ENSEIGNER L'ALGORITHME POUR QUOI ? QUELLES NOUVELLES QUESTIONS POUR LES MATHEMATIQUES ? QUELS APPORTS POUR L'APPRENTISSAGE DE LA PREUVE ?

Simon **MODESTE**

Université de Montpellier, I3M, équipe Didactique et Épistémologie des Mathématiques

simon.modeste@univ-montp2.fr

Résumé

Récemment introduit dans les curriculums du lycée français, l'algorithme est un concept fortement lié à l'informatique, aux mathématiques et à la preuve et qui soulève de nombreuses questions didactiques. Notre travail de thèse (Modeste, 2012) propose une analyse épistémologique du concept d'algorithme dans le but d'étudier sa transposition didactique et de construire des situations didactiques.

Dans ce texte, nous présenterons tout d'abord une analyse épistémologique détaillée du concept en mettant en avant ses aspects fondamentaux. Cette étude nous servira d'appui pour proposer un modèle de conceptions (Balacheff & Margolinas, 2005; Vergnaud, 1990) pour l'algorithme du point de vue du savoir savant, prenant en compte l'ensemble des formes de l'algorithme et les aspects outils et objet (Douady, 1986) du concept. Nous montrerons alors dans quelle mesure ces résultats, validés expérimentalement par les analyses d'entretiens avec des chercheurs, ont permis de mener une étude de la transposition en jeu dans l'enseignement au lycée en France. À travers l'étude des instructions officielles, de manuels scolaires et de ressources en ligne, nous mettrons en évidence une transposition partielle du concept principalement orientée vers la programmation et l'usage de l'algorithme comme un outil. Enfin nous proposerons une caractérisation des problèmes fondamentaux pour l'algorithme et des perspectives pour la construction et l'étude de situations didactiques en algorithmique.

Mots clés

Algorithme, algorithmique, épistémologie, didactique, lycée, transposition didactique

Remerciements

L'auteur souhaite remercier Sylvain Gravier et Cécile Ouvrier-Buffet, encadrants du travail de thèse présenté ici.

CONTEXTE ET PROBLEMATIQUE

Le travail présenté dans ce texte est issu d'un travail de thèse soutenu en décembre 2012 à l'Université de Grenoble ayant pour titre « Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ? ». Nous n'allons pas présenter ici l'intégralité du travail. Nous nous focaliserons sur certains aspects en les replaçant dans le contexte du travail mené. Pour les détails du travail, nous renvoyons au manuscrit de thèse (Modeste, 2012).

Contexte

On peut situer les motivations de ce travail de recherche à trois niveaux. Au niveau épistémologique, il est motivé par la volonté de prise en compte du développement de l'informatique et l'influence que ce développement peut avoir sur les mathématiques et leur pratique. Au niveau du contexte de l'enseignement des mathématiques, on peut constater l'introduction de nombreux éléments d'informatique dans l'enseignement secondaire et plus particulièrement d'éléments d'algorithmique (dans ou à côté du curriculum de mathématiques) ainsi que l'usage courant des TICE, qui peuvent amener des questions d'informatique dans la classe de mathématiques. Au niveau français, enfin, l'introduction en 2009 d'algorithmique dans les curriculums de mathématiques du lycée et la création en 2012 d'un enseignement de spécialité « Informatique et Sciences du Numérique » (ISN) en Terminale Scientifique poussent à s'intéresser aux liens entre mathématiques et informatique. Notre travail s'axe plus particulièrement sur un point central de cette interaction, le concept d'algorithme et l'algorithmique.

Problématique

Ce travail se place dans une perspective classique en didactique des mathématiques (Artigue, 1990) : il propose un travail épistémologique sur le concept d'algorithme et l'activité algorithmique, qui permet de soutenir une étude didactique, celle de la transposition didactique et le développement de situations d'apprentissage.
Les questions abordées dans ce travail de thèse peuvent être regroupées autour de quatre pôles, relativement à leur thématique et aux méthodes proposées pour y répondre.

Épistémologie

- Quels sont les éléments constitutifs spécifiques du concept d'algorithme ? Quels sont l'objet et les préoccupations fondamentales de la discipline algorithmique ? Quels liens entretiennent algorithme et preuve ?
- Peut-on parler d'activité ou de pensée algorithmique dans les mathématiques ? Dans l'informatique ? Quelles perspectives cela peut-il ouvrir pour une approche didactique ?
- Peut-on construire un modèle de l'activité algorithmique permettant de questionner la transposition didactique ?

Pour répondre à ces questions nous nous sommes appuyés sur l'analyse du texte et des discours du savoir savant. Nous présenterons seulement le travail autour de la première et de la troisième question.

Conception de chercheurs

- L'analyse épistémologique développée s'accorde-t-elle avec la conception des chercheurs sur l'algorithme ? Quels apports à notre questionnement peuvent fournir les points de vue des chercheurs ?
- Peut-on repérer des variations dans les conceptions des chercheurs en fonction de leur discipline ?

Pour étudier ces questions nous avons conçu et mis en œuvre des entretiens avec vingt-deux chercheurs en mathématiques et informatique et analysé leur discours. Cette partie du travail, qui a servi à la fois de validation du modèle épistémologique présenté et de ressource pour mieux comprendre la place de l'algorithme dans les mathématiques et l'informatique ne sera pas développée ici.

Transposition didactique

- Quel rôle et quelle place sont attribués à l'algorithmique dans les programmes de mathématiques et d'ISN au lycée ? Quelles transpositions ont été effectuées ?
- Qu'en est-il des manuels de mathématiques du lycée ? Quels types de questions et de problèmes sont mis en jeu en algorithmique ?
- Quels types de ressources en ligne sont mises à disposition des enseignants en algorithmique ?

Pour répondre à ces questions, nous avons réalisé une analyse des programmes, de deux collections de manuels et des ressources en ligne des IREM, à l'aide des outils épistémologiques développés.

Situations d'apprentissage pour l'algorithme

- Comment caractériser les problèmes mathématiques à fort potentiel pour l'algorithmique ? Quels problèmes répondant à ces critères peut-on proposer ?
- Quels éléments épistémologiques sont à prendre en compte pour construire des situations didactiques autour de l'algorithme ? Quelles situations didactiques peut-on proposer ?

Tenter de répondre à ces questions nous a conduits à formuler des propositions de problèmes et de situations, leur analyse a priori à l'aide du modèle épistémologique développé et l'expérimentation d'une de ces situations.

Plan de l'article

Dans un premier temps nous introduirons quelques exemples d'algorithmes à même d'illustrer nos propos. Nous présenterons, dans une seconde partie, le travail épistémologique développé. Une troisième partie traitera des résultats de l'étude de la transposition didactique de l'algorithme au lycée français, pour laquelle nous donnerons quelques illustrations. Enfin, nous présenterons des éléments relatifs à la conception de situations en algorithmique et la situation expérimentée.

EXEMPLES ET DEFINITIONS PRELIMINAIRES

Nous commencerons par donner ici quelques définitions utiles pour la suite de l'article (en particulier pour les termes *algorithme* et *algorithmique*) et par étudier quelques exemples d'algorithmes choisis afin de mettre en avant et d'illustrer les éléments que nous formaliserons dans le travail de conception d'un modèle épistémologique.

Algorithme, définition et exemples

Définition

Sur la base de la littérature du savoir savant, notamment (Aho, Hopcroft, & Ullman, 1989; Beauquier, Berstel, & Chrétienne, 1992; Bouvier, George, & Le Lionnais, 2005; Chabert, 2010; Cormen, Leiserson, Rivest, & Cazin, 1994; Garey & Johnson, 1979; Donald E. Knuth, 1973; Donald Ervin Knuth, 1996; Lovász & Plummer, 2009; Sedgewick, 1988), nous avons retenu la définition suivante, synthèse des différentes définitions que nous avons pu rencontrer :

Un **algorithme** est une procédure de résolution de problème, s'appliquant à une famille d'instances du problème et produisant, en un nombre fini d'étapes constructives, non-ambiguës et organisées, la réponse au problème pour toute instance de cette famille.

Un exemple pour introduire et illustrer nos propos : cherchez la frontière

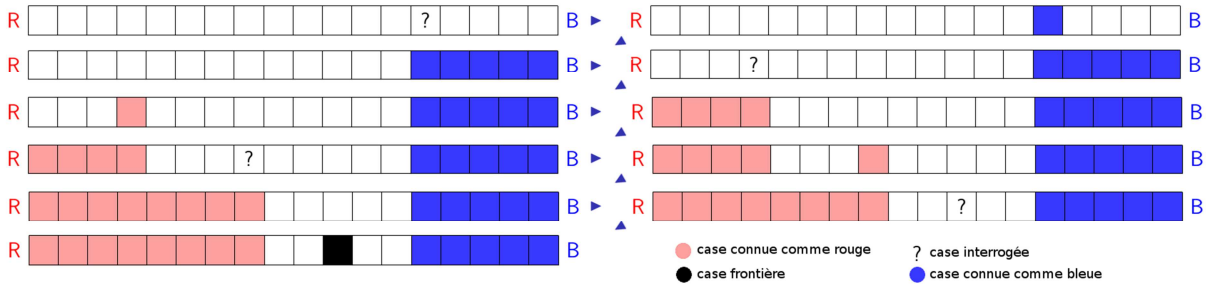
Nous présentons et étudions brièvement un premier exemple pour illustrer et soutenir le travail des parties à suivre.
Ce problème est inspiré d'un problème proposé par Giroud (2011) dans son travail de thèse sur la démarche expérimentale en mathématiques. Nous l'avons simplifié par commodité.
On se donne une bande constituée d'un nombre fini de cases consécutives, que nous appellerons territoire, comme dans la figure 1. Parmi ces cases, l'une est de couleur noire et est appelée la frontière. Toutes les cases situées à droite de la frontière sont bleues, toutes celles situées à gauche sont rouges. Initialement, on ne connaît pas la couleur des cases. On peut interroger les cases, une à une, pour connaître leurs couleurs. Le but est de retrouver la case frontière.

Figure 1 : un territoire de 18 cases

Lorsque l'on interroge une case, il y a trois possibilités :

- elle est noire, on a trouvé la frontière,
- elle est rouge, la frontière se situe à droite de la case interrogée,
- elle est bleue, la frontière se situe à gauche de la case interrogée.

Dans les deux derniers cas, on se retrouve à chercher la frontière dans un nouveau territoire, plus petit, constitué des cases tout à droite (ou tout à gauche) de la case interrogée.
En interrogeant ainsi une nouvelle case, on réduit le domaine de recherche et l'on finit par



trouver la frontière. La figure 2 donne un exemple d'une recherche d'une frontière.

Figure 2 : exemple de recherche d'une frontière, trouvée en 4 coups.

Si l'on cherche comment retrouver la frontière en interrogeant le moins de cases possible (quelle que soit la taille du territoire et l'emplacement de la frontière), on doit toujours interroger la case (ou l'une des deux cases) qui divise(nt) le territoire en deux territoires de tailles équilibrées. C'est un algorithme de recherche par dichotomie.
Cet algorithme donne une méthode systématique pour retrouver la frontière de manière effective, quelle que soit la taille du territoire et la position de la frontière.
Nous n'entrons pas ici dans le détail de ce que signifie précisément « en interrogeant le moins de cases possible ». Nous précisons simplement qu'il s'agirait ici de trouver un algorithme optimal pour la complexité dans le pire des cas, c'est-à-dire un algorithme qui minimise c(n) où c(n) est le nombre maximum d'interrogations pour trouver la frontière dans un territoire de taille n (pris pour toutes les positions de la frontière). Cela met en jeu la notion de complexité d'un algorithme, c'est-à-dire l'évaluation de son efficacité à traiter les différentes instances du problème qu'il résout. Pour plus de détails sur la notion de complexité, nous renvoyons par exemple à Beauquier et al. (1992) ou Cormen et al. (1994). Trouver un algorithme optimal

pour un problème donné est un enjeu important. Connaître la complexité d'un algorithme optimal résolvant un problème donné permet d'associer cette complexité au problème. Cela permet de classifier les problèmes selon des classes de complexité. C'est le cas ici : la dichotomie étant l'algorithme optimal de recherche de la frontière et nécessitant un nombre d'interrogations logarithmique en fonction du nombre de cases, on peut dire que le problème de recherche de la frontière est logarithmique (cela se démontre, nous ne détaillerons pas ici).

Cherchez la frontière (variante)

Intéressons-nous maintenant à une variante du problème de recherche de la frontière, en gardant les mêmes règles mais en ajoutant la règle suivante : il est interdit d'interroger plus d'une fois une case bleue. On rencontre alors 3 cas lorsqu'on interroge une case :

- elle est noire, on a trouvé la frontière,
- elle est rouge, la frontière se situe à droite de la case interrogée, et l'on cherche à résoudre le même problème sur le territoire plus petit situé à droite de la case interrogée,
- elle est bleue, on peut se restreindre à une recherche sur le territoire situé à gauche de la case interrogée. Mais nous sommes confrontés à résoudre un problème différent : ne pouvant plus interroger de case bleue, nous sommes contraints d'interroger à chaque fois la case la plus à gauche du territoire jusqu'à rencontrer la frontière.

Le problème est alors bien différent et l'algorithme optimal pour cette variante (pour la complexité dans le pire des cas) n'est pas celui qui interroge la case médiane du territoire.
On voit ici comment le problème étudié et les problèmes qui apparaissent lors de sa résolution influencent fortement la construction de l'algorithme de résolution.

Algorithmique

Définition

De la même façon que pour l'algorithme, nous avons retenu la définition suivante :

> L'**algorithmique** est l'activité ou le domaine qui a pour objet l'étude des algorithmes, c'est-à-dire, leur conception, leur description, leur analyse et l'étude des problèmes qu'ils permettent de résoudre.

Un exemple de résultat algorithmique négatif : le dixième problème de Hilbert

Lors du congrès international des mathématiciens de 1900 à Paris, David Hilbert énonce ses célèbres 23 problèmes. Le dixième concerne la recherche générale des solutions entières des équations diophantiennes à plusieurs inconnues, que l'on peut formuler ainsi :

- Entrée : Un entier n et un polynôme P à coefficients dans $\boldsymbol{Z}$.
- Question : L'équation en $X_1, \dots, X_n$, $P(X_1, \dots, X_n)=0$ a-t-elle une solution dans $\boldsymbol{Z}^n$?

En 1970, Matiiassevitch démontre qu'il n'existe pas d'algorithme permettant de résoudre ce problème, c'est-à-dire d'algorithme qui répondrait à la question pour tous entiers n et polynôme P. Ce résultat et bien d'autres résultats de non-existence d'algorithmes, s'appuient sur la construction de modèles théoriques de ce qu'est un algorithme et de ce qui est algorithmiquement résoluble. C'est un champ complet des mathématiques qui s'est ouvert, qui a produit des résultats parmi les plus importants du XX^{e} siècle et dont le célèbre problème « P=NP » fait partie des problèmes du millénaire de l'institut Clay.

ÉPISTEMOLOGIE DE L'ALGORITHME

Nos recherches nous ont d'abord amené à faire un premier travail épistémologique sans faire

appel à une théorie particulière, en étudiant le texte du savoir savant (notamment les ressources citées précédemment). Nous avons décliné ce premier modèle en cinq aspects que nous avons appelés fondamentaux. Nous avons ensuite conçu un second modèle épistémologique plus avancé, permettant de palier à certaines difficultés qui apparaissaient dans le premier modèle. Pour que ce second modèle ait du sens, il est nécessaire de donner quelques détails relativement à ce travail préliminaire.

Cinq aspects fondamentaux du concept d'algorithme

Un premier travail a donc consisté à repérer les aspects et les spécificités du concept d'algorithme. Nous avons ensuite regroupé ces aspects autour de cinq aspects fondamentaux, qui font écho à des points soulevés dans les exemples précédents. Ce travail s'est appuyé sur l'étude d'un corpus de textes académiques (cités précédemment) et des interviews menées auprès de chercheurs en mathématiques et informatique.

Effectivité

L'effectivité réfère au fait qu'un algorithme apporte une réponse constructive et qui peut être effectivement mise en œuvre (en un nombre fini d'étapes, notamment) pour résoudre chaque instance du problème, par un opérateur (qui peut être humain ou machine). Il est nécessaire que l'ensemble des instructions à réaliser ne contienne pas d'ambiguïté pour l'opérateur afin que l'exécution de l'algorithme sur une instance donnée du problème ne dépende pas de l'opérateur mais seulement des instructions prescrites.

Résolution de problème

Cela réfère à un point important, mis en avant précédemment, qui est le fait qu'un algorithme soit une réponse à un problème et que l'algorithme soit une résolution qui permet de traiter toute instance du problème. Ces instances sont spécifiées en entrée de l'algorithme et la sortie de l'algorithme associée est la solution spécifique à l'instance du problème. L'exemple précédent met très bien en avant l'importance de cette notion de problème et le fait qu'un algorithme n'a de sens qu'associé au problème qu'il résout.

Preuve

Algorithme et preuve ont un lien extrêmement étroit. À la construction d'un algorithme s'associe une preuve de cet algorithme, c'est-à-dire la preuve qu'il résout bien le problème pour toute instance et la preuve qu'il le fait en un nombre fini d'étapes à chaque fois. Un lien peut être aussi fait entre preuves constructives (dont les preuves par récurrence) et algorithmes car de chaque preuve de ce type on peut extraire un algorithme sous-jacent, permettant de construire effectivement les solutions au problème. Les questions mathématiques relatives aux modèles théoriques (que nous évoquons juste après) mettent aussi en jeu des liens étroits entre algorithme, preuve et logique.

Complexité

La complexité est l'étude du comportement du nombre d'étapes ou de l'espace nécessaire à la mise en œuvre d'un algorithme donné en fonction de la taille de l'instance étudiée. Cet aspect regroupe tout ce qui a trait à cette notion, comme les différents types de complexité (en temps en espace, au pire, en moyenne...), la notion de complexité d'un problème, et toutes les classes de complexité qui permettent de classifier algorithmes et problèmes.
Cette notion de complexité est complètement spécifique à l'algorithmique. Selon certains auteurs comme Knuth (1985), cette préoccupation de l'efficacité des processus de résolution

proposés pour un problème est même un des points essentiels qui distingue la pensée mathématique de la pensée algorithmique.

Modèles théoriques

Cet aspect fait référence à l'ensemble des éléments relatifs à la description et à l'étude de ce qui est algorithmiquement résoluble. Ces questions ont structuré et structurent encore un pan de la recherche en informatique et mathématiques et constituent un aspect essentiel du concept d'algorithme et de l'activité algorithmique.

Intérêt de cette classification : une dialectique outil-objet pour l'algorithme

Ces cinq aspects permettent de décrire une dialectique outil-objet pour l'algorithme, au sens de Douady (1986). Les aspects *effectivité* et *résolution de problème* relèvent de l'algorithme outil, les aspects *preuve*, *complexité* et *modèles théoriques* relèvent de l'algorithme objet. Cette distinction des différents aspects et la dialectique outil-objet a permis de mener de premiers travaux d'analyse des curriculums et de manuels (Modeste, 2009).

Limites de ce modèle et motivation pour un modèle plus structuré

Ce modèle « naïf » s'est avéré efficace pour étudier un discours sur l'algorithme mais beaucoup moins pour analyser l'activité d'un sujet ou des activités proposées dans des ressources. Il met aussi sur un même plan, dans la dialectique outil-objet, des éléments que nous ne considérons pas relever du même niveau de description. Cela apparaît en particulier lorsqu'on regarde l'aspect preuve. Alors qu'on peut faire usage d'un algorithme comme d'un outil de preuve, on classifie cet usage plutôt du côté objet car cela demande une prise de recul sur l'objet algorithme. De même, on peut ne pas souhaiter mettre sur le même plan la preuve algorithmique, qui est une forme de preuve et la preuve d'un algorithme, dont l'objet même est l'algorithme.
Pour dépasser ces difficultés, nous avons proposé un second modèle, plus structuré et faisant appel à un modèle théorique de didactique des mathématiques permettant de modéliser le savoir : la notion de conception du modèle ck¢.

Construction d'un modèle de µ-conceptions pour l'algorithme

Afin de présenter ce modèle, revenons sur les outils théoriques choisis pour le construire et introduisons chacun des éléments pris en compte. Nous présenterons d'abord brièvement le modèle ck¢ que nous avons retenu et sa structure en Problèmes - Opérateurs - Système de représentation - Structures de contrôle. Nous préciserons ensuite des éléments liés au concept d'algorithme utilisés pour l'adapter à notre situation et prendre en compte le fait que l'on veut référer ici au savoir savant pour construire une référence épistémologique. Ceci donnera lieu à une définition spécifique de la notion de problème et une classification des différentes formes sous lesquelles un algorithme peut être décrit et manipulé. Enfin nous montrerons comment le modèle ck¢ et nos choix peuvent rendre compte de la dialectique outil-objet du concept d'algorithme et apporter des clarifications concernant la question de la relation preuve-algorithme soulevée au paragraphe précédent.

Conceptions et µ-conceptions dans le modèle ck¢

La notion de conception du modèle ck¢ (Balacheff & Margolinas, 2005) se base sur la notion de concept (Vergnaud, 1990) qui décrit un concept comme un triplet (**S**,I,*S*), où :

- **S** est l'ensemble des situations qui donnent du sens au concept,
- I est un ensemble d'invariants opératoires du concept, et

- *S* l'ensemble des représentations du concept.

La notion de conception de ck¢ reprend cette description en distinguant deux types d'invariants opératoires : des opérateurs qui décrivent l'action dans la situation et des structures de contrôle, qui structurent la relation entre opérateur et situation. Plus précisément, une conception C est décrite comme un quadruplet (P,R,L,Σ) avec :

- P un ensemble des problèmes donnant du sens au concept,
- R un ensemble d'opérateurs agissant sur les problèmes,
- L un système de représentation qui permet d'exprimer les éléments de P et de R, et
- Σ un ensemble de structures de contrôle qui décrivent la relation P-R.

Balacheff et Margolinas précisent :

> Nous appelons opérateur ce qui permet la transformation des problèmes ; ces opérateurs sont attestés par des productions et des comportements. Un système de représentation (langagier ou non) permet l'expression des problèmes et des opérateurs. Enfin, une structure de contrôle assure la non contradiction de la conception et contient au moins sous la forme d'oracles les outils de décision sur la légitimité de l'emploi d'un opérateur ou sur l'état (résolu ou non) d'un problème. (Balacheff & Margolinas, 2005, p. 80)

Nous nous intéressons ici à la description d'un modèle épistémologique autour du concept d'algorithme, en nous appuyant sur le savoir-savant. Nous voulons décrire ce qui est appelé une µ-conception dans le modèle ck¢, c'est-à-dire la conception « dérivée du corpus des savoirs consensuels (académiques) en mathématiques » (et en informatique dans notre cas) (Balacheff & Margolinas, 2005, p. 98).

Une notion de problème adaptée

Alors que les problèmes décrits dans une conception sont attestés par les déséquilibres du système sujet-milieu, il nous faut ici les décrire autrement dans une µ-conception. Nous choisissons, spécifiquement au cadre de l'étude du concept d'algorithme de prendre une définition de problème adaptée de la théorie de la complexité algorithmique, et donc proche de celle dont les problèmes sont attestés dans le savoir académique.

Nous décrirons un problème comme une couple (I,Q) où :

- I est l'ensemble des instances du problème,
- Q est une question générale qui peut se poser pour chaque instance du problème.

On peut par exemple décrire le problème de recherche du pgcd sous cette forme avec I l'ensemble des couples d'entiers non-nuls (c'est-à-dire $\mathbf{N}^{*2}$) et la question Q « Quel est le pgcd du couple d'entiers ? ».

Résoudre le problème algorithmiquement, c'est fournir un algorithme qui donne la réponse à la question Q quelle que soit l'instance proposée. Prouver un algorithme, c'est prouver qu'il donne toujours (c'est-à-dire après un nombre fini d'étapes) une réponse correcte pour toute instance du problème. On appelle *preuve de terminaison* la preuve du fait que l'algorithme donne une réponse après un nombre fini d'étapes pour toute instance et *preuve de correction* la preuve de la validité des réponses de l'algorithme pour toute instance.

Une µ-conception pour l'algorithme

Sur la base de ce que nous venons de décrire, nous pouvons décrire une µ-conception pour l'algorithme ainsi :

- des problèmes donnés sous la forme Instance-Question,
- des opérateurs qui sont les algorithmes permettant de résoudre ces problèmes,
- des structures de contrôles qui relèvent de la preuve de ces algorithmes (terminaison et correction).

En ce qui concerne la description des systèmes de représentation (Σ), nous proposons, sur la base de notre étude épistémologique, trois systèmes de représentation pour décrire les algorithmes. Il s'agit en fait de trois grands groupes de types de représentations :

- La preuve algorithmique (PA) : il s'agit de la description d'un algorithme au travers d'une preuve (par exemple une preuve par récurrence). Dans ce cas, l'algorithme et sa preuve ne sont pas séparés.
- L'algorithme mathématique (AM) : c'est la description d'un algorithme dans un langage mathématique mettant en jeu certains éléments issus de la programmation. La preuve de l'algorithme peut alors être décrite séparément de l'algorithme lui-même.
- L'algorithme informatique (AI) : l'algorithme est décrit dans un langage de programmation. Sont alors distingués le texte de l'algorithme, sa sémantique, et sa validation.

Dialectique outil-objet et place de la preuve

La description précédente ne prend en compte qu'une partie de la µ-conception algorithme. En effet, elle ne considère que les problèmes qui peuvent être (ou non) résolus par un algorithme, c'est-à-dire pour lesquels les algorithmes sont les opérateurs. L'algorithme y est alors outil.
À ces problèmes il faut ajouter ceux portant sur les algorithmes ou questionnant les algorithmes : problèmes de complexité, de validité, de propriétés, etc. L'algorithme est alors objet.
Intégrer ces problèmes permet ainsi de décrire d'une autre manière la dialectique outil-objet pour l'algorithme avec ces deux familles de problèmes.
Ce choix permet aussi de préciser plusieurs niveaux de relation entre algorithme et preuve, d'une part au travers du système de représentation PA et d'autre part dans les deux familles de µ-conceptions outil et objet où la preuve relèvera de Σ dans le cas de l'outil et de R dans le cas de l'objet. Ces précisions permettent de clarifier les questions que soulevait notre premier modèle.

Un modèle en six µ-conceptions

Le modèle que nous proposons intègre donc six µ-conceptions reparties selon les trois systèmes de représentation présentés précédemment et selon les deux familles de problèmes décrivant la dialectique outil-objet. La figure 3, page suivante, donne quelques détails sur ces µ-conceptions. C'est à partir de ce modèle que nous avons pu produire l'analyse de la transposition didactique dont nous présentons les résultats dans la section suivante.

TRANSPOSITION DIDACTIQUE AU LYCEE EN FRANCE

Le modèle précédent nous a permis de mener une analyse fine de la transposition didactique de l'algorithme au lycée en France. Nous avons pu étudier de manière effective différents types de données : instructions officielles (programmes et documents ressources), manuels, ressources pour la classe et la formation des enseignants. Nous n'entrerons pas ici dans les détails de la mise en œuvre du modèle épistémologique pour l'analyse mais nous nous restreindrons à présenter les résultats de cette analyse, assortis d'exemples illustratifs.

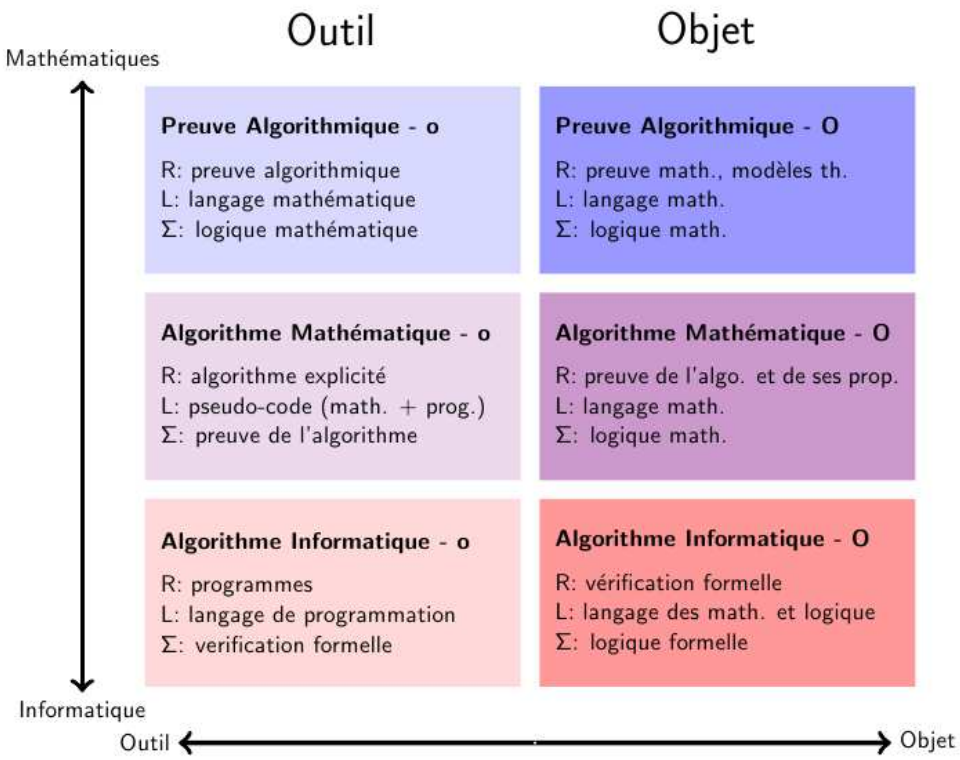


Figure 3 : Résumé des éléments constitutifs des μ-conceptions de notre modèle. Les problèmes (P) relèvent de l'outil ou de l'objet, chaque conception décline ensuite opérateurs (R), système de représentation (L) et structures de contrôle (Σ).

Corpus de documents analysés pour saisir la transposition didactique

Nous avons étudié la transposition du concept d'algorithme au lycée à travers différents documents. L'algorithme coexistant dans la classe de mathématiques et la classe d'ISN, nous avons étudié les programmes des deux enseignements, en étudiant tous les niveaux du lycée général en ce qui concerne les mathématiques. La mise en place de l'algorithmique dans les curriculums de mathématiques a commencé en 2009 avec la classe de seconde et un document d'accompagnement a été produit. Nous faisons l'hypothèse que ce document vise à préciser les attentes curriculaires et constitue dans ce sens un vecteur important dans le processus de transposition, c'est pourquoi il fait partie de notre corpus. Les manuels sont un élément souvent très riche pour étudier la transposition didactique en France, ils permettent notamment de voir comment les instructions officielles se déclinent en tâches et nous informent sur le savoir enseigné. Nous avons étudié des collections complètes de manuels pour le lycée général : les collections Transmaths (Nathan), Math'x (Didier) et Indice (Bordas). Nous avons aussi analysé un autre type de ressources, disponible très rapidement après l'introduction de l'algorithmique au lycée : les ressources mises à disposition en ligne par les IREM. Il nous a semblé important d'inclure des ressources en ligne, car elles font pleinement partie des ressources de l'enseignant. Certains des groupes IREM ayant produit ces ressources ont aussi participé à la formation continue des enseignants à l'algorithmique, leurs productions sont donc un indicateur utile pour mesurer la transposition didactique dans le lycée français.

Utilisation du modèle épistémologique pour l'analyse : deux exemples détaillés

Nos analyses de la transposition didactique s'appuient sur le modèle épistémologique de conceptions présenté. Nous illustrons son utilisation avant de présenter les résultats obtenus.

Nous choisissons pour cela l'exemple du traitement de la dichotomie dans le document ressource pour la classe de seconde[1] et dans une ressource de l'IREM de Lyon[2].

Dichotomie dans le document ressource de mathématiques de seconde

Dans ce document, dont l'extrait concerné est en annexe, la dichotomie est présentée dans le cadre de la recherche d'un zéro d'une fonction. Une première activité, le jeu du nombre à deviner, est proposée pour introduire la dichotomie. Dans ce jeu, il s'agit de trouver un nombre compris entre 10 et 100, l'ordinateur répondant « c'est plus » ou « c'est moins » aux propositions du joueur. Un premier algorithme (en fait, un programme-papier, au sens ou nous le définissons plus loin) est proposé, il simule le joueur qui choisit le nombre. La programmation du jeu est proposée comme une première activité algorithmique. Ensuite, l'élève doit jouer contre le programme et essayer de gagner à tous les coups, c'est-à-dire en moins de 6 essais. À ce niveau, l'algorithme en jeu est la stratégie de dichotomie qui doit émerger du côté du joueur qui devine. On peut relever les éléments de la conception en jeu dans le tableau suivant :

P	Instances : le choix d'un nombre gagnant entre 10 et 100. Question : Trouver ce nombre en moins de 6 essais. (Il s'agit évidemment d'une reformulation dans les termes de notre modèle, de même que dans la suite.)
R	Les stratégies possibles du joueur qui cherche le nombre (l'élève) : une règle qui dit quel nombre tester selon les réponses précédentes du programme.
L	Cette stratégie est sûrement attendue en langage courant, étant donné qu'elle est formulée (et justifiée) de manière très informelle : Une bonne stratégie conduit à l'algorithme utilisant la dichotomie. Cette méthode consiste, en choisissant à chaque fois la valeur située au milieu de l'intervalle en cours, à réduire de moitié à chaque fois l'amplitude de l'intervalle dans lequel se trouve le nombre et comme 2^6 est égal à 64, le dernier intervalle, sur cet exemple, est d'amplitude 1.
Σ	La citation ci-dessus montre aussi la justification de la méthode. La réussite systématique face à l'ordinateur constitue un autre contrôle de l'algorithme. Notons que la preuve de l'optimalité n'est pas proposée, ni de manière générale, ni dans ce cas particulier d'un intervalle d'entiers de 10 à 100 car il faudrait aussi prouver que 5 coups ne suffisent pas toujours. On peut aussi remarquer que l'argument de validité de l'algorithme est très peu détaillé.

On reconnaît ici la conception AM-outil (conception outil de paradigme Algorithme Mathématique) avec une structure de contrôle (assez minimale) de l'ordre de la preuve mathématique. La conception AM-objet est-elle aussi présente ? Peu de questions sont posées sur l'algorithme lui-même, il est simplement précisé :

> Le choix des valeurs 10 et 100 qui encadrent le nombre à trouver, ainsi que le nombre d'essais est à mettre en débat dans la classe. En effet, selon le choix de ces valeurs, il sera ou non possible de déterminer à coup sûr la solution avec une bonne stratégie, ou on pourra seulement optimiser les chances de gagner.

Nous dirons que le potentiel de l'algorithme de dichotomie pour vivre en tant qu'objet est soulevé, mais qu'aucun opérateur n'est mis en jeu pour étudier cet algorithme. On ne peut pas vraiment considérer que la conception AM-objet soit présente. Relevons un autre point : le programme de jeu n'a pas de rôle algorithmique réel dans cette activité, étant donné que des élèves pourraient jouer l'un contre l'autre. Pourtant, programmer un adversaire virtuel semble aussi considéré dans le document comme une activité algorithmique (cela relève d'un amalgame algorithme-programme).

La seconde activité sur la dichotomie est la recherche d'un zéro d'une fonction par dichotomie. En recherchant les éléments de la conception en jeu, on obtient le tableau suivant :

1 Document *Ressources pour la classe de seconde – Algorithmique*, DGESCO, accessible à l'adresse : http://cache.media.eduscol.education.fr/file/Programmes/17/8/Doc_ress_algo_v25_109178.pdf

2 Document *Diviser pour* régner, IREM de Lyon, accessible à l'adresse : http://math.univ-lyon1.fr/irem/IMG/pdf/diviserpourregner_avec_solutions.pdf

P	Instance : une fonction *f* qui change de signe entre *a* et *b*. Question : résoudre *f(x)=0* (sic).
R	Les opérateurs sont les instructions définies par le *pseudo-code* (introduit plus ou moins implicitement tout au long du document ressource).
L	Il s'agit de ce *pseudo-code* : **Variables** m , valeur milieu de l'intervalle « courant » **Initialisation** a et b, les bornes de l'intervalle [a ; b] *f*, la fonction (rappel : *f* change de signe entre *a* et *b*) **Traitement** **Pour** i **variant de** 1 **à** 50 m **prend la valeur** (a+b)/2 **Si** *f*(m) **et** *f* (a) **sont de même signe alors** a **prend la valeur** m **sinon** b **prend la valeur** m **Sortie** **Affiche** a et b
Σ	La validité de la méthode n'est pas questionnée. D'ailleurs, il n'est même pas précisé que la fonction s'annule au moins une fois (parce qu'elle serait continue ou d'autres hypothèses). La structure de contrôle qui semble valider l'efficacité de la méthode pour trouver le zéro est l'activité précédente, faisant émerger la dichotomie : « On transpose cette méthode ici ». Notons que cette méthode n'a été validée comme optimale que pour une recherche parmi un intervalle de 91 entiers.

On est dans la conception AI-outil et la validation de l'algorithme est absente. La conception objet, qui pourrait apparaître ici, étant donné la richesse de la situation, n'est touchée qu'à travers une remarque sur la précision de la méthode en fonction du nombre d'itérations (c'est-à-dire par des considérations de méthodes numériques).

Une ressource de l'IREM de Lyon

Cette ressource est à destination des enseignants et formateurs à la recherche d'approfondissements en algorithmique. Elle introduit le principe algorithmique « diviser pour régner » au travers de plusieurs exemples. Concernant la dichotomie, elle propose la comparaison de deux algorithmes donnés pour le problème de recherche d'un élément dans une liste strictement croissante. Le document met en jeu les conceptions Outils et objet du paradigme AI, présentées et illustrées dans les deux tableaux suivants.

On peut noter tout d'abord la présence de la conception AI-outil :

	Conception	Extraits du document (illustration)
P	Instance : une liste triée strictement croissante et un élément *x*. Question : Quel est le rang de *x* dans la liste ?	On se propose de comparer les deux algorithmes suivants (langage Xcas) qui ont pour objectif de chercher le rang d'un élément dans une liste dont les éléments sont supposés triés dans l'ordre strictement croissant.

R	Algorithmes séquentiels et de dichotomie décrits à l'aide des opérateurs du langage Xcas.	Xcas
		sequentiel(liste,valeur):={
		local rang;
		rang:=0;
		tantque liste[rang]<>valeur **faire**
		rang:=rang+1;
		ftantque;
		retourne rang;
		};;

L	Langage Xcas.	
Σ	Pas de structure de contrôle évoquée dans la ressource.	

On note aussi la présence de la conception AI-objet :

	Conception	Extraits du document (illustration)
P	Instance : un algorithme (sous forme d'un programme) *A* (instancié pour deux algorithmes : les recherches séquentielle et dichotomique d'un élément dans une liste triée) Question : Combien au plus d'itérations de la boucle ont lieu pour une liste de taille 2^p ?	1. Combien, au plus, d'itérations de la boucle auront lieu lors d'une recherche pour chacun des deux programmes ? Pour simplifier on se placera dans le cas où le nombre d'éléments dans la liste en de la forme $n=2^p$.
R	Opérateurs d'ordre mathématique (recherche du pire des cas, étude de suites...)	(b) Pour la recherche dichotomique. Notons υ_n le nombre maximum d'itérations. Pour $n=1$, on a $\upsilon_1=1$. Pour $n=2$, on a $\upsilon_2=2$. Pour $n=2^2$, on a $\upsilon_4=3$... Pour $n=2^p$, les sous-listes contiennent au plus 2^{p-1} éléments. On a donc $\upsilon_2{}^p \leq 1+\upsilon_2{}^{p-1}$ (croissance de la suite (υ_n)). On en déduit $\upsilon_2{}^p \leq p+\upsilon_1$ c'est-à-dire $\upsilon_n \leq 1+\log_2(n)$. Pour n quelconque. On a : $n \leq 2^p$ où l'on a posé $p=[\log_2(n)]$. On a donc $\upsilon_n \leq \upsilon_2{}^p \leq 1+p$. Soit $\upsilon_n \leq 1+[\log_2(n)]$.
L	Langage mathématique et langage naturel.	
Σ	Raisonnements et outils de logique mathématiques valident l'action des opérateurs sur l'algorithme.	On a donc […] On en déduit […] Pour *n* quelconque […] On a donc

Les aspects outil et objet sont donc présents dans cette ressource et on les repère à la fois par les problèmes abordés (reformulés ici dans notre cadre) mais aussi les opérateurs et contrôles mis en jeu. Dans cet extrait, c'est surtout autour de la complexité que l'algorithme existe en tant qu'objet. L'optimalité de l'algorithme de recherche dichotomique n'est pas évoqué ici bien que les questions se centrent sur la complexité.

Vers une application du modèle dans l'analyse des ressources

Ces deux extraits, sur le même thème de la dichotomie, montrent deux conceptions de l'algorithme différentes, l'une centrée sur l'outil, dans AI et AM, l'autre mettant en jeu outil et objet, plutôt dans AI. Ce résultat n'est pas surprenant étant donné que les deux ressources n'ont pas le même objectif ni le même public. Cependant, on peut relever qu'aucun des deux ne traite l'optimalité de la méthode dichotomique dans le cas étudié.

On voit bien dans les deux extraits de ressources sélectionnés comment notre modèle de conceptions guide l'analyse, en mettant l'accent sur les problèmes soulevés mais aussi sur la présence ou non d'opérateurs et de contrôles et leurs caractéristiques. C'est ainsi que nous avons pu produire les analyses dont les résultats sont présentés maintenant.

Savoir à enseigner en algorithmique au lycée : deux transpositions bien distinctes

En étudiant les programmes proposés en mathématiques et en ISN, nous pouvons prendre la mesure de la distance entre les deux transpositions.

Dans le curriculum de mathématiques du lycée

En mathématiques, l'algorithmique est présente de la seconde à la terminale dans toutes les voies générales. On retrouve, dans chacun des programmes officiels, le même paragraphe d'algorithmique déclinant les mêmes objectifs pour le lycée. Dans le détail des contenus de chaque année, on retrouve différents algorithmes attendus explicitement ou le travail explicite de l'algorithmique à certains moments. Le document d'accompagnement de seconde explicite les type d'activités attendues en algorithmique et les choix d'utilisation et de présentation des algorithmes. Nos remarques les plus importantes sur ces instructions officielles, c'est-à-dire sur le savoir à enseigner, sont les suivantes :

- Le concept d'algorithme ne fait pas l'objet d'une définition dans ces curriculums.
- L'algorithmique y est présentée comme une démarche.
- Les concepts d'algorithme et de programme sont très peu distingués, voire confondus.
- L'algorithme est uniquement présenté sous son aspect outil.
- L'activité autour des algorithmes est assez souvent réduite à une activité langagière, dans le but de produire un programme.
- Le système de représentation principal et très majoritaire est AI.

Ces résultats montrent une première étape de transposition didactique de l'algorithme montrant une distance avec le concept tel que nous l'avons étudié plus haut. Les figures 4 et 5 présentent des extraits des instructions officielles illustrant nos analyses.

La grille ci-dessous peut donner quelques éléments en ce sens, non limitatifs il va de soi, et on adaptera ce questionnement aux situations effectivement rencontrées.

Critère	Excellent	Bon	Moyen	Insuffisant
Respect des bons usages Le but visé par l'algorithme est explicité, et des commentaires précisent le déroulement. Les variables ont des noms bien choisis.	Aucune erreur	De petits détails sont négligés. Le but est difficile à déterminer	Des détails manquent, mais le programme tente quand même d'accomplir ses fonctions essentielles.	Ne répond pas au problème posé. Objectif impossible à déterminer
Correction du code : L'algorithme fonctionne.	Fonctionne correctement dans **tous** les cas.	Fonctionne pour des données (entrées) standard mais échecs mineurs sur des cas particuliers.	Échoue pour des données (entrées) standard, mais pour une raison mineure.	Échoue pour des données (entrées) standard, pour une raison importante.
Interface utilisateur : (entrées, sorties) Elle est claire et commode.	Aucune faute	1-3 fautes mineures	Plus de trois fautes mineures ou une faute majeure	Plus d'une faute majeure

Figure 4 : Grille d'évaluation des algorithmes proposée dans le document d'accompagnement de seconde, on n'y retrouve que des critères relatifs à la maîtrise du langage de programmation et aux bonnes pratiques de programmation.

Dans le curriculum d'Informatique et Sciences du Numérique

Les instructions officielles de l'enseignement de spécialité de terminale scientifique Informatique et Sciences du Numérique (ISN) montrent une transposition bien différente.

Même s'il s'agit d'un enseignement uniquement à destination d'élèves de terminale scientifique, on relève des éléments qui vont au-delà d'une question de difficulté conceptuelle mais bien de choix de transposition didactique d'un objet, dont atteste la figure 5 :

- Une définition d'algorithme, assez proche de celle que nous avons rapportée, est donnée dans le programme.
- L'algorithmique est présentée comme une branche de l'informatique.
- La distinction est faite, dans le savoir à enseigner, entre algorithme et programme.
- L'algorithme est présent à la fois comme outil et comme objet (par exemple, par une référence à la notion de complexité).
- L'algorithmique n'est pas restreinte à la programmation : certains algorithmes sont explicitement mentionnés comme trop complexes pour être programmés par l'élève mais dont les principes doivent être connus.
- Comme corollaire de cela, on retrouve les systèmes de représentation AI et AM.

On retrouve donc ici des choix curriculaires qui font entrevoir un positionnement vis-à-vis de l'algorithmique très différent du précédent. Il peut même sembler, à première vue, paradoxal de retrouver ici certains éléments que nous aurions plutôt attendus en mathématiques, notamment les algorithmes non restreints à la programmation, la présence du paradigme AM, ou une définition du concept d'algorithme.

Informatique et Sciences du Numérique	Mathématiques
Le programme est construit autour de quatre parties : représentation de l'information, **algorithmique**, langages et programmation, architectures matérielles. […] Un algorithme **se définit** comme une méthode opérationnelle permettant de résoudre, en un nombre fini d'étapes clairement spécifiées, toutes les instances d'un problème donné. Cette méthode peut être exécutée par une machine ou par une personne. […] On présente simultanément les notions d'algorithme et de programme **puis on les distingue**. […] L'objectif est une compréhension de ces algorithmes et la capacité à la mettre en œuvre.	La **démarche algorithmique** est, depuis les origines, une composante essentielle de l'activité mathématique. […] Ce qui est proposé dans le programme est une formalisation en langage naturel propre à donner lieu à **traduction** sur une calculatrice ou à l'aide d'un logiciel. Il s'agit de familiariser les élèves avec les grands principes d'organisation d'un algorithme : gestion des entrées-sorties, affectation d'une valeur et mise en forme d'un calcul. […] À l'occasion de l'écriture d'algorithmes et de petits programmes, il convient de donner aux élèves de bonnes habitudes de **rigueur** et de les entraîner aux **pratiques systématiques de vérification et de contrôle**.

Figure 5 : Extraits des programmes de mathématiques et d'ISN, des choix de transpositions didactiques très distincts apparaissent (mis en gras par nous).

Une analyse plus approfondie de la transposition didactique en mathématiques

Le travail d'analyse de la transposition didactique de l'algorithme dans l'enseignement de mathématiques du lycée français, soutenu par le modèle épistémologique développé, nous a permis de dégager plusieurs indicateurs relatifs au savoir enseigné. Ce que l'on retrouve dans les ressources et manuels est en totale continuité avec ce que proposent les programmes. L'algorithme est essentiellement traité comme un outil, le système de représentation très majoritaire est AI et l'activité est très souvent de l'ordre de la traduction dans un langage de programmation comme l'attestent les exemples de la figure 6 ci-dessous.

En ce qui concerne les ressources des IREM, il faut distinguer deux types de ressources : des ressources pour la classe et des ressources pour la formation des enseignants. La figure 7 montre la prédominance, pour la classe, d'activités de programmation et mettant en jeu l'algorithme uniquement comme outil.

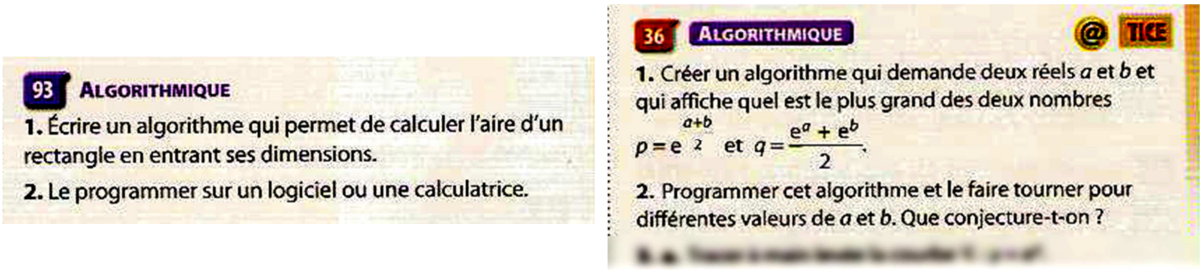


Figure 6 : Deux exercices d'algorithmique. Manuels Math'x (Didier) seconde et terminale S : Des tâches de traduction de formules mathématiques dans un langage de programmation.

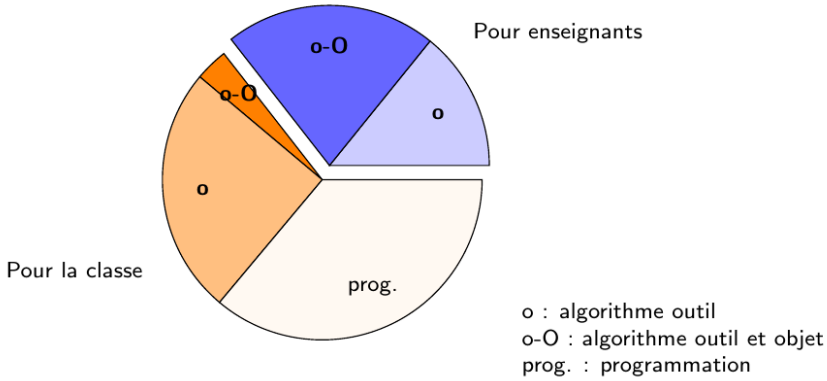


Figure 7 : Répartition des ressources en algorithmique en ligne des IREM, entre ressources pour la classe et ressources pour la formation des enseignants et statut de l'algorithme dans ces ressources. Prédominance de l'algorithme outil et de la programmation pour la classe.

L'algorithme, comme objet, n'apparaît pratiquement que dans les ressources pour la formation des enseignants. L'exemple de la figure 8 montre ce phénomène, en réservant aux enseignants les questions où l'algorithme est objet : les premières questions portent sur la construction et l'implémentation d'algorithmes (on est dans l'outil, comme indiqué sur la figure) et les deux dernières questions, mettant en jeu l'algorithme comme objet via la complexité (recherche d'instances induisant un certain comportement de l'algorithme et optimalité de l'algorithme), font l'objet d'une suggestion de ne pas les proposer aux élèves.

7.1 Je cherche un nombre entre 1 et 1000

Prérequis algo : Instructions conditionnelles et boucles conditionnelles.
Prérequis math : Mathématiques du collège. Récurrence pour la preuve de la dernière question.

Ce jeu se joue à deux joueurs A et B. Le joueur A choisit secrètement un nombre cible compris strictement entre 1 et 1000. Le joueur B doit deviner ce nombre en faisant le minimum de propositions. À chaque proposition du joueur B, le joueur A répond par « le nombre cherché est plus grand », « le nombre cherché est plus petit » ou « bravo, vous avez gagné » selon la position de la proposition par rapport à la cible à atteindre.

Exercice 1 :

Le but de cet exercice est de jouer contre l'ordinateur.

Question 1 – Proposer un algorithme pour que l'ordinateur tienne le rôle du joueur A.

Question 2 – Programmer et tester cet algorithme en langage Python. On utilisera les instructions suivantes :

```
from random import *  // pour importer la librairie
randrange(1000)       // pour tirer un nombre entre 1 et 1000
```

[...]

Question 5 – Écrire et tester cet algorithme en langage Python.

Question 6 – Quels sont les nombres à choisir pour que l'ordinateur trouve la solution en au moins 9 coups ?

Question 7 – Montrer que l'algorithme « optimum » permet de toujours trouver l'entier en au plus 10 essais.

[...]

Remarque – Dans une situation pour la classe, on utilisera les mêmes exercices sans les questions 6 et 7.

Algorithme outil

Algorithme objet

Figure 8 : Extrait d'une ressource pour la formation des enseignants de l'IREM de Grenoble. Les questions 1 à 5 mettent en jeu l'algorithme outil, puis dans les questions 6 et 7 l'algorithme devient objet. La remarque finale montre une restriction de l'algorithme objet à la seule formation des enseignants (annotations à droite par nous).

Trois concepts didactiques pour éclairer la transposition didactique

L'analyse des ressources nous a amenés à proposer trois concepts didactiques, témoins des phénomènes de transposition didactique en jeu, et caractéristiques de ce que l'on retrouve dans l'ensemble des ressources pour le lycée étudiées.

Programme-papier

Nous avons appelé programme-papier le fait de présenter les algorithmes comme des programmes (au niveau de la forme) et dans lesquels, bien que l'on ne soit pas sur une machine, on se soucie de points liés à l'implémentation et à la machine. La figure 9 donne deux exemples de tels programmes-papiers. Ces objets témoignent d'une activité concentrée sur le système de représentation AI, sur l'activité algorithmique restreinte à une activité langagière et, souvent, à un amalgame entre programme et algorithme.

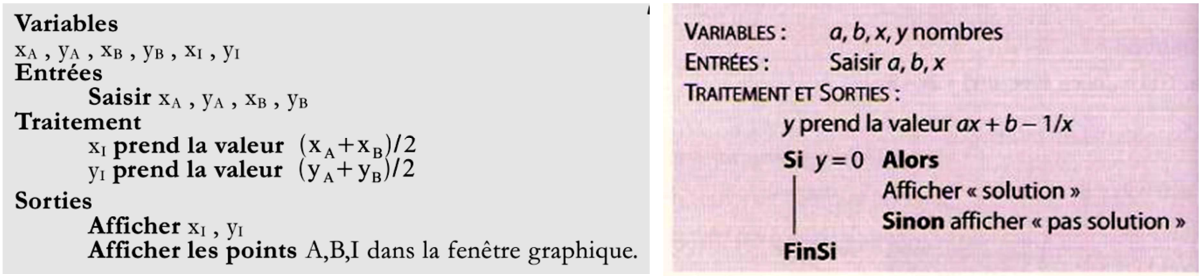


Figure 9 : Deux programmes-papier issus du document d'accompagnement de seconde et du manuel Math'x (Didier) première S. On retrouve des éléments de saisies, d'affichage de texte

à l'écran ou de points dans un fenêtre graphique

Algorithme-instancié

Un algorithme-instancié est l'expression d'un algorithme dont les variables représentant l'entrée ont été remplacées par des valeurs précises. Un algorithme-instancié est à mi-chemin entre un algorithme et son exécution sur une instance. La figure 10 donne deux exemples d'algorithmes-instanciés. Ces objets posent un problème didactique vis-à-vis des relations algorithme-problème et algorithme-preuve, puisqu'on ne s'intéresse qu'à une instance donnée du problème. Quelle généralité représente la méthode décrite ? Comment montre-t-on que l'algorithme répond correctement à toutes les instances s'il n'y en a qu'une ? Si l'on décrit une méthode générale erronée sur une instance pour laquelle le résultat obtenu est valide, comment rejeter cette méthode ?

Mettre 5000 dans S
Mettre 0 dans N
Effacer l'écran
Tant que S est strictement inférieur à 8000
    Remplacer S par S*1,02
    Augmenter N de 1
    Afficher N et S
Fin du Tant Que

début
  Donner à *res* la valeur 1
  pour *i de 1 à 10* **faire**
    Donner à *res* la valeur $res * i$
  fin
  Afficher *res*
fin

Algorithme 7 : Factorielle "Pour"

Figure 10 : Deux algorithmes-instanciés issus du document d'accompagnement de seconde et des ressources des IREM. On ne peut pas différencier ces procédures d'autres, erronées, qui donneraient le même résultat (N=24 pour le premier et res=3628800 pour le second).

La présence d'algorithmes-instanciés est témoin d'une transposition didactique où l'algorithme est restreint à un outil ou à une formulation spécifique d'une méthode de résolution et où la notion d'instance est mise de côté.

Programme de modélisation-simulation

Nous avons appelé programme de modélisation-simulation l'expression d'une méthode systématique visant à simuler un phénomène ou un modèle donné de ce phénomène. Il peut s'agir de simuler un jeu, une machine, une expérience, un modèle de comportement d'un appareil, etc. Il ne s'agit pas d'algorithme au sens où nous les avons définis, et il n'est pas possible d'exprimer le problème dont ces programmes de modélisations-simulations résoudraient un ensemble d'instances. Ces objets témoignent d'un amalgame entre algorithme et programme, car il s'agit uniquement ici de faire réaliser une ou plusieurs simulations pour, en général, conjecturer une propriété. La figure 11 donnent deux exemples de tels objets.

55 ALGORITHMIQUE **Face cachée**

Un sac contient trois jetons : l'un est rouge, l'autre est noir et le troisième, bicolore, a une face rouge et une face noire.

1. On tire au hasard un jeton et on ne regarde qu'une seule face : elle est noire.
Quelle est la probabilité d'avoir en main le jeton noir ? le jeton bicolore ? le jeton rouge ?

2. a. Programmer un algorithme simulant 2 500 réalisations de la situation précédemment décrite et faire afficher la fréquence d'apparition de l'issue $(n\,;n)$ parmi les issues commençant par n.

b. Les fréquences fournies confortent-elles les probabilités calculées à la question 1. ?

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    n PREND_LA_VALEUR 0
    x PREND_LA_VALEUR random()
    TANT_QUE (x<0.5) FAIRE
        DEBUT_TANT_QUE
        n PREND_LA_VALEUR n+1
        x PREND_LA_VALEUR random()
        FIN_TANT_QUE
    AFFICHER "Paul le poulpe s'est trompé au match n°"
FIN_ALGORITHME
```

Figure 11 : Deux programmes de modélisation-simulation issus du manuel Math'x (Didier) terminale S et des ressources des IREM. Il s'agit ici de simuler des expériences aléatoires.

La présence de ces trois objets, programme-papier, algorithmes-instanciés et programmes de modélisation-simulation dans les ressources pour le lycée soutien les résultats de l'étude de la transposition didactique que nous avons présentés plus haut.

VERS DES SITUATIONS POUR L'ALGORITHME

Sur la base du modèle épistémologique présenté précédemment, nous pouvons proposer une caractérisation des problèmes susceptibles de produire des situations d'apprentissage riches en algorithmique. Les critères que nous proposons sont les suivants :

- Le problème doit faire partie des problèmes pour lequel l'algorithme est outil de résolution, et mettre en jeu une famille d'instance.
- Le concept d'algorithme doit être indispensable à la résolution (il n'y a pas de résolution ne nécessitant pas d'algorithme).
- Le problème soulève des questions dans lesquelles l'algorithme est objet.

Nous ajoutons deux critères non nécessaires mais apportant un enrichissement :

- La résolution du problème peut mettre en jeu plusieurs systèmes de représentation.
- Le problème soulève de nouveaux problèmes qui peuvent être résolus algorithmiquement.

Un problème de pesées

Nous proposons le problème suivant pour répondre à la caractérisation précédente.

> Problèmes des fausses pièces :
> Dans un ensemble de pièces indiscernables se trouvent des fausses pièces. Les vraies pièces pèsent toutes le même poids, les fausses aussi mais leur poids est différent de celui des vraies. À l'aide d'une balance Roberval à deux plateaux, peut-on retrouver les fausses pièces ? Si oui, quelle est la méthode qui permet de les retrouver en effectuant le moins de pesées possible ?

Plusieurs variantes peuvent être étudiées selon qu'il y a une ou au moins une fausse pièce, ou selon que l'on sait si les fausses pièces sont plus légères ou plus lourdes que les vraies ou non.
L'étude mathématique et didactique de ce problème a été menée dans (Modeste, Gravier, & Ouvrier-Buffet, 2010). Cette analyse révèle que le problème répond bien aux critères suivants :

- Le problème se résout algorithmiquement.
- Le concept d'algorithme est essentiel à la résolution.
- Il soulève des questions où l'algorithme est objet (complexité, comparaison, optimalité).
- Il peut mettre en jeu les systèmes de représentation PA et AM.
- Il soulève un ensemble de problèmes algorithmiquement résolubles ou pour lesquel on peut chercher des algorithmes.

Ce problème permet de mettre en jeu une dialectique outil-objet. Le fait qu'il s'agisse d'un problème d'optimisation est un facteur important dans cette perspective.
Des expérimentations ont été menées pour construire l'analyse a priori (Brousseau, 1998) de cette situation, sont aussi présentées dans (Modeste et al., 2010) et ont montré l'intérêt de ce problème. Le travail de thèse propose aussi une liste de problèmes à expérimenter.

CONCLUSION

Ce travail de thèse permis la construction d'un modèle épistémologique du concept

d'algorithme, prenant en compte le savoir savant en mathématiques et en informatique. Le modèle produit a été validé et renforcé par confrontation aux conceptions de chercheurs.
Ce modèle s'est avéré pertinent et efficace pour étudier la transposition didactique du concept au lycée en France, en permettant d'analyser des ressources de différentes formes et aux objectifs variés. Cela permet de révéler une transposition partielle, concentrée sur une activité langagière à visée de programmation et limitant globalement l'algorithme à son aspect outil.
Le modèle permet aussi de proposer une caractérisation des problèmes ayant du potentiel pour l'apprentissage de l'algorithme, de proposer de tels problèmes et d'apporter des éléments, à la fois pour guider leur analyse mathématique mais aussi pour développer leur expérimentation et leur analyse a priori.

Perspectives

Ce travail offre des perspectives pour proposer et expérimenter des activités en algorithmique et développer des ressources pour la classe. Le modèle proposé (ainsi que les concepts de programme-papier, algorithme-instancié, et programme de modélisation-simulation) peut fournir des éléments pertinents pour la formation continue et initiale des enseignants, dans un cadre où les liens entre mathématiques et informatique seront certainement de plus en plus présents dans les curriculums. Enfin, plus généralement, le travail mené montre l'importance de prendre en charge les interactions et les concepts à la frontière entre mathématiques et informatique, à la fois dans l'enseignement des mathématiques mais aussi dans la recherche en didactique des sciences mathématique et informatique. C'est un axe de recherche que nous souhaitons développer, dans la continuité du travail présenté ici.

REFERENCES BIBLIOGRAPHIQUES

AHO, A. V., HOPCROFT, J. E., & ULLMAN, J. D. (1989). *Structures de données et algorithmes*. Paris: Inter Éd, D.L.

ARTIGUE, M. (1990). Épistémologie et Didactique. *Recherches En Didactique Des Mathématiques*, *10*(2.3), 241–285.

BALACHEFF, N., & MARGOLINAS, C. (2005). ck¢ Modèle de connaissances pour le calcul de situations didactiques. In A. MERCIER & C. MARGOLINAS (Eds.), *Balises pour la didactique des mathématiques* (pp. 1–32). Grenoble: La Pensée Sauvage.

BEAUQUIER, D., BERSTEL, J., & CHRETIENNE, P. (1992). *Éléments d'algorithmique*. Masson.

BOUVIER, A., GEORGE, M., & Le LIONNAIS, F. (2005). *Dictionnaire des mathématiques*. Puf.

BROUSSEAU, G. (1998). *Théorie des Situations Didactiques*. La Pensée Sauvage.

CHABERT, J.-L. (2010). *Histoire d'algorithmes - du caillou à la puce* (2nd ed.). Belin.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., & CAZIN, X. (1994). *Introduction à l'algorithmique*. Paris: Dunod. http://opac.inria.fr/record=b1077377

DOUADY, R. (1986). Jeux de cadres et dialectique outil-objet. *Recherches En Didactique Des Mathématiques*, *7*(2), 5–31.

GAREY, M. R., & JOHNSON, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

GIROUD, N. (2011). *Étude de la démarche expérimentale dans les situations de recherche pour la classe*. Université de Grenoble. http://tel.archives-ouvertes.fr

KNUTH, D. E. (1973). *The Art of Computer Programming, Volume I: Fundamental Algorithms* (2nd ed.). Addison-Wesley.

KNUTH, D. E. (1985). Algorithmic thinking and mathematical thinking. *The American Mathematical Monthly, 92*(1), 170–181.

KNUTH, D. E. (1996). *Selected Papers on Computer Science*. Center for the Study of Language and Inf.

LOVÁSZ, L., & PLUMMER, D. (2009). *Matching Theory*. AMS Chelsea Pub.

MODESTE, S. (2009). *La place et le rôle de l'algorithme dans l'enseignement : vers un apprentissage de la preuve*. Université Joseph Fourier.

MODESTE, S. (2012). *Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ?* Université de Grenoble.

MODESTE, S., Gravier, S., & Ouvrier-Buffet, C. (2010). Algorithmique et apprentissage de la preuve. *Repères IREM, 79*, 51–72.

SEDGEWICK, R. (1988). *Algorithms*. Addison Wesley.

VERGNAUD, G. (1990). La théorie des champs conceptuels. *Recherches En Didactique Des Mathématiques, 10*(2-3), 133–170.

ANNEXE

Extrait de : R*essources pour la classe de seconde – Algorithmique*, DGESCO, pp. 25-26 :

4 / Recherche de solution d'équation et d'extremum

a. La dichotomie

On se donne une fonction qui change de signe entre *a* et *b*. Résoudre l'équation *f(x)=0*.

De nombreuses situations abordées dans l'année donneront lieu à la résolution graphique, numérique ou algébrique d'équations et d'inéquations.

Face à la multiplicité de ces problèmes, un algorithme automatisant cette tâche est légitime. Évidemment de nombreux outils logiciels intègrent les fonctionnalités numériques ou formelles permettant cette résolution.

Algorithme 5 : jeu du nombre à deviner

Ce texte propose la programmation d'un petit jeu sur calculatrice, avant d'aborder la dichotomie.

Programmer un jeu : deviner le nombre en six essais.

On demande à l'utilisateur de deviner en moins de six essais un nombre tiré au hasard entre 10 et 100.

On lui indique à chaque fois si le nombre proposé est supérieur ou inférieur au nombre cherché. Sans stratégie, il est difficile d'y parvenir.

Le choix des valeurs 10 et 100 qui encadrent le nombre à trouver, ainsi que le nombre d'essais est à mettre en débat dans la classe. En effet, selon le choix de ces valeurs, il sera ou non possible de déterminer à coup sûr la solution avec une bonne stratégie, ou on pourra seulement optimiser les chances de gagner.

```
Variables
    N nombre choisi par l'utilisateur
Initialisation
    S, un nombre entier au hasard entre 10 et 100
    essai prend la valeur 1
Traitement
    Tant que essai est inférieur ou égal à 6
    |   Saisir N
    |   Si N est supérieur à S alors
    |   |   Affiche « c'est moins »
    |   Si N est inférieur à S
    |   |   Affiche « c'est plus »
    |   Si n=S alors
    |   |   Affiche « gagné »
    |   |   fin de programme
    |   essai prend la valeur essai+1
Sortie
    Affiche « perdu ».
```

Une bonne stratégie conduit à l'algorithme utilisant la dichotomie. Cette méthode consiste, en choisissant à chaque fois la valeur située au milieu de l'intervalle en cours, à réduire de moitié à chaque fois l'amplitude de l'intervalle dans lequel se trouve le nombre et comme 2^6 est égal à 64, le dernier intervalle, sur cet exemple, est d'amplitude 1.

Algorithme 6 : recherche d'un zéro par dichotomie

On transpose cette méthode ici. Cela donne l'algorithme suivant :

```
Variables
    m , valeur milieu de l'intervalle « courant »
Initialisation
    a et b, les bornes de l'intervalle [a ; b]
    f, la fonction (rappel : f change de signe entre a et b)
Traitement
    Pour i variant de 1 à 50
    |   m prend la valeur (a+b)/2
    |   Si f(m) et f (a) sont de même signe alors
    |   |   a prend la valeur m
    |   sinon
    |   |   b prend la valeur m
Sortie
    Affiche a et b
```

Remarque :

Les variables *a* et *b* changent de valeur au fur et à mesure de l'exécution de la boucle. Comme on exécute 50 fois celle-ci la largeur de l'intervalle initial est divisé par 2^{50} (environ 10^{15}), ce qui donne un bon encadrement de la valeur cherchée.