La programmation informatique dans la formation initiale des enseignants de mathématiques au Québec.

Prendre en compte les enjeux algorithmiques.

Résumé : Nous présentons une réflexion sur le rôle à donner à l'algorithmique dans la formation initiale des enseignants de mathématiques au secondaire au Québec, dans un contexte où l'algorithmique n'est pas inscrite dans les programmes officiels du secondaire. Ancré à la fois en didactique des mathématiques et en didactique de l'informatique, notre travail cherche à prendre en compte les enjeux didactiques algorithmiques et mathématiques dans l'utilisation d'activités de programmation informatique pour des futurs enseignants en formation initiale à la fois en didactique et en mathématiques. Il s'agit de préserver la richesse intrinsèque de l'activité algorithmique, au carrefour des sciences informatiques et mathématiques, tout en répondant à la volonté institutionnelle québécoise d'intégrer la programmation informatique à l'école dans une vison très instrumentale et transversale. Le cadre théorique de l'approche instrumentale nous permet d'aborder les liens possibles entre activité algorithmique et activité mathématique dans la conception de ressources algorithmiques pour l'enseignement des mathématiques.

Mots-clés : algorithmique, programmation informatique, formation initiale, enseignement des mathématiques au secondaire, genèse instrumentale.

La programmation informatique au Québec

Au Québec, suivant une dynamique internationale, l'apprentissage de la programmation informatique à l'école prend de l'expansion. Le plan d'action numérique en éducation et en enseignement supérieur (MEES, 2020) prévoit le développement de l'usage pédagogique de la programmation dans les écoles. Barma et al. (2018), dans un rapport visant à proposer des actions optimales à poser pour favoriser l'usage pédagogique de la programmation informatique dans les écoles du Québec, recommandent que la programmation informatique soit ajoutée au curriculum québécois comme une nouvelle compétence transversale en lien avec la pensée informatique. L'algorithmique n'est donc pas considérée comme un domaine d'étude à part entière. De fait, elle n'est envisagée que de façon implicite pour sous-tendre une compétence prenant ancrage dans plusieurs compétences disciplinaires, en mathématiques, en science et technologie, en arts… Il s'agit là d'une vision purement instrumentale (Fluckiger & Bart, 2012) de l'informatique en général, et de la programmation en particulier, qui s'inscrit en rupture assez nette avec les arguments précurseurs sur l'enseignement de l'informatique à l'école, et l 'irréductibilité de l'informatique aux autres sciences :

> L'informatique a sa place à l'école au milieu des autres disciplines scientifiques [...] en raison de sa spécificité, de l'originalité de ses méthodes, et de l'extraordinaire enrichissement de la pensée scientifique qui en est résulté. (Arsac, 1980, p.1)

Dans cette communication, nous nous intéressons à la relation particulière qui lie l'informatique, plus précisément l'algorithmique, et les mathématiques scolaires. Nous abordons cette question dans le cadre de la formation initiale des enseignants de mathématiques au secondaire. Cette formation commence dès la première année d'université. Les étudiants n'ont donc pas reçu de formation universitaire en mathématiques préalablement à leur formation didactique, et doivent développer simultanément des connaissances mathématiques, des connaissances didactiques et des connaissances pédagogiques. En deuxième année, ils doivent suivre un cours d'initiation à la programmation informatique. Nous cherchons à savoir si l'activité algorithmique qu'ils développent dans ce cours peut favoriser également la mobilisation de connaissances ou de raisonnements mathématiques. Notre réflexion relève donc à la fois de la didactique de l'informatique et de la didactique des mathématiques. Du côté mathématique, nous nous inscrivons dans le cadre théorique de la genèse instrumentale (Rabardel, 1995) appliquée à l'enseignement des mathématiques (Trouche, 2005). La didactique de l'informatique nourrit notre réflexion sur l'algorithme en tant qu'objet à apprendre ou à enseigner (Caron et al., 2020), et des liens qu'il entretient avec l'activité mathématique (Modeste, 2012).

Enseigner l'algorithmique

La dualité objet ou outil d'enseignement

L'informatique, en tant qu'objet à enseigner ou à apprendre, ne se laisse pas facilement appréhender (Caron et al., 2020). Elle comporte par nature une pluralité interne (Bruillard, 2016 ; Lang, 2009 ; Mirabail, 1990). Il n'est donc pas facile d'identifier dans les pratiques d'enseignement des activités et des objets propres à l'informatique (Fluckiger, 2019), ni, dès lors, d'identifier les frontières entre informatique et disciplines scolaires. Les images mentales que se forgent les élèves à propos des programmes et de leur exécution lors de leurs premiers contacts avec la programmation sont difficiles à appréhender (Grandbastien, 1988). Le regard que porte la didactique de l'informatique sur son objet de recherche révèle plusieurs tensions, entre science et technique d'une part, entre ensemble de connaissances spécialisées et pratiques instrumentées de l'autre. Baron (2018) propose une structuration des enseignements scolaires relatifs au numérique qui rend compte de ces tensions. Les enseignements informatiques relèvent de trois ensembles ayant une intersection commune non vide : la science du traitement de l'information proprement dite (incluant tout ce qui relève du développement d'une « pensée informatique », le codage et l'acquisition de concepts informatiques), la culture technique (incluant l'appropriation d'outils comme les traitements textes ou le tableur), la culture citoyenne (incluant les enjeux éthiques et les réseaux sociaux). L'enseignement de l'algorithmique, surtout quand il est mis en œuvre à travers des activités de programmation informatique, peut relever des trois composantes à la fois. Il importe donc, dans la conception de ressources pédagogiques, d'être conscient de cette multiplicité. Modeste (2012), dans le même ordre d'idée, souligne la dualité de l'algorithme dans son rapport aux mathématiques :

> Cette dualité nous semble particulièrement utile en ce qui concerne l'algorithmique : la particularité des algorithmes d'être des méthodes de résolution de problème (aspect problème) faciles à transférer (aspect effectif) implique un risque de ne considérer l'algorithmique uniquement sur ce point de vue outil (notamment dans un contexte d'enseignement). Or l'analyse ci-dessus laisse percevoir un objet riche avec un potentiel fort d'enrichissement de l'activité mathématique. (p.37)

Dans la lignée de Modeste (2012), Briant (2013) et Laval (2018), nous souhaitons prendre en compte cette dialectique objet/outil, au sens de Douady (1986), ce qui suppose de prendre en compte le fait

qu'« un algorithme n'est pas uniquement un outil pour la résolution de problèmes mais c'est aussi un objet mathématique à part entière » (Modeste, Gravier et Ouvrier-Buffet, 2010, p. 52)

Des instruments algorithmiques pour travailler les mathématiques ?

On constate, aussi bien en France (Laval, 2018), qu'au Québec (Barma, 2020), une volonté institutionnelle de nier l'importance de la dualité objet/outil de l'algorithmique et de réduire son enseignement à un outil pour la résolution de problèmes ou la construction de connaissances mathématiques. Or les enjeux didactiques dans l'enseignement de l'algorithmique sont multiples et rendent impossible la réduction d'un algorithme à un artefact favorisant la médiation entre les élèves et les concepts mathématiques. Il nous paraît plus riche de promouvoir un enrichissement mutuel entre activité algorithmique et mathématique. Cela nous amène, dans la lignée de Lagrange et Rogalski (2017), à interroger les concepts et les compétences relevant de chacune de ces activités, et leurs possibles corrélations. C'est à travers l'approche instrumentale de la didactique que nous abordons ces liens. Initialement développée par Rabardel (1995) dans une perspective d'ergonomie cognitive, cette théorie repose sur la distinction fondamentale entre un artefact, initialement conçu pour répondre à des objectifs précis, et l'instrument qui est construit par un sujet à partir de cet artefact au cours de son usage dans une activité située. Un artefact n'est donc qu'une proposition, qui peut être développée ou non, par un sujet lors d'une activité (Trouche, 2005). La genèse instrumentale désigne le processus par lequel un instrument est construit par un sujet engagé dans une action mettant en jeu un artefact. Cette construction passe par la mise en place et le développement de schèmes d'utilisation par le sujet engagé dans une action, et par l'enrichissement des propriétés de l'artefact. Dans la lignée de Gueudet et al. (2014), nous cherchons à répondre à la question suivante : « Dans quelle mesure l'écriture de programmes informatiques peut-elle se constituer en instrument de travail permettant d'approfondir la compréhension des notions mathématiques ? »

Une étude préliminaire

Méthodologie

Nous avons mené une étude préliminaire, dans le contexte du cours d'initiation à la programmation informatique inclus dans la formation des enseignants de mathématiques au secondaire à l'UQAM. Pour cela nous avons observé des étudiants en train de résoudre, par équipes de deux, des tâches de programmation informatique. L'organisation de la salle est flexible, ce qui permet aux équipes de collaborer. Nous avons enregistré les conversations des étudiants, et recueilli les fichiers informatiques qu'ils ont produits (fichiers Scratch et Python). Les tâches que nous avons sélectionnées ont en commun de chercher à établir des liens entre mathématiques et informatique, soit en proposant la mise en œuvre de concepts centraux en algorithmique dans un contexte mathématique, soit en permettant l'exploration d'une question mathématique classique via la construction d'un algorithme. La première s'intéresse à l'algorithme comme objet d'enseignement, l'objectif étant la compréhension de la boucle *jusqu'à ce que*. La deuxième vise la construction d'un instrument pour comprendre une propriété mathématique, à travers l'écriture d'un algorithme informatique. Les grilles d'analyse ont été conçues pour mettre au jour les traces des schèmes instrumentaux mis en place et les concepts mathématiques et algorithmiques convoqués. L'analyse des programmes informatiques produits par les étudiants permet également d'observer la mobilisation des variables informatiques et son adéquation avec les structures de contrôle utilisées.

Nous présentons ici deux de ces tâches et quelques éléments d'analyse du travail réalisé par les étudiants lors de leur mise en œuvre dans le cadre de la formation initiale des enseignants de mathématiques au secondaire à l'UQAM.

Tâche 1 : la division euclidienne

La première tâche que nous présentons a été construite autour d'un enjeu algorithmique : la mise en œuvre de la structure de contrôle *jusqu'à ce que* (nous travaillons dans Scratch). Pour cela, nous avons demandé aux étudiants d'écrire un programme permettant de tester la divisibilité d'un nombre entier par un autre, sans avoir recours aux opérateurs *arrondi*, *modulo* ou *partie entière*. D'un point de vue mathématique, la contrainte de ne pas utiliser d'opérateur prédéfini force le passage vers une conception de la division entière par multiplication ou soustraction répétée. Du point de vue algorithmique, cette même contrainte amène à structurer l'algorithme avec une boucle *jusqu'à ce que*. On cherche ici à exploiter la dualité objet/outil de cette structure de contrôle : elle constitue l'objet du travail algorithmique tout en jouant un rôle d'outil permettant de revenir à la définition de la division euclidienne et de clarifier son statut opératoire. L'algorithme peut ainsi se constituer en un instrument permettant de rendre explicite le processus itératif sous-jacent à toute division.

Toutes les équipes ont spontanément opté pour une conception de la division comme soustraction répétée, plutôt que multiplication répétée. Ce choix semble avoir été conditionné par une tâche vue préalablement dans le même cours lors du travail sur l'instruction conditionnelle. Les étudiants avaient écrit un programme permettant de tester la parité d'un nombre. En réalisant cette activité, les étudiants ont activé le schème d'utilisation de l'opérateur *modulo* pour tester la divisibilité entre deux nombres. Ce schème est encore très présent au moment où ils se lancent dans la présente tâche. Ils cherchent donc spontanément à calculer le reste de la division entière de *a* par *b*, puis en viennent à devoir expliciter le travail de l'opérateur *modulo* qui leur est interdit. En revanche, l'écriture de la boucle *jusqu'à ce que,* dont la genèse a été activée plus anciennement, est plus difficile pour eux. Les schèmes algorithmiques visés par l'activité sont « choix d'un test de fin » et « gestion de l'incrémentation des variables ». Le premier schème émerge naturellement dans la plupart des équipes, mais son installation est encore fragile chez d'autres. La figure 1 montre un extrait du programme écrit par l'équipe 1. Le recours à une boucle infinie non nécessaire traduit chez ces étudiants une incertitude quant à l'efficacité de leur test d'arrêt. Cette incertitude traduit une instrumentation encore fragile de l'artefact « boucle *jusqu'à ce que* ».

Figure 1. Recours à une boucle infinie par l'équipe 1

Ces étudiants ont utilisé la boucle pour calculer le reste, puis sont revenus à la définition de la division pour calculer le quotient. Bien que faisant l'objet d'un apprentissage, la boucle *jusqu'à ce que* se constitue en outil pour expliciter le processus de division.

L'équipe 3 a choisi de calculer le quotient à même la boucle, en incrémentant un compteur. Paradoxalement, cette même équipe, qui semble plutôt à l'aise avec l'incrémentation, n'a pas acquis le schème permettant d'affecter à une variable sa valeur initiale incrémentée (par exemple, reste←reste-b). Les étudiants ont donc recours à une variable intermédiaire « NouveauReste » inutile (figure 2). La genèse instrumentale algorithmique relative à l'affectation, et qui se traduit par une utilisation adéquate de l'instruction « mettre … à … » est encore fragile. L'existence dans Scratch de deux instructions différentes pour l'affectation (« ajouter … à… » et « mettre … à … ») pourrait favoriser cette conceptualisation séparée de l'affectation et de l'incrémentation. On voit ici l'influence de l'artefact sur la conceptualisation algorithmique. Par ailleurs, cette même équipe a produit un algorithme à la structure complexe, envisageant plusieurs cas de figure non nécessaires (cas où le quotient est nul, cas où la division aboutit en une seule étape). L'instrument d'explicitation mathématique que nous avions anticipé est construit, mais il reflète une conception de la division qui n'a pas encore le statut d'un processus général s'appliquant de la même façon à toute paire de nombres entiers.

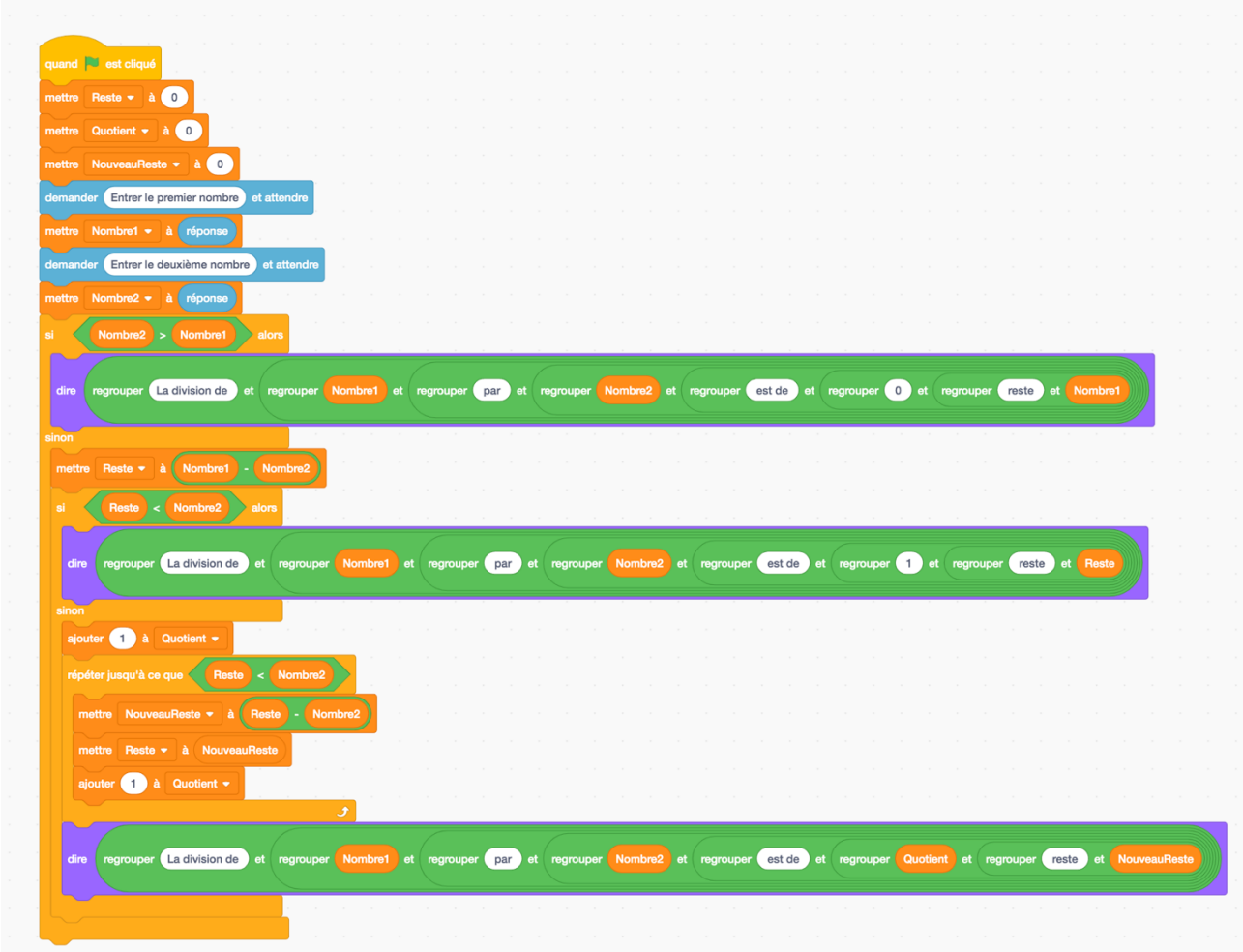

Figure 2. Incrémentation et structuration par cas particuliers pour l'équipe 3

Tâche 2 : simplification de la racine carrée.

Notre démarche d'exploration des liens entre mathématique et algorithmique nous a également amenés à partir de propriétés classiques en mathématiques et à explorer les effets de leur transposition dans une démarche algorithmique. Nous avons retenu ici l'exemple de la simplification de la racine carrée proposé par Briant et Bronner (2015). Nous demandons aux étudiants d'écrire un programme

permettant de simplifier la racine carrée de n'importe quel entier naturel *n*, c'est-à-dire d'afficher une écriture sous la forme $a\sqrt{b}$ avec *b* le plus petit possible. Notons qu'au Québec, les étudiants n'ont pas été habitués, dans leurs études secondaires, à simplifier systématiquement les racines carrées. Ce n'est donc pas une pratique habituelle pour eux. Certains d'entre eux ne connaissent pas cette propriété des nombres entiers. Notre idée est donc de les amener à construire un instrument de travail mathématique leur permettant de l'explorer, de la comprendre et de se l'approprier.

Nous présentons ci-dessous l'exploration réalisée par deux étudiantes, Nathalie et Geneviève. Elles passent les dix premières minutes de l'activité à essayer de comprendre la propriété mathématique en travaillant sur des exemples. Elles ne sont pas très à l'aise avec la notion de racine carrée, ce qui semble leur faire perdre momentanément l'accès au sens mathématique de la divisibilité : *« On a n est égal à 20, je vais commencer par tester si la racine de 19 divise 20. (Inaudible). Mais c'est quoi la division ? Comment je fais ma division avec une racine ? »* À ce stade, leur réflexion est essentiellement algorithmique. Elles activent alors deux schèmes d'utilisation algorithmiques bien ancrés, celui consistant à recourir à l'opérateur *modulo* pour tester la divisibilité, et celui associant à tout algorithme une structure de boucle. Elles cherchent donc à raisonner en termes d'itérations d'une opération, mais elles peinent à trouver cette opération. Elles sont handicapées par une conception fragile des liens entre la racine carrée d'un nombre et son carré et finissent par se perdre : « racine de 16, racine de 4 fois racine de 4 mais là tu peux toujours avoir racine de 20 donc… je comprends pas ce que je fais… ». Elles abandonnent ce premier algorithme et entrent dans une réflexion plus mathématique : « *20 pourquoi ça marchait ? C'est parce que j'avais 2 et 2 donc là je prenais ça. Racine de 4 fois 5 […]. Ok, mais je peux pas faire racine de trois fois 15. …Donc là si je fais b=15. Y en a pas un que je peux réduire, mais […] mettons que j'ai 3, 3 et 5, ça fait 9 fois 5, donc je peux le faire avec 45.* ». La recherche d'un algorithme se constitue alors en un instrument d'exploration mathématique. Elles décident de quitter le papier-crayon et de commencer à programmer. Elles reprennent un programme déjà travaillé dans le cours et permettant d'obtenir la liste des diviseurs premiers d'un entier *n*. Ce faisant, elles poursuivent leur exploration mathématique : « *on prend les deux premiers éléments et on multiplie tous les autres. Mais le truc c'est que c'est pas nécessairement les premiers. Mon problème, c'est que si, mettons, j'avais 75, il faudrait prendre les deux derniers.* ». Elles réalisent alors que l'algorithme vers lequel elles s'orientent est valide mathématiquement, mais difficile à programmer. C'est ici qu'entre en jeu la dialectique objet/outil de l'algorithme. Une meilleure connaissance des algorithmes de gestion des listes (objet d'un apprentissage algorithmique) leur aurait permis d'aboutir dans leur démarche et d'écrire elles-mêmes un programme (outil de mobilisation de connaissances mathématiques) leur permettant d'explorer la propriété mathématique en jeu. Au lieu de ça, un peu découragées, elles demandent de l'aide à leur collègue Arturo. Arturo va alors leur imposer l'algorithme qu'il a construit et qui consiste à trouver le plus grand nombre *a* tel que a^2 divise *n*. Elles écrivent sous sa dictée, mais ne comprennent pas vraiment ce qu'elles font. Ce n'est que quand il les aura laissées et qu'elles débogueront le programme qu'elles comprendront à la fois l'algorithme et son résultat : *« parce que, dans le fond, 5 racine de 3 c'est égal à racine de 5 au carré fois 3, c'est racine de 25 fois 3, donc racine de 75. C'est-tu pas merveilleux ? ».* Ce faisant, elles n'accèdent qu'à l'aspect outil de l'algorithme.

Conclusion

Les résultats obtenus montrent à quel point la dialectique objet/outil intervient dans toute tentative de construire un instrument de travail mathématique à partir d'un algorithme. Une genèse instrumentale

algorithmique trop peu solide ne permet pas en effet la construction d'un instrument de travail mathématique intéressant. L'algorithme ne peut pas, alors, jouer le rôle d'un outil pour la mobilisation de connaissances mathématiques. Par exemple, dans la tâche 1, le travail algorithmique montre, pour l'équipe 3, que le processus de généralisation de l'opération division entière est encore en cours, mais que les connaissances encore fragiles sur la boucle *jusqu'à ce que* ne favorisent pas sa finalisation. Réciproquement, si les connaissances mathématiques ne sont pas assez solides pour que l'apprenant s'engage dans la recherche d'un algorithme, les schèmes algorithmiques visés par la tâche ne peuvent pas être activés ou mis en place. La tâche échoue alors dans son objectif de construction de connaissances sur l'algorithme en tant qu'objet d'enseignement. Cependant, le travail réalisé par Nathalie et Geneviève montre que même les étudiants peu avancés peuvent bénéficier des allers-retours entre pensée mathématique et algorithmique. L'exploration algorithmique leur a permis de réactiver des connaissances mathématiques relatives à la manipulation du radical, et à la relation inverse entre la racine et le carré. L'algorithme mathématique qu'elles ont partiellement mis en place, en passant par la décomposition en facteurs premiers, est très proche de la démonstration mathématique de la propriété. Il correspond à ce que Briant et Bronner (2015) appellent la résolution mathématique, qui doit, selon eux, subir une double transposition pour devenir un algorithme informatique. Dans notre cas, les étudiants possédaient les connaissances et les outils pour implémenter la décomposition en facteurs premiers, le programme ayant été écrit dans un exercice précédent. Cependant, elles étaient trop peu familières avec la recherche d'éléments dans une liste pour poursuivre dans cette voie qui aurait provoqué une avancée considérable dans leur genèse instrumentale algorithmique. Elles ont alors adopté la résolution algorithmique proposée par Arturo, qui reprenait la piste qu'elles avaient suivie initialement, puis abandonnée, faute de la comprendre mathématiquement. La complémentarité de ces approches se révèle particulièrement intéressante dans le travail de débogage réalisé par Nathalie et Geneviève. C'est en effet au cours de cette ultime étape qu'elles en viennent à comprendre simultanément la structure itérative de l'algorithme proposé par Arturo et les enjeux mathématiques de la propriété étudiée. Ici c'est donc davantage la compréhension de l'algorithme que sa construction qui se constitue en instrument de travail mathématique.

Les deux tâches proposées ici diffèrent dans leurs intentions didactiques. Cependant, elles ont en commun de reposer sur une exploration mathématique préalable, nécessaire pour déterminer les opérations mathématiques à itérer. Dans les deux cas, la construction de l'algorithme permet de revisiter des connaissances mathématiques. La dualité objet/outil de l'algorithme est incontournable et exploitable au profit d'une exploration mathématique. Exploiter un contexte mathématique simple peut permettre de mettre en place des schèmes algorithmiques avancés. La complexité mathématique peut cependant faire obstacle à la genèse instrumentale algorithmique. En effet, certains algorithmes peuvent être plus pertinents d'un point de vue mathématique, mais plus difficiles à traduire en langage informatique. Réciproquement, une genèse algorithmique avancée peut bloquer l'accès à des algorithmes mathématiques plus efficaces, notamment en contexte de preuve. Enfin, la confrontation d'algorithmes différents résolvant une même question, et le débogage sont des activités à favoriser explicitement. Ces considérations sont à prendre en compte dans la formation continue et initiale des enseignants de mathématiques.

Bibliographie

Arsac, J. (1980). *Premières leçons de programmation*. Cedic.

Barma, S. (2018). *Rapport final : Réaliser une étude de cas multiple qui vise à affiner les connaissances sur l'usage pédagogique ou didactique de la programmation dans les écoles du Québec* (Rapport no 118982). Québec, Québec : Université Laval. https://lel.crires.ulaval.ca/oeuvre/rapport-final-realiser-une-etude-de-cas-multiple-qui-vise- affiner-les-connaissances-sur

Baron, G.-L. (2018). Informatique et numérique comme objets d'enseignement scolaire en France : entre concepts, techniques, outils et culture. Dans G. Parriaux (resp.), *De 0 à 1 ou l'heure de l'informatique à l'école*. Colloque Didapro 7–DidaSTIC. Lausanne, Suisse, Peter Lang.

Briant, N., & Bronner, A. (2015). Étude d'une transposition didactique de l'algorithmique au lycée : une pensée algorithmique comme un versant de la pensée mathématique. Dans L. Theis (resp.), Pluralités culturelles et universalités des mathématiques : enjeux et perspectives pour leur enseignement et leur apprentissage. Colloque EMF2015–GT3 (pp. 231-246). Alger, Algérie.

Bruillard, É. (2016). Quelle informatique à repenser et à construire pour les élèves de l'école primaire. Dans F. Villemonteix, G.-L. Baron et J. Béziat (dir.), *L'école primaire et les technologies informatisées. Des enseignants face aux TICE* (p. 29-37). Presses Universitaires du Septentrion.

Caron, P.-A., Fluckiger, C., Marquet, P., Peter, Y., & Secq, Y. (2020). *L'informatique, objets d'enseignements enjeux épistémologiques, didactiques et de formation*. Université de Lille. Colloque DIDAPRO 8 -DIDASTIC, Lille, France.

Douady, R. (1986). Jeux de cadre et dialectique 'outil-objet'. *Recherches en Didactique des Mathématique*s, 7 (2), 5-32.

Fluckiger, C. (2019). *Une approche didactique de l'informatique scolaire*. Presses universitaires de Rennes.

Fluckiger, C., & Bart, D. (2012). L'introduction du B2i à l'école primaire : évaluer des compétences hors d'une discipline d'enseignement ? *Questions vives. Recherches en éducation*, 7(17), 71-87.

Gueudet, G., Buteau, C., Mesa, V., & Misfeldt, M. (2014). Instrumental and documentational approaches : from technology use to documentation systems in university mathematics education. *Research in Mathematics Education*, 16(2), 139-155.

Lang, B. (2009). *L'informatique : Science, techniques et outils*. LexiPraxi 98, journée de réflexion sur le thème «Former des citoyens pour maîtriser la société de l'information», Paris.

Lagrange, J. B., & Rogalski, J. (2017). Savoirs, concepts et situations dans les premiers apprentissages en programmation et en algorithmique. *Annales de Didactiques et de Sciences Cognitives.* 22, 119-158.

Laval, D. (2018). *L'algorithmique au lycée entre développement de savoirs spécifiques et usage dans différents domaines mathématiques*. [Thèse de doctorat, université Sorbonne Paris Cité].

Mirabail, M. (1990). *La culture informatique*. Aster.

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

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

Rabardel, P. (1995). *Les hommes et les technologies ; approche cognitive des instruments contemporains.* Armand Colin.

Trouche, L. (2005). *Des artefacts aux instruments, une approche pour guider et intégrer les usages des outils de calcul dans l'enseignement des mathématiques*. Le calcul sous toutes ses formes. Saint-Flour, France.