Modèle d'apprentissage d'Anderson et algorithmique

Jacques Malouin, Faculté d'éducation, Université de Sherbrooke

Colloque AFDI - Québec - Avril 1994

TIRPA (Tutoriel Intelligent pour la Résolution de Problèmes en Algorithmique) est un système informatisé qui a été développé pour servir d'instrument d'observation et d'analyse de certains aspects du processus d'acquisition de connaissances. Il a été conçu en utilisant une théorie d'apprentissage explicite, à savoir la théorie ACT* de J. R. Anderson et son corollaire le Model Tracing permettant d'inférer l'état courant des connaissances d'un élève à partir de son comportement. Cette théorie et son application à des systèmes intelligents d'apprentissage ont démontré les influences mutuelles que pouvaient avoir les recherches en psychologie cognitive et celles sur les tutoriels intelligents.

L'expérience de TIRPA a permis d'apprécier certains aspects la théorie cognitive ACT* en tant que guide d'élaboration d'un système d'E.I.A.O. Ce qui est présenté ici réfère à certaines adaptations au modèle d'Anderson qui ont été rendues nécessaires pour pouvoir appliquer ce dernier non pas à la programmation dans un langage mais plutôt à l'algorithmique, c'est-à-dire essentiellement à une démarche de résolution de problème.

Dans un premier temps, il parait utile de rappeler certains éléments distinctifs du modèle d'Anderson. Ensuite, les connaissances à faire acquérir à l'apprenant dans le contexte de l'algorithmique comme domaine d'apprentissage sont identifiées. Sont abordées enfin, les questions associées à la représentation des connaissances à acquérir et les difficultés d'analyse des solutions formulées par l'élève, notamment en fonction de *la liberté de manoeuvre* de l'élève.

La modèle d'Anderson

John R. Anderson a proposé un modèle général d'apprentissage[1] qui s'appuie sur une théorie cognitive principale, l'ACT*, et constitue une réponse plausible à des questions relatives à

1Le modèle d'Anderson est bâti autour d'une théorie de la cognition fondée sur des hypothèses quant à l'organisation et à l'acquisition d'habiletés cognitives complexes. Il est décrit dans la théorie d'apprentissage ACT* (Adaptive Control of Thought) (Anderson, 1983), dans la théorie PUPS (PenUltimate Production System) qui a suivi (Anderson, 1986) et dans la méthodologie d'implantation du Model Tracing (Anderson, 1987).

l'acquisition d'habiletés intellectuelles complexes, à la façon de guider un élève durant un tel apprentissage et d'effectuer le design d'un système d'E.I.A.O. jouant le rôle de tuteur. Pour les fins des questions abordées ici, le modèle d'Anderson est défini en fonction des deux éléments distinctifs suivants: la modélisation des connaissances par des règles de production et les modalités d'acquisition de connaissances procédurales en contexte de résolution de problème.

*Modélisation des connaissances par des règles de production*EX "règles de production"§
Au coeur de la théorie ACT* d'Anderson, il y a la capacité de représenter les fonctions cognitives par un ensemble de règles de production. Ces règles représentent les façons correctes ou incorrectes pour un élève d'arriver à des solutions semblables ou différentes de celle du *modèle idéal*: c'est le *modèle de performanceEX "modèle de performance"§*.

Dans le *Model Tracing* d'Anderson, on distingue donc les connaissances que l'élève manifeste extérieurement pendant qu'il résout un problème, le *modèle de performance*, de celles qu'il possède véritablement, le *modèle d'apprentissage*, en se référant aux différentes règles de production représentant les connaissances du domaine. Essentiellement, le mécanisme permettant le passage d'un modèle à l'autre est le suivant: dans le modèle de performance, on observe le comportement de l'élève en situation de résolution de problème et on tente d'identifier un ensemble de règles ou mal-règles du système qui pourrait expliquer un tel comportement. Cette identification permet de suivre les états cognitifs de l'élève en temps réel.

Acquisition de connaissances procédurales en contexte de résolution de problème
Dans la théorie ACT*, une connaissance est d'abord acquise *déclarativement*, quand l'élève lit ou entend une représentation de cette connaissance. Dans cette première étape, il s'agit d'une compréhension uniquement langagière. C'est l'expérience, c'est-à-dire l'accomplissement d'une tâche (ou plusieurs) impliquant l'utilisation de cette connaissance, qui permet la réorganisation et la conversion de celle-ci en *procédure*. Anderson appelle *compilation* le passage de l'état déclaratif à l'état procédural d'une connaissance sous l'effet d'un apprentissage en contexte de résolution de problème. Un tel apprentissage est efficace quand il fournit un ensemble de conditions permettant à l'élève d'encoder à la fois l'applicabilité de la connaissance ainsi que sa pertinence en rapport avec le but poursuivi. C'est l'idée centrale du *learning by doingEX "learning by doing"§*.

Pour qu'un système d'enseignement soit à même de sélectionner les tâches qui vont optimiser l'apprentissage d'un élève, il faut que cette sélection soit effectuée en fonction de ce que l'élève connaît et des connaissances requises pour accomplir les différentes tâches. C'est pour atteindre cet objectif qu'Anderson a développé et mis en oeuvre dans ses systèmes la méthodologie du *Model Tracing,* méthodologie qui permet d'apprécier les changements dans les connaissances de l'élève à mesure que ce dernier progresse dans la résolution de différentes tâches.

L'algorithmique comme domaine d'apprentissage

Avec l'algorithmique comme domaine d'apprentissage, l'accent est mis sur le volet *planification* d'une activité. C'est la résolution de problème à l'état pur. Mais l'algorithmique doit *s'incarner* dans une activité pour être exercée. La programmation est souvent utilisée comme champ d'application de l'algorithmique, en particulier parce qu'elle implique l'élaboration de stratégies et la décomposition d'un problème en sous-problèmes.

*Liberté de manoeuvre*EX "Liberté de manoeuvre"§
Les habiletés cognitives à acquérir en algorithmique se réfèrent essentiellement (1) aux stratégies et (2) à la logique de décomposition (ou à l'utilisation des structures de décomposition) permettant à un élève de décomposer un problème en sous-problèmes jusqu'à ce qu'il atteigne un niveau de raffinement correspondant à un ensemble structuré de commandes dans le langage de programmation utilisé. Dans ce contexte, de nombreuses stratégies peuvent être utilisées avec succès pour arriver à une solution. Ces stratégies auront des formulations potentiellement très différentes quant au fond et quant à la forme; en outre, le choix d'une de ces stratégies pourra avoir une répercussion directe sur la facilité ou la difficulté de résolution complète du problème.

C'est lors de cette recherche générale dans l'espace-problème que les élèves sont le plus susceptibles de se distinguer les uns des autres et d'innover dans la solution adoptée. Il est donc impératif qu'un élève dispose d'une grande liberté de manoeuvre afin d'explorer et/ou d'expérimenter différentes stratégies ou pistes de solution.

Connaissances d'algorithmique dans le système TIRPA
Deux types de connaissances sont manifestées par l'élève pendant qu'il résout un problème dans le système TIRPA. Il y a d'abord les apprentissages liés au *quand* et au *comment* de l'utilisation des différentes structures de contrôle. Il s'agit essentiellement de choisir la structure qui convient, compte tenu du but poursuivi, puis de formuler correctement cette structure. Pour les fins du modèle de l'élève, une connaissance spécifique est associée à chaque structure de contrôle. Elle correspond davantage à la capacité d'appliquer la bonne structure au bon moment (plutôt qu'à celle de l'appliquer selon la bonne syntaxe, parce que l'interface du système laisse peu de place à des erreurs au niveau de la formulation de structures).

L'autre type de connaissances a trait au *niveau de complexité des problèmes* et est associé à la capacité de formuler des stratégies de résolution correctes ainsi qu'à la capacité d'abstraction manifestée par le fait de *nommer des actions complexes* durant le processus de décomposition d'un problème. À cet égard, il faut mentionner que dans le système développé, l'élève formule sa solution dans un bloc de résolution, un médium dont il détermine lui-même la portée depuis l'instruction élémentaire jusqu'à la résolution de l'ensemble de la tâche dans un seul bloc.

De la codification à la planification

L'idée de base du Model Tracing consiste à utiliser le modèle de performance pour tracer l'état de la solution d'un élève à l'intérieur d'une tâche et à utiliser le modèle d'apprentissage pour tracer l'état des connaissances d'un élève à mesure qu'il accomplit les différentes tâches. Ces tâches sont choisies par le tuteur en fonction des connaissances diagnostiquées comme faibles ou absentes dans les connaissances de l'élève. Pendant qu'un élève résout un problème, des commentaires explicatifs sont générés et permettent une interprétation correcte de la solution. C'est essentiellement grâce à ce scénario d'apprentissage qu'un élève est susceptible de franchir les étapes du processus de compilation du modèle d'Anderson.

Dans ce modèle d'Anderson, tel que reflété dans le LISP-TUTOR, l'élève interagit avec le

système strictement au niveau des codes de programmation. L'élève doit bien sûr utiliser une certaine stratégie pour résoudre son problème, mais celle-ci ne se manifeste qu'à travers l'organisation des codes de programmation. Il n'est pas question, dans un tel système, d'exprimer explicitement une stratégie ou de procéder à une décomposition fonctionnelle du problème. Dans le LISP-TUTOR, il n'y a pas de place pour des solutions d'élèves qui ne font pas partie du modèle du système; cette contrainte est acceptable quand le domaine d'apprentissage est bien délimité comme dans la programmation, mais devient un sérieux handicap quand le domaine est relativement ouvert.

En cherchant à appliquer la méthodologie du Model Tracing au domaine de l'algorithmique dans lequel les connaissances à faire acquérir incluent des habiletés au niveau de la planification et non pas seulement au niveau de la codification, plusieurs problèmes surgissent notamment en relation avec la représentation des connaissances à acquérir et les difficultés d'analyse des solutions formulées par l'élève.

Représentation des connaissancesEX "Représentation des connaissances"§ à acquérir dans TIRPA

Dans les systèmes d'Anderson, chaque règle de production représentant les connaissances du domaine correspond à une étape dans la solution du problème (Anderson, Boyle et Reiser, 1985). En comparaison, les règles de génération de TIRPA représentent-elles vraiment les connaissances que le système veut faire acquérir aux élèves? Il est attendu qu'un élève maîtrise l'utilisation des structures de contrôle après avoir complété avec succès une séance d'apprentissage. Cette maîtrise implique notamment l'utilisation de structures imbriquées à plusieurs niveaux. Or aucune règle individuelle de génération ne représente une telle habileté (en particulier à cause de l'élaboration de l'algorithme par raffinements successifs). Dans les règles de génération, cette habileté est plutôt représentée par un ensemble de règles, la structure enveloppante étant du niveau d'abstraction le plus élevé.

La correspondance entre les règles de production et les connaissances à maîtriser n'est donc pas évidente, du moins à première vue. En effet, on compte plus de 150 règles dans le système TIRPA alors que les connaissances à maîtriser dépassent à peine une demi-douzaine. À cet égard, il faut rappeler (1) que certaines connaissances plus complexes sont représentées par un groupe de règles et non pas par une seule règle et (2) que c'est à une unité de connaissances (l'ensemble des règles partageant un même but) et non pas à chacune des règles individuelles qu'il faut associer une connaissance à maîtriser. En tenant compte de ce qui précède, on peut conclure que les règles de production représentent essentiellement les connaissances que l'élève doit acquérir dans le système TIRPA.

Puissance de l'analyseurEX "Analyse syntaxique"§

Dans la version originale du Model Tracing, le tuteur intervient *immédiatement* après l'identification d'une erreur. Cette contrainte est un peu adoucie dans la théorie PUPS (plus récente), mais demeure sérieuse quand il s'agit de permettre à un élève d'explorer librement différentes pistes de solution. Une intervention rapide du tuteur, comme c'est le cas dans le LISP-TUTOR, a deux effets sur le plan de l'analyse à effectuer. D'abord, l'analyse n'est faite qu'en fonction d'une seule ou d'un nombre très réduit de règles et ensuite, le nombre d'erreurs peut difficilement dépasser un puisque le tuteur intervient dès qu'une erreur est détectée.

Dans le système TIRPA, le niveau d'intervention diagnostique est le bloc de résolution. La liberté de manoeuvre exigée par le domaine d'apprentissage, liberté qui doit permettre à l'élève d'explorer librement différentes pistes de solution sans crainte d'être interrompu par le tuteur, implique que l'analyseur peut retracer simultanément plusieurs erreurs potentielles. Par conséquent, l'analyse diagnostique est plus complexe, notamment parce que le bloc formulé par l'élève peut contenir plusieurs différences (erreurs et/ou équivalences) avec la solution générée par le système pour atteindre le but poursuivi dans ce bloc. Il en résulte des difficultés particulières et la nécessité de recourir à des heuristiques pour limiter l'espace-problème.

Analyse diagnostique et Model Tracing
L'élève qui résout un problème comme *déplacer une balise de 2 pas vers l'est* peut obtenir un verdict VERT dans le premier bloc de résolution notamment en formulant l'un ou l'autre des ensembles suivants d'actions et de structures:

Ramasser une balise \ Aller déposer une balise.
Ramasser une balise \ Répéter 2 fois (avancer d'un pas) \ Déposer une balise.
Ramasser une balise \ Avancer d'un pas \ Avancer d'un pas \ Déposer une balise.

Comment interpréter ces différents comportements *corrects* pour les fins de mise à jour du modèle d'apprentissage? Et plus spécifiquement, un élève manifeste-il une connaissance particulière

du fait même de décomposer sans égard à la complexité du problème?
ou du fait d'utiliser une structure de contrôle sans égard au nombre d'itérations?
ou encore du fait de formuler une solution plus générale que nécessaire, utilisant ainsi des actions ou structures particulières?

La réponse à ces questions est bien sûr qu'il ne faut pas simplement se fier aux règles déclenchées pour les fins de la mise à jour du modèle de l'élève, même si chacune de ces règles manifeste un certain comportement de l'élève. Dans la présente version du système TIRPA pourtant, un élève se trouve parfois avantagé, en termes de connaissances réputées maîtrisées, par le fait qu'il utilise un chemin plus long pour atteindre un but donné, et cela simplement parce que ce chemin occasionne le déclenchement d'un plus grand nombre de règles. C'est le prix à payer pour la souplesse et la puissance de l'analyse incorporées dans ce premier prototype.

Conclusion

Le Model Tracing est un élément clé de la théorie ACT*. Dans les systèmes où il a été appliqué par Anderson, notamment dans le LISP TUTOR et le GEOMETRY TUTOR, il constitue véritablement une méthodologie efficace sur laquelle repose l'analyse des comportements des élèves et les rétroactions intelligentes du tuteur. Mais le Model Tracing (du moins dans sa version originale) comporte une limite importante: il oblige l'élève à suivre le modèle du système de très près. Cela est possible avec un système relativement directif et une structure assez rigide du domaine d'apprentissage. Avec l'algorithmique comme domaine d'apprentissage et la nécessité pour l'élève d'explorer différentes pistes de solution, le Model Tracing est beaucoup plus difficile à appliquer.

Références

Anderson, J. R. (1983) *The Architecture of Cognition.* Harvard University Press, Cambridge, Mass.

Anderson, J.R., Boyle, C.F. and Reiser, B.J. (1985) Intelligent tutoring systems. *Science,* 228, 4698, pp. 456-462.

Anderson, J.R. (1986) *Cognitive Modelling and Intelligent Tutoring.* National Science Foundation, Washington, D.C.

Anderson, J.R. (1987) Production systems, learning, and tutoring. In Klahr, Langley and Neches, (ed.): *Production System Models of Learning and Development,*pp. 437-457. MIT Press, Cambridge, Mass.