet article. On constate également que le taux de désaccord augmente substantiellement avec la complexité de la compétence évaluée.

Ces résultats suggèrent donc deux limites fonctionnelles : l'augmentation du temps de réponse moyen du système, et l'augmentation du taux de désaccord avec la complexité de la compétence. La seconde limite dépend principalement de la qualité de la sélection des copies et du nombre initial de copies k, et relève des problématiques du domaine de l'apprentissage actif : nous traiterons ultérieurement de cette limite. Nous nous concentrerons donc dans la suite de cet article sur la première limite : l'augmentation du temps de réponse moyen du système.

Suite à ces résultats, nous avons cherché à identifier pourquoi la phase d'apprentissage requiert autant de temps.

3.3 Précisions

Pour comprendre ce qui nécessite autant de calcul pendant l'apprentissage, nous avons relevé combien de temps est nécessaire pour chacune des sous-parties de la phase d'apprentissage pour la compétence c3. On rappelle que la phase d'apprentissage se décompose en 4 sous-parties, comme décrit dans la partie 2.3 : l'énumération des hypothèses possibles, le calcul du score de chacune d'elles, l'identification de l'hypothèse la plus probable (score minimal) et enfin la déduction de la fonction d'évaluation partielle qui nous permettra de calculer le niveau de compétence des copies non encore évaluées comme l'aurait fait l'enseignant.

Le tableau 3 présente la durée de calcul de chaque sous-partie pendant la phase d'apprentissage ainsi que le pourcentage de temps total de la phase d'apprentissage de chaque sous-partie : on constate que la sous-partie d'énumération des hypothèses possibles représente plus de 82 % de la durée de la phase d'apprentissage et qui représente elle-même 99,9 % du temps de réponse moyen du système.

Tableau 3. Durée de calcul de chaque sous-partie de l'algorithme d'apprentissage

	énumération	calcul de score	identification de s^*	déduction de f_e	total apprentissage
durée (ms)	278,187	57,933	0,012	0,046	336,178
pourcentage du total d'apprentissage (%)	82,7	17,2	<0,1	<0,1	100

En réduisant la durée de cette sous-partie, nous espérons donc réduire substantiellement le temps de réponse moyen du système. La section suivante présente notre contribution sur l'énumération des hypothèses possibles.

4 Enumération des hypothèses possibles

En étudiant l'espace des hypothèses consistantes et les propriétés de la fonction de score, nous avons découvert qu'il n'est pas toujours nécessaire de comparer le score de toutes les hypothèses consistantes. Lorsque la fonction de score vérifie une certaine propriété, les hypothèses consistantes qui ont le meilleur score se trouvent toujours dans un sous-ensemble particulier des hypothèses consistantes. Ce sous-ensemble regroupe les hypothèses dites minimales consistantes. Nous noterons ce sous-ensemble MiC (pour hypothèses MInimales Consistantes). Ainsi il nous suffira d'énumérer et de comparer le score des hypothèses de MiC pour déterminer l'hypothèse la plus probable parmi toutes les hypothèses possibles, réduisant ainsi les calculs nécessaires pour chaque itération.

Dans la partie qui suit, nous nous attacherons à définir MiC et la propriété que le score doit vérifier pour pouvoir simplifier l'énumération des hypothèses possibles. Nous présenterons ensuite à titre indicatif l'algorithme d'énumération de MiC que nous avons implémenté et terminerons en comparant expérimentalement sur des données réelles les durées moyennes d'apprentissage des deux méthodes d'énumération des hypothèses possibles.

4.1 Sous-ensemble MiC

Nous présenterons ici la définition et les caractéristiques du sous-ensemble MiC des hypothèses minimales consistantes, basé sur la notion d'hypothèse consistante, d'hypothèse simplifiée et d'hypothèse minimale consistante.

On dira qu'une hypothèse s′ est une hypothèse simplifiée de s si et seulement si $s' \subset s \wedge s' \neq s$.

Une hypothèse s est dite minimale consistante par rapport à TS si et seulement si elle est consistante par rapport à TS et que toutes les hypothèses s′ simplifiées de s sont inconsistantes par rapport à TS.

L'ensemble de toutes les hypothèses minimales consistantes est noté MiC. La figure Fig.3 montre les ensembles C et MiC limités par $l_{\max}$ sur un exemple à 5 questions. On peut ainsi apprécier combien d'hypothèses consistantes le système doit générer et comparer par le score dans chacun des cas (hypothèses inconsistantes en rouge hachuré, consistantes considérées en bleu et consistantes écartées en gris barré).

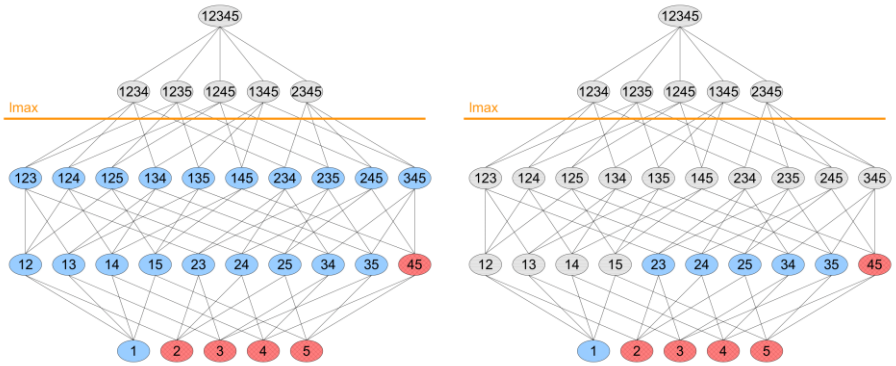


Fig.3. Énumérations limitées par $l_{\max}$: C, 20 hypothèses (gauche) et MiC, 6 hypothèses (droite)

4.2 Propriété pro-MiC sur le score

On dira que le score h est pro-MiC (il favorise les hypothèses de MiC) lorsque toute hypothèse consistante s′ simplifiée de s est associée à un meilleur score que celui de s. Formellement si et seulement si pour toute hypothèse consistante s, pour toute hypothèse s' simplifiée de s, $s' consistante \rightarrow h(s') < h(s)$ si la meilleure hypothèse est associée au score le plus faible.

Par récurrence sur la structure de treillis, on remarque que si le score vérifie cette propriété alors les hypothèses qui auront le meilleur score n'admettront aucune hypothèse simplifiée consistante. On rejoint ainsi la définition de l'ensemble MiC ; un score vérifiant la propriété précédemment citée attribuera en fait systématiquement les meilleurs scores aux hypothèses de MiC. Ainsi si le score est pro-MiC, l'hypothèse consistante s^* associée au meilleur score appartient à MiC.

Formellement : si h est pro-MiC, alors

$$s^* = amin_{s \in C \ tel \ s<l_{max}} h(s) = amin_{s \in MiC \ tel \ s<l_{max}} h(s)$$

Avec C représentant l'ensemble des hypothèses consistantes par rapport à TS et MiC l'ensemble des hypothèses minimales consistantes par rapport à TS. Comme la fonction de score h(s) de notre système est pro-MiC, nous allons donc pouvoir réduire le nombre d'hypothèses à énumérer et à comparer. On remarque que d'autres fonctions de score, comme celui de la méthode Minimum Description Length [8], vérifient la propriété pro-MiC et sont donc susceptibles d'être améliorées par notre approche.

En s'appuyant sur ce résultat théorique, nous avons pu développer un algorithme qui énumère les hypothèses de l'ensemble MiC. La partie suivante présente notre cheminement.

4.3 Algorithme récursif d'énumération de MiC

L'ensemble MiC est équivalent à l'ensemble S de l'espace des versions [6,7] à quelques redéfinitions près, mais il ne nous est pas possible d'utiliser directement l'algorithme de l'espace des versions pour calculer MiC pour des raisons de complexité algorithmique. Le temps de calcul requis pour cet algorithme s'avérerait trop élevé dans notre contexte.

Nous avons donc développé un algorithme récursif qui va construire directement chacune des hypothèses de $S \equiv$ MiC pour la compétence c. Cet algorithme repose sur la reformulation de la définition d'une hypothèse minimale consistante en plusieurs propriétés à vérifier sur les questions qui forment cette hypothèse. Il est alors possible de générer les hypothèses minimales consistantes d'une taille donnée l_i, en connaissant les hypothèses minimales consistantes de taille inférieure $l<l_i$.

Algorithm 1 Calcul des MiC^l. appel initial : $MiC\_l(l_i, QS, MiC^{<l_i}, MaI^{\geq l_i})$

function $MiC\_l(l_i$,*pool*,F_c,$F_i)$
  if $l_i = 1$ **then**
    $pool' \leftarrow pool \setminus (\bigcup_{e \in F_i} \mathcal{P}_1(e) \cup \bigcup_{e \in F_c \ as \ \#e=1} e)$
    return $\mathcal{P}_1(pool')$
  else
    $pool' \leftarrow pool \setminus (\bigcup_{e \in F_c \ as \ \#e=1} e)$
    $X \leftarrow \emptyset$
    for $elu \in pool'$ **do**
      $F_c' \leftarrow \{x' \ as \ \exists x \in F_c \ as \ (elu \in x \rightarrow x' = x \setminus \{elu\} \wedge elu \notin x \rightarrow x' = x)\}$
      $F_i' \leftarrow \{x' \ as \ \exists x \in F_i \ as \ (elu \in x \wedge x' = x \setminus \{elu\})\}$
      $X \leftarrow X \cup (\{elu\} \times MiC\_l(l_i - 1, pool' \setminus \{elu\}, F_c', F_i'))$
    end for
    return X
  end if
end function

Nous avons implémenté l'algorithme 1 tel que $P_1(e)$ représente l'ensemble des parties de taille 1 de l'ensemble e, $\cup$, $\setminus$ et $\times$ représentent respectivement l'union, le complément relatif et le produit cartésien entre ensembles, QS représente l'ensemble des questions, $MiC^{<l_i}$ représente l'ensemble des hypothèses minimales consistantes de taille inférieure à l_i (précédemment calculées), $MaI^{\geq l_i}$ représente l'ensemble des hypothèses maximales inconsistantes de taille supérieure ou égale à l_i. Ce dernier ensemble peut être filtré par taille à partir de l'ensemble MaI qui correspond à l'ensemble de l'ensemble des questions associées à des réponses identiques entre deux copies de TS associées à des niveaux de compétence différents pour la compétence c.

4.4 Comparaison expérimentale des énumérations C et MiC

Après avoir implémenté cet algorithme récursif d'énumération des MiC de taille l_i, nous avons conduit une expérience afin de mesurer expérimentalement son apport :

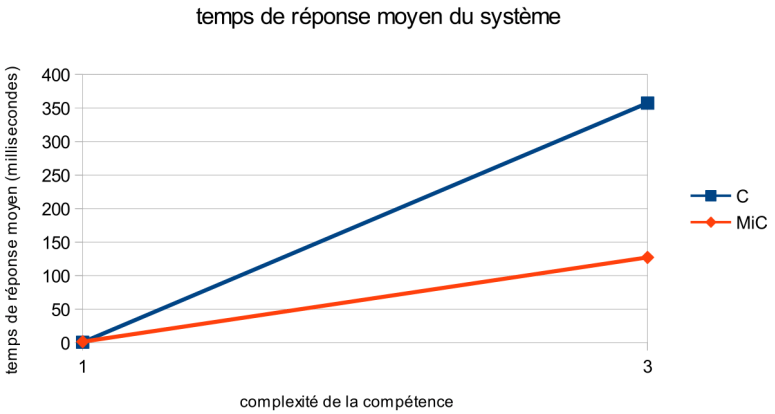
nous avons répété l'expérience décrite dans la section 3.1 en utilisant simplement l'énumération MiC à la place de l'énumération C.

Le tableau 4 et le diagramme Fig.4 reprennent les résultats de cette expérience. Ils montrent que la fiabilité et la vitesse de convergence des deux versions du système sont similaires, puisque le taux de désaccord final et le nombre moyen d'itérations de la version MiC sont respectivement compris dans l'intervalle de confiance à 95 % de ceux de la version C. Ils montrent également que comme nous l'avions espéré, la durée d'apprentissage et donc le temps de réponse moyen du système sont substantiellement moins élevés dans la version MiC que dans la version C : nous avons réduit le temps de réponse moyen du système de 64.4% pour la compétence c3 de complexité 3.

Tableau 4. Performances du système MIMICRY, versions C et MiC

version	compétence	complexité	$\%err_f$ (%)	durée apprentissage (ms)	durée décision (ms)	nombre itérations
C	c_1	1	$0,28\% \pm 0,19\%$	$0,795 \pm 0,00956$	$0,0955 \pm 0,00246$	$5,09 \pm 0,08$
C	c_3	3	$9,72\% \pm 0,5\%$	$357,4 \pm 20,9$	$0,121 \pm 0,00202$	$8,94 \pm 0,43$
MiC	c_1	1	$0,16\% \pm 0,14\%$	$1,550 \pm 0,0348$	$0,095 \pm 0,006$	$5,1 \pm 0,07$
MiC	c_3	3	$9,95\% \pm 0,53\%$	$127,2 \pm 10,7$	$0,119 \pm 0,0014$	$9,05 \pm 0,42$

Fig.4. Comparaison du temps de réponse moyen du système avec C et avec MiC



Nous avons encore une fois effectué des régressions avec plusieurs modèles afin d'estimer la différence de vitesse de croissance de la durée d'apprentissage pour la version MiC. Le temps de réponse du système dépasse 1 seconde à partir d'une complexité de 16 pour MiC contre 6 pour C avec le modèle linéaire ($y = a * x + b$), à partir d'une complexité de 5 pour MiC contre 4 pour C avec le modèle puissance ($y = a * x^b$) et à partir d'une complexité de 4 pour MiC (complexité > 3,94) et C (complexité > 3.34) avec le modèle exponentiel ($y = a * e^{b*x}$).

La version MiC permet ainsi de rester utilisable pour des compétences de complexité plus élevées dans la plupart des modèles de régression envisageables.

5 Conclusion

Dans cet article, nous avons testé le système MIMICRY sur des données réelles avec des compétences non triviales et identifié deux limites potentielles : l'augmentation du temps de réponse moyen du système et du taux de désaccord final avec la complexité de la compétence évaluée. Nous nous sommes concentrés sur la première limite et avons présenté une nouvelle méthode pour réduire fortement le nombre d'hypothèses consistantes à considérer indépendamment de la fonction de score à condition qu'elle soit pro-MiC. Enfin nous avons mesuré expérimentalement le gain apporté en termes de temps de calcul : nous avons réduit le temps de réponse moyen du système de 64 % par rapport à la version antérieure pour une compétence de complexité 3.

En perspective, nous souhaiterions améliorer notre système au regard de la seconde limite potentielle : l'augmentation du taux final de désaccord avec l'enseignant grâce aux apports du domaine de l'apprentissage actif afin de sélectionner plus efficacement les copies à faire évaluer par l'enseignant et ainsi se rapprocher du taux de désaccord parfait de 0 % que les tests sur compétences triviales laissent espérer. Nous souhaiterions également quantifier précisément à partir de quelle complexité de compétence et de combien de questions notre système n'est plus capable de répondre en « temps réel ». Enfin nous envisageons d'implémenter notre prototype en tant que plugin sur la plateforme Moodle afin de le proposer comme outil pratique aux enseignants.

Références

1. Rust, C., Price, M., & O'Donovan, B. (2003). Improving students' learning by developing their understanding of assessment criteria and processes. Dans Assessment & Evaluation in Higher Education, 28(2), 147-164.
2. Follet, D., Delestre, N., Malandain, N., & Vercouter, L. (2014, Novembre). Définition expérimentale du cadre de fonctionnement d'un système d'évaluation par grille basé sur l'apprentissage articiel. Dans TICE 2014.
3. Follet, D., Delestre, N., Malandain, N., & Vercouter, L. (2013, Décembre). A three-step classification algorithm to assist criteria grid assessment. Dans NIPS 2013, Workshop Data Driven Education.
4. Legg. S., (2011). Solomonoff induction (technical report cdmtcs-030), centre for discrete mathematics and theoretical computer science, university of Auckland.
5. Solomonoff. Ray J. (2009). Algorithmic probability: Theory and applications. Dans Information Theory and Statistical Learning, pages 1–23. Springer.
6. Mitchell, Tom M. (1982). Generalization as search. Dans Artificial Intelligence 18 (2): 203–226.
7. Rendell, L. (1986). A general framework for induction and a study of selective induction. Dans Machine Learning 1 (2): 177–226.
8. Adriaans, P., & Vitanyi, P. (2007, Juin). The power and perils of MDL. Dans Information Theory, 2007. ISIT 2007. IEEE International Symposium on (pp. 2216-2220). IEEE.