Quels sont les enjeux de l'enseignement de l'informatique à l'école ?

Introduction

Il est devenu difficile de comprendre le monde d'aujourd'hui, d'y travailler et, dans une certaine mesure, d'y conduire certaines activités, sans un minimum de compréhension de ce qu'est l'informatique.

Au niveau professionnel, le métier d'informaticien continue de se développer de façon très importante : l'explosion des besoins crée actuellement un manque de main-d'œuvre disponible (Dares, 2022), et le nombre d'emplois va continuer à progresser (Sciberras *et al.*, 2022). Il devient par ailleurs courant que des non-informaticiens soient amenés à effectuer des tâches informatiques ponctuelles (définir ou modifier des formules de calcul dans un tableur, paramétrer un logiciel existant, etc.). Plus généralement, la maîtrise des outils et services informatiques élémentaires (traitement de texte, tableur, navigateur internet, dépôt ou téléchargement de données, gestion de formulaires en ligne, etc.) est devenue une compétence professionnelle de base.

Au niveau des sphères personnelles et sociales, l'équipement (ordinateurs, tablettes, smartphones) et son utilisation (traitement de texte, navigateur internet, jeux, messageries en ligne, réseaux sociaux, musique et vidéos en ligne, etc.) se sont généralisés. L'informatique est devenue utile, sinon nécessaire, à de nombreuses activités : démarches administratives (déclaration d'impôts, etc.), parentalité (informations scolaires diffusées via des plateformes spécialisées, groupes de parents d'élèves ou associations sportives communiquant via des réseaux sociaux, etc.), socialisation (pour les jeunes, les réseaux sociaux sont la façon normale – au sens de : qui constitue la norme – d'être en contact avec les autres, et leur utilisation est donc quasi-nécessaire), etc. Régulièrement, de nouvelles technologies (internet, les réseaux sociaux, l'intelligence artificielle générative comme ChatGPT actuellement) sont annoncées comme allant bouleverser le monde et, dans certains cas, le font.

L'importance de l'informatique dans le monde d'aujourd'hui a logiquement amené à se poser la question de son enseignement à l'école. Les enjeux de cet enseignement sont cependant complexes à cerner : les raisons pédagogiques ou politiques avancées pour mettre en place cet enseignement sont multiples, et de natures très différentes ; ce dont il est question est souvent

confus ; enfin, les travaux scientifiques analysant les pratiques et les résultats de l'enseignement de l'informatique à l'école sont foisonnants, complexes, et parfois difficiles à interpréter.

Cette note propose un cadre pour réfléchir aux enjeux de l'enseignement de l'informatique à l'école. Après avoir précisé quelques éléments de contexte (I), le texte aborde successivement les questions suivantes : quelles sont les raisons avancées pour justifier un enseignement de l'informatique de l'école ? (II) ; quels objectifs d'enseignement peut-on considérer ? (III) ; comment enseigner l'informatique à l'école ? (IV) ; est-ce que les élèves d'âge scolaire sont en mesure de tirer bénéfice de ces enseignements ? (V) ; et enfin : est-ce que la pratique de l'informatique améliore les résultats des élèves dans les autres disciplines ? (VI).

I. Cadre général

L'une des difficultés que pose la réflexion sur l'enseignement de l'informatique à l'école est de cerner ce dont on parle. L'informatique est un domaine complexe, qui mêle des aspects scientifiques et techniques, et évolue constamment. Par ailleurs, tout le monde (élèves, enseignants, formateurs, prescripteurs institutionnels, parents d'élèves) en développe une perception personnelle, liée à ses usages et ses représentations, qui influe sur son analyse. Enfin, l'utilisation de termes encore plus généraux (par exemple, le numérique) ou d'expressions à la mode mais qui n'ont que peu de sens (par exemple, enseigner le code) ajoute à la confusion. Sans chercher à proposer une solution générale à ce problème, cette section propose un cadre (très simplifié) et fixe le vocabulaire utilisé dans ce texte.

Quand on parle d'informatique à l'école, le terme « informatique » peut renvoyer à différentes choses : les dispositifs matériels (ordinateurs, tablettes, smartphones – qui sont de fait de petits ordinateurs –, réseaux, robots, objets connectés) ; les logiciels et principes qui permettent de faire fonctionner ces matériels (les systèmes d'exploitation comme Windows ou Linux, les protocoles et systèmes de codage permettant d'échanger des données sur Internet, etc.) ; les logiciels qui permettent de réaliser les tâches pour lesquelles on utilise l'ordinateur, que l'on nomme le plus souvent « programmes » ou « applications » (le traitement de texte, l'application de messagerie, le lecteur vidéo, etc.) ; ou encore les notions, connaissances, façons de faire, techniques, outils (etc.) qu'ont forgés les informaticiens pour construire ces différents logiciels.

Lorsque les écoles ont commencé à disposer d'ordinateurs, les enseignants les ont principalement utilisés pour initier les élèves à ce que l'on appelle maintenant les 'outils numériques de base' (autrefois désignés sous le terme de 'bureautique') : il s'agit ici de savoir utiliser un ordinateur et des logiciels de type traitement de texte, navigateur internet ou envoi de mails, et de connaître les bonnes pratiques associées. Par ailleurs, certains enseignants ont également utilisé des logiciels éducatifs conçus pour enseigner des disciplines scolaires : programme permettant de construire des figures géométriques ; programme posant des questions sur un sujet donné (par exemple en géographie ou en anglais) et indiquant si les réponses saisies au clavier sont correctes ; programme permettant de 'naviguer' dans un ensemble de ressources – explications textuelles, images, vidéos – consacrées à une thématique donnée (par exemple une période de l'histoire de France) ; etc.

Dans cette approche de l'informatique, les élèves sont en situation d'utilisateurs : il s'agit essentiellement de savoir utiliser un ordinateur, un traitement de texte ou un logiciel éducatif (même si ce peut être l'occasion, pour l'enseignant, d'expliquer un peu comment cela marche).

Ce qui a changé depuis quelques années c'est la volonté de proposer également des enseignements qui abordent certains aspects de l'informatique en tant que domaine scientifique et technique et, notamment, les méthodes et techniques relatives à la construction des programmes. La situation pédagogique prototypique de ce type d'enseignement est de proposer aux élèves de construire un programme qui fait quelque chose (qui résout automatiquement un problème de mathématiques, qui simule un phénomène physique, qui fait se déplacer des objets à l'écran ou un robot posé sur le sol, etc.) et, dans ce contexte, de leur enseigner les approches, les notions et/ou les connaissances informatiques utilisées pour construire ce type de programmes.

Cette évolution a lieu partout en Europe (Bocconi *et al.*, 2022) et plus largement (Hsu *et al.*, 2018). En France, elle a été actée lors de la refonte des programmes scolaires de 2015 : tous les élèves doivent recevoir une initiation à l'informatique dès le cycle 3 (ce qui, en pratique, n'est pas toujours le cas), puis des enseignements spécifiques en cycle 4.

Enseigner comment se construisent (et comment fonctionnent) des programmes informatiques conduit notamment à faire travailler les élèves sur la notion clé d'algorithme. Un algorithme est une suite d'actions dont l'application mécanique permet de résoudre un problème (au sens large). La notion d'algorithme n'est pas propre à l'informatique. Ainsi, au cycle 2, les élèves apprennent et appliquent les algorithmes de l'addition et de la multiplication (aligner les nombres, prendre les chiffres des unités, si la somme/produit est supérieure à 10, alors…). De même, une recette de cuisine est un algorithme. L'algorithmique est cependant centrale en informatique car un programme n'est rien d'autre qu'un algorithme écrit dans un langage de programmation, c'est-à-dire dans un langage qui est interprétable par un ordinateur et permet donc à cette machine d'appliquer mécaniquement les actions que décrit l'algorithme. La 'programmation', appelée parfois 'codage', est donc l'expression d'un algorithme dans un langage de programmation (soit on écrit l'algorithme en français puis on le traduit, soit on l'écrit directement dans le langage de programmation). Le langage de programmation généralement utilisé à l'école est le langage Scratch.

II. Quelles sont les raisons avancées pour justifier un enseignement de l'informatique de l'école ?

La première des raisons généralement invoquées pour justifier l'enseignement de l'informatique à l'école est que cette science/technologie est devenue un élément clé de nos sociétés, et que tous les citoyens devraient donc en avoir une certaine compréhension.

Un second argument a cependant émergé avec la prise de conscience du point suivant. Construire un programme nécessite d'identifier les données sur lesquelles il porte, les résultats qu'il doit produire, et l'algorithme (la suite d'actions) qui permet d'obtenir ces résultats. C'est donc, fondamentalement, une activité d'analyse et de résolution de problèmes. Alors que, aux débuts de l'informatique, l'approche algorithmique a été essentiellement appliquée à des calculs mathématiques, il est maintenant apparu qu'elle s'applique à une grande diversité de problèmes : organiser un plan d'action, trouver le meilleur itinéraire entre deux villes, gérer une bibliothèque ou un compte en banque, modéliser un phénomène de physique ou de chimie, rechercher des liens entre différents objets, etc. Au-delà de son utilité pour la construction de programmes, l'approche algorithmique (analyser et modéliser les données, identifier les actions à réaliser, les structurer sous la forme d'un algorithme) apparaît donc comme une façon d'aborder et de résoudre une diversité de problèmes. Elle est donc potentiellement utile à tout le monde.

Cette prise de conscience a donné lieu à l'apparition du terme « pensée informatique » – traduction imparfaite de *computational thinking* (Wing, 2006) –, qui est maintenant utilisé dans la plupart des travaux internationaux sur l'enseignement de l'informatique à l'école.

De façon générale, le terme de 'pensée informatique' désigne la façon dont l'informatique amène les informaticiens à aborder la résolution de problème et la conduite de tâches complexes (attention au contre-sens : il ne s'agit pas du tout de suggérer que l'on devrait amener les élèves à penser comme des ordinateurs !). De nombreux travaux ont cherché à cerner les compétences et notions relevant de la « pensée informatique », *cf.* par exemple (Grover & Pea, 2013), (Shute *et al.*, 2017), (Hsu *et al.*, 2018) ou (Tikva & Tambouris, 2021). Il y a un consensus pour considérer qu'il s'agit en tout premier lieu de capacités d'abstraction et de construction d'algorithmes et, notamment : savoir décomposer un problème en sous-problèmes plus simples ; savoir réfléchir à ce qu'il faut faire pour résoudre le problème considéré en termes d'étapes et d'actions ; savoir décrire les problèmes et les solutions à différents niveaux d'abstraction, ce qui permet d'identifier des similitudes entre problèmes et, par suite, de pouvoir réutiliser des éléments de solutions. Comme on le voit, ces compétences sont effectivement générales, et utiles dans de nombreuses situations tant professionnelles que de la vie courante (organiser un voyage, planifier ses actions, etc.). Selon les travaux, le terme « pensée informatique » inclut par ailleurs différentes autres compétences liées à la résolution de problème en général (par exemple, l'évaluation ou la simulation) et/ou des notions ou méthodes plus spécifiques à l'informatique (notion de variable, programmation, test, parallélisation, etc.).

Une revue systématique des pratiques d'enseignement en Europe (Bocconi *et al.*, 2022) montre que, dans presque tous les pays, les raisons avancées pour justifier un enseignement de l'informatique à l'école mettent en avant le renforcement des compétences en résolution de

problèmes et le développement de la logique (qui sont des compétences transversales que l'on pratique en informatique, mais sont utiles beaucoup plus largement), ainsi que la promotion de la programmation (qui est une compétence propre à l'informatique). Viennent ensuite des arguments comme le fait d'attirer les élèves vers des études en informatique ou de renforcer l'employabilité. Enfin, selon les pays, l'enseignement de l'informatique est perçu comme pouvant aider au développement d'autres compétences générales comme le raisonnement, l'analyse, la communication ou encore la créativité.

En France, l'introduction de l'informatique dans les programmes scolaires a fait l'objet de notes et demandes de différentes personnalités et/ou acteurs institutionnels, qui ont notamment avancé que « *comme dans les autres disciplines fondamentales, la sensibilisation précoce aux grands concepts de la science et technique informatique est essentielle. Elle donne des clés aux élèves pour comprendre le monde qui les entoure, elle évite que se forgent des idées fausses et représentations inadéquates, elle fabrique un socle sur lequel les connaissances futures pourront se construire au Collège et au Lycée.* »[1]. A la même époque (2013), l'Académie des sciences a quant à elle mis en avant les arguments suivants : l'informatique est une discipline scientifique clé ; il est nécessaire de comprendre le monde dans lequel on vit ; il y a en France un certain "illettrisme informatique" ; la France est en retard ; les emplois sont nombreux[2].

Il convient donc de dissocier différents types d'arguments dont, notamment :

- Il est important de former des citoyens qui comprennent le monde dans lequel ils vivent et peuvent agir dans/sur celui-ci (ce qui inclut la compréhension des aspects scientifiques et techniques mais, également, éthiques).

- L'enseignement de l'informatique à l'école peut participer au développement de compétences transversales, qui sont mises en jeu en informatique mais ne lui sont pas propres (abstraction, décomposition, synthèse, reconnaissance de structures, résolution de problèmes, analyse de données, modélisation, évaluation). La pratique de l'informatique permettrait donc de faire progresser les élèves de façon générale (ce point est discuté en Section 7).

- L'enseignement de l'informatique à l'école permet de familiariser les élèves avec les notions, principes et processus propres à l'informatique (les notions et techniques utilisées dans les langages de programmation, la façon dont sont représentées et codées les données informatiques, etc.).

Les arguments généralement invoqués pour justifier l'enseignement de l'informatique à l'école sont donc de natures et de portées très différentes. Ainsi, disposer d'une représentation correcte de ce qu'est un algorithme et de comment fonctionne un moteur de recherche quand on lui pose une question permet de comprendre que ces systèmes ne renvoient pas la 'bonne' réponse à cette

[1] Proposition d'orientations générales pour un programme d'informatique à l'école primaire (2013). http://www.epi.asso.fr/revue/editic/itic-ecole-prog_2013-12.htm

[2] Avis de l'Académie des sciences « L'enseignement de l'informatique en France - Il est urgent de ne plus attendre » (2013). www.academie-sciences.fr/pdf/rapport/rads_0513.pdf

question, mais les données qu'un algorithme particulier a identifiées comme pertinentes étant donné les critères qu'il applique et donc, de fait, l'information que les concepteurs ou promoteurs du moteur de recherche ont décidé (pour certaines raisons, légitimes ou pas) de renvoyer. De même, comprendre les principes généraux régissant les 'Intelligences Artificielles' génératives comme ChatGPT permet d'en percevoir les intérêts mais, également, les limites, biais et risques. Il s'agit là de connaissances utiles à tous les élèves en tant que futurs citoyens. Par ailleurs, si l'enseignement de l'algorithmique a un effet positif sur la résolution de problèmes en général, c'est également utile à tous. En revanche, de même que l'enseignement de l'écriture à l'école ne vise pas à former des écrivains (même si certains élèves le deviendront peut-être), l'enseignement de l'informatique à l'école ne peut pas avoir comme but de former de futurs programmeurs. La maîtrise de la programmation et de la conception de programmes est une compétence qui s'acquiert après le bac et nécessite plusieurs années d'étude.

III. Quels objectifs d'enseignement peut-on considérer ?

En raison de la complexité de ce qu'est l'informatique et à la diversité des arguments invoqués en faveur de son enseignement, comprendre les enjeux de cet enseignement nécessite de rentrer dans les détails. L'analyse proposée en Tableau 1, qui s'appuie sur (Tchounikine, 2017) et la littérature scientifique sur le domaine, dresse une liste de différents types d'objectifs potentiels.

Tableau 1 : liste d'objectifs pédagogiques potentiels

Objectifs pédagogiques		Commentaires
autour de la technologie	faire comprendre comment fonctionnent un ordinateur, un réseau d'ordinateurs, des objets ou logiciels d'usage courant comme une page Web, une messagerie électronique, un moteur de recherche, un réseau social, etc.	ce type d'objectif peut être abordé de différentes façons : comme un moyen de démystifier le fonctionnement des technologies (comme une connaissance nécessaire à l'honnête citoyen) ; et/ou comme un support pour enseigner des notions informatiques (représentation des données, processus mis en jeu, etc.)
autour de l'algorithmique et de la résolution de problèmes	faire comprendre la notion d'algorithme	ce type d'objectif peut être abordé en travaillant sur des algorithmes de la vie courante (recette de cuisine, multiplication, accord du participe passé, organisation des actions nécessaires pour se rendre d'une ville à une autre, etc.) et/ou des choses plus spécifiques (résolution d'un problème de mathématiques, simulation d'un phénomène en physique ou en chimie, création d'une animation, etc.)
	faire comprendre les notions utilisées pour construire des algorithmes (notions de séquence d'instructions, de variable, de structure conditionnelle, de boucle, etc.)	
	enseigner les algorithmes classiques (par exemple, comment trier des données ou chercher le meilleur itinéraire de façon efficace)	
	faire pratiquer la construction d'algorithmes (élaborer un algorithme est une compétence très différente de la simple analyse d'algorithmes existants)	
	faire pratiquer une démarche d'analyse de type : identification des objectifs, analyse des données, décomposition en sous-problèmes, écriture des algorithmes puis, éventuellement, programmation et tests	
autour de la programmation	faire en sorte que les élèves sachent programmer dans un langage de programmation, par exemple Scratch ou un langage structuré type 'langage de commande de robot' (avancer, tourner-à-droite, tourner-à-gauche, etc.)	ce peut être l'occasion de réfléchir à la notion de langage (langage naturel, langage informatique) en tant que telle (comme un système de codage, comme un outil de communication, comme un outil de pensée), et donc à l'existence et l'utilité d'une multiplicité de langages
liés à une autre discipline scolaire	faire travailler les élèves sur des problèmes de mathématiques, de physique, de français, de langue_2, etc.	il est possible de construire des séances d'enseignement ayant un double enjeu d'apprentissage (par exemple : mathématiques et informatique) ou d'utiliser l'informatique pour créer une dimension ludique et motivante pour l'autre discipline
autour de la créativité	faire en sorte que les élèves développent leur créativité et sachent exprimer leurs idées créatives (jeux, dialogues, fictions, etc.) en termes de problèmes à résoudre, d'algorithmes et de programmes	ce type d'objectif est très présent dans la vision *étasunienne* de l'enseignement de l'informatique (et, du coup, dans le langage Scratch), beaucoup moins en Europe
autour des usages	faire en sorte que les élèves réfléchissent à des questions de propriété intellectuelle, de traces numériques, de qualité (véracité, précision, etc.) des résultats d'un programme, d'éthique, etc.	ce type d'objectif rejoint les questions d'éducation morale et civique, de prévention des risques, etc.

Il est donc possible de construire des situations pédagogiques visant uniquement à enseigner l'informatique en tant que telle (comme une matière en soi, au même titre que le français ou les mathématiques), mais également d'aborder *via* ou dans le contexte de l'informatique des compétences ou connaissances plus générales (atteindre le but général de l'école de former des citoyens éclairés ; contribuer au fait que les élèves développent des compétences générales/transversales comme l'abstraction, l'organisation de l'action ou la résolution de problème ; contribuer à l'enseignement d'autres disciplines et/ou à certaines compétences du socle commun).

IV. Comment enseigner l'informatique à l'école ?

Outre la nature des objectifs pédagogiques poursuivis (*cf.* Tableau 1), il est possible de considérer l'enseignement de l'informatique selon différents angles d'analyse.

Le premier est celui de la focalisation. Selon les pays, l'informatique est enseignée comme une matière propre ou dans le cadre de l'enseignement d'autres matières. Les deux approches sont cependant généralement hybridées (Bocconi *et al.*, 2022).

Le second est celui de l'approche pédagogique. L'approche la plus pratiquée est de faire travailler les élèves sur des problèmes et/ou des projets nécessitant plusieurs séances (Hsu *et al.*, 2018).

Il est enfin possible de prendre comme angle d'analyse l'utilisation de matériel informatique et de dissocier 'programmation', 'programmation de robots' et 'informatique débranchée'.

- La programmation consiste à proposer aux élèves de construire des programmes (ou : des algorithmes *puis* des programmes) à l'aide d'un langage de programmation comme Scratch. Parmi les avantages : les langages à vocation pédagogique comme Scratch sont conçus pour aider les élèves à construire des algorithmes ; faire exécuter le programme par la machine permet aux élèves de voir si la solution qu'ils proposent fonctionne ; la programmation introduit, pour la plupart des élèves, une dimension ludique motivante. Parmi les inconvénients : il faut disposer d'ordinateurs (ou de tablettes) ; la programmation met l'accent ou, en tout cas, amène à considérer, des dimensions techniques qui peuvent être inutiles et/ou néfastes à l'atteinte des objectifs pédagogiques considérés ; la gestion de la classe est rendue plus compliquée (soucis techniques, excitation des élèves, inquiétude de nombreux enseignants de ne pas savoir répondre aux questions ou problèmes techniques des élèves, etc.).

- La programmation de robots est un cas particulier de programmation : il s'agit de construire un programme qui, en général, pilote le déplacement de petits véhicules automatiques via des instructions comme 'avancer' ou 'tourner à gauche'. Selon les robots, la programmation se fait sur ordinateur (par exemple, avec Scratch) ou directement sur le robot.

- L'informatique débranchée (Bell & Vahrenhold, 2018) vise à faire pratiquer certaines compétences (abstraction, algorithmique, etc.) et à initier à certaines connaissances et techniques (notion de variable ou de boucle, techniques de codage, etc.) sans ordinateur.

Parmi les avantages : cela évite d'avoir à acheter et gérer des ordinateurs ; cela évite que les enseignants aient besoin de savoir programmer et gérer les problèmes techniques (et évite donc également les questions de formation professionnelle que cela soulève) ; en dissociant l'informatique de son substrat technique, cela amène ou, en tout cas, peut aider, à se focaliser sur les principes.

Les arguments avancés en faveur de ces différentes approches sont de natures différentes, et souvent peu étayés. Ainsi, l'approche 'débranchée' bénéficie en France d'une certaine aura, mais les raisons n'en sont pas très claires. Elle a en tout cas été mise en avant par de nombreux acteurs (Main à la pâte, Instituts de Recherche sur l'Enseignement des Mathématiques, etc.). Pour de nombreux informaticiens et groupes de pression informatiques, l'un des facteurs est qu'elle donne une perspective plus scientifique (et moins technique) de l'informatique, ce qui est un enjeu pour la discipline. De même, l'utilisation des robots a été introduite comme présentant un intérêt pédagogique spécifique (caractère concret, tangible, non virtuel des robots), mais sans étayer, au départ du moins, ces arguments. Autre facteur jouant un rôle : comme pour l'équipement des écoles avec des tablettes par exemple, il est facile de communiquer sur l'achat de robots pédagogiques et de valoriser ce type d'action.

L'évaluation scientifique des effets de ces différentes approches est encore une question ouverte. Une étude récente portant sur un ensemble de classes de CM1-CM2 montre de meilleurs résultats pour l'approche 'branchée' (programmation en Scratch) que pour l'approche 'débranchée' et, surtout, que pour l'approche « Scratch+robots » qui, dans ces travaux, présente les moins bons résultats (Sigayret *et al.*, 2022, 2023). D'autres études empiriques ont cependant trouvé des résultats différents, et la question reste donc ouverte.

La question de l'enseignement de l'informatique avec/sans ordinateurs peut également s'envisager en termes de temps d'exposition des enfants/élèves aux écrans, qui est maintenant considéré comme un problème de santé publique (Bousquet-Bérard & Pascal, 2024). Si ne pas faire utiliser d'ordinateurs à l'école diminue de facto ce temps, c'est cependant de façon parfaitement marginale (1 ou 2h certaines semaines ?). Par ailleurs, des travaux récents suggèrent que c'est le contexte d'utilisation des écrans, et non simplement le temps passé devant, qui aurait un effet sur le développement cognitif (Yang *et al.*, 2024). De ce point de vue, l'activité cognitive de résolution de problèmes que développent les élèves lors de séances d'enseignement de l'informatique n'a rien à voir avec la situation passive de visionnage de vidéos. Il est également possible que montrer comment un ordinateur permet de s'engager dans des activités créatives (construire des programmes, des animations, etc.), et n'est pas simplement un robinet d'images consommées passivement, puisse contribuer à de meilleurs usages ; ceci reste cependant à analyser de façon scientifique.

V. Les élèves d'âge scolaire sont-ils en mesure de tirer bénéfice de ces enseignements ?

De très nombreux travaux empiriques ont montré que les élèves de niveau CM1-CM2 pouvaient acquérir des connaissances et compétences en informatique.

Plus spécifiquement, les travaux ont montré que des enseignements couplant informatique et matière-2 (le plus souvent, une matière scientifique : mathématique, physique, chimie, biologie, etc.) permettaient généralement aux élèves de développer des connaissances et compétences en informatique et dans cette matière-2.

Il convient cependant de noter que, s'il est possible de créer des situations pédagogiques permettant d'enseigner ensemble l'informatique et une matière-2, cette façon d'enseigner la matière-2 n'est pas nécessairement plus efficace que celle d'un enseignement sans informatique, et peut même s'avérer moins efficace. La pertinence de ce type de situation doit donc être considérée, au cas par cas, en fonction des objectifs : augmenter la motivation de certains élèves pour matière-2 ; enseigner l'informatique tout en amenant les élèves à mobiliser leurs connaissances en matière-2, avant ou après un enseignement de matière-2 spécifique ; etc.

VI. La pratique de la pensée informatique améliore-t-elle les résultats des élèves dans les autres disciplines ?

Ainsi qu'indiqué en Section 3, l'hypothèse selon laquelle la pratique de la pensée informatique fait progresser les élèves de façon générale et, notamment, sur des compétences comme l'abstraction et la résolution de problèmes, est l'une des raisons souvent invoquées pour justifier l'enseignement de l'informatique à l'école, et l'un des enjeux potentiels de cet enseignement.

Cette idée est extrêmement ancrée chez certains acteurs. Elle relève cependant le plus souvent d'une croyance, similaire à la façon dont, en d'autres temps, il était affirmé que l'enseignement du latin avait un effet bénéfique général. Cela ne veut pas dire que cette idée est erronée, mais qu'elle demande confirmation.

Historiquement, le fait que la pratique de la pensée informatique puisse avoir un effet positif sur les apprentissages mathématiques est à la source des travaux fondateurs de Papert (1980).

Les travaux empiriques menés dans les années 80 et 90 se sont cependant révélés globalement décevants. Ils suggèrent que, en général, les compétences que développent les élèves lors d'activités informatiques ne se transfèrent pas à leurs autres activités (Denning & Tedre, 2019).

La relance par Wing (2006) de l'idée selon laquelle la pensée informatique est une compétence générale qui, à ce titre, devrait être enseignée à tous, ne repose pas sur des études ayant démontré que, contrairement à ce que suggéraient les travaux des années 80-90, la pensée informatique a bien un impact bénéfique pour les élèves.

D'un point de vue scientifique, la question reste donc ouverte. La notion de 'digital native' est une invention journalistique, et les mécanismes cognitifs des élèves (et donc leurs capacités de 'transfert') n'ont pas évolué depuis les années 80. En revanche, il est possible que la façon dont on a abordé la question à cette époque, et/ou le peu d'ergonomie des moyens informatiques de l'époque, aient pesé sur les résultats. La compréhension des principes structurant la façon dont l'informatique amène à aborder la résolution de problèmes, ainsi que les moyens de faire pratiquer l'informatique aux élèves, ont profondément évolués depuis les années 80. Il est donc légitime de reconsidérer la question.

L'une des difficultés des travaux actuels est qu'il y a différentes approches et/ou interprétations de la question de l'apport de la pratique de l'informatique à d'autres disciplines.

L'analyse de situations d'enseignements couplant informatique et matière-2 montre généralement une progression des élèves en informatique et en matière-2 (Scherer et al., 2019). Cependant, dans ce type de situation, il est difficile de déterminer ce qui relève de l'effet spécifique de la dimension informatique et ce qui relève du fait que la construction du programme amène les élèves à réfléchir sur la matière-2 (au même titre que le ferait une situation de résolution problème en matière-2 non liée à l'informatique).

Il existe cependant quelques travaux montrant qu'une pédagogie adaptée peut avoir un effet sur le transfert de compétences développées dans le cadre d'activités informatiques (Hutchins *et al.*, 2020). Par ailleurs, des effets sur des mécanismes cognitifs généraux comme la planification (par exemple, la capacité à organiser un plan d'action permettant de réaliser un objectif) ou l'inhibition (par exemple, la capacité à contrôler des réponses impulsives) ont également été mis en évidence (Arfé *et al.*, 2020).

Pour résumer, il est clair que l'algorithmique et de la programmation amènent les élèves à s'engager dans des activités de résolution de problème, et à le faire avec de nouveaux outils conceptuels (notion de boucle, etc.) et techniques. Ceci est positif en soi (à moins, bien sûr, que cela ne soit fait au détriment d'autres choses plus importantes). Il est peu probable que ce type d'activité n'ait aucun effet sur les capacités générales de résolution de problèmes des élèves et/ou sur des compétences transverses comme l'abstraction. Il ne faut cependant pas tenir pour acquis que les connaissances et compétences travaillées dans le cadre d'activités pédagogiques en informatique vont se transférer vers d'autres contextes.

Il convient de noter que le fait que les compétences acquises dans un domaine-1 ne se transfèrent pas à un domaine-2 (ou que l'on n'arrive pas à le montrer) n'est pas spécifique à l'informatique. L'analyse de ce type de situation pose des questions extrêmement délicates, tant d'un point de vue théorique (compréhension des mécanismes cognitifs sous-jacents) que méthodologique. Ainsi, alors que les mesures de l'effet immédiat d'un enseignement d'une matière-1 sur les résultats des élèves dans une matière-2 sont souvent décevants, une analyse plus large peut montrer des résultats plus positifs (Bransford & Schwartz, 1999) : l'enseignement de matière-1 peut avoir semé des graines qui ne sont pas perceptibles immédiatement mais qui, combinées à d'autres, peuvent avoir un effet par la suite.

Conclusion

Après avoir posé un cadre général, ce document a présenté un certain nombre d'axes de réflexion. Ils doivent être abordés en gardant en tête les éléments suivants :

- Étant donné la place qu'elle a prise dans la société, les élèves de l'école primaire sont déjà, en toute hypothèse, en contact avec l'informatique. La question n'est donc pas d'initier les élèves à l'informatique ou au contraire de les en préserver. Il s'agit, plus fondamentalement, de considérer la place et le rôle de l'école dans la perception que développent les élèves de l'informatique, leur compréhension des technologies informatiques (comment elles fonctionnent mais, également, leurs limites ou leurs biais), les questions liées aux usages (dimensions éthiques, risques, impacts sur les individus et les groupes, etc.), ou encore les bénéfices généraux qu'ils peuvent en retirer (par exemple en résolution de problème).

- Aborder les questions relatives à l'enseignement de l'informatique à l'école nécessite de définir précisément ce dont on parle. Il est extrêmement fréquent que, lors de discussions sur ce sujet, les interlocuteurs pensent se comprendre mais, en fait, fassent référence à des choses très différentes. Une attention toute particulière est donc nécessaire.

- Les travaux scientifiques sur l'enseignement et la pratique de l'informatique à l'école sont extrêmement nombreux. Cependant, pour différentes raisons (dont, notamment : la complexité du domaine, le manque de conceptualisations de référence comme on peut en avoir pour les autres domaines scientifiques, la complexité des situations pédagogiques et de leur évaluation, le prosélytisme –ou, au contraire, l'opposition de principe– de certains acteurs), ces travaux sont souvent assez imprécis et/ou confus. La question (évoquée en Section 7) du transfert vers d'autres disciplines des compétences acquises via l'informatique en est un bon exemple. Là encore, il convient donc de faire attention aux sources que l'on utilise, et à ne pas faire de généralisations abusives.

- L'un des écueils de l'enseignement de l'informatique à l'école primaire est la formation des enseignants. Comme ils n'ont pour la plupart jamais étudié l'informatique dans leur parcours de formation initiale (et ne l'ont pas non plus rencontrée comme élèves), ils en ont souvent une perception par défaut qui est celle de l'utilisateur. Par ailleurs, ils ont de bonnes raisons de s'en méfier et/ou d'exprimer des réticences à l'aborder : outre les problèmes de formation il faut disposer de machines, résoudre les difficultés techniques qui ne manqueront pas de se présenter, gérer l'excitation des élèves, ou encore accepter d'être mis en défaut par un problème ou une question hors de son domaine de compétence. Il n'y a pas besoin d'avoir fait des études en informatique ou même des études scientifiques pour enseigner l'informatique à l'école. Il faut cependant que les formations des enseignants soient ciblées et, notamment, abordent des situations d'enseignement précises.

Références

Arfé, B., Vardanega, T. & Ronconi, L. (2020). The effects of coding on children's planning and inhibition skills. *Computers & Education*, 148, 103807.

Bell, T., & Vahrenhold, J. (2018). CS unplugged—how is it used, and does it work?. In H. Räcke & A. Srinivasan (Eds.), *Adventures between lower bounds and higher altitudes: Essays dedicated to Juraj Hromkovič on the occasion of his 60th birthday* (pp. 497-521). Lecture Notes in Computer Science.

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M. A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V., & Stupurienė, G. (2022). *Reviewing Computational Thinking in Compulsory Education*. Publications Office of the European Union. https://doi.org/10.2760/126955

Bousquet-Bérard, C. & Pascal, A. (2024). *Enfants et écrans. À la recherche du temps perdu*. Accessible en ligne à https://www.vie-publique.fr/files/rapport/pdf/293978.pdf

Bransford, J. D. & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. Review of research in education, 24(1), 61-100.

Dares (2023). Les tensions sur le marché du travail en 2022. Dares Résultats, n° 59.

Denning, P. J. & Tedre, M. (2019). *Computational thinking*. MIT Press.

Grover, S. & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, 42(1), 38–43.

Hsu, T. C., Chang, S. C. & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, 126, 296-310.

Hutchins, N., Biswas, G., Wolf, R., Chin, D., Grover, S. & Blair, K. (2020). Computational Thinking in Support of Learning and Transfer. In Gresalfi, M. and Horn, I. S. (Eds.), *The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS), Volume 3 (pp. 1405-1412*). International Society of the Learning Sciences.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

Scherer, R., Siddiq, F. & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology*, 111(5), 764–792.

Sciberras, J.-C., Bernier, A., Doyen, D., Grégoire, A., & Rieutort, L. (2022). *Les métiers en 2030. Rapport du groupe Prospective des métiers et qualification*, Rapport France Stratégie-DARES (Direction de l'animation de la recherche, des études et des statistiques).

Shute, V. J., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. E*ducational research review*, 22, 142-158.

Sigayret, K., Tricot, A. & Blanc, N. (2022). Unplugged or plugged-in programming learning: A comparative experimental study. *Computers & Education*, 184, 104505. https://doi.org/10.1016/j.compedu.2022.104505

Sigayret, K., Blanc, N. & Tricot, A. (2023). Comparing three different approaches to teach programming and computational thinking in 5th grade. K. Belotti, M. Bergey, A. de Borman, & P. Dessus (Eds.), *27th Conference of the JUnior REsearchers of EARLI 2023* (pp. 184, 104505). EARLI-European Association for Research on Learning and Instruction. Experimental study. *Computers & Education*. (https://doi.org/10.1016/j.compedu.2023.104505)

Tchounikine, P. (2017). *Initier les élèves à la pensée informatique et à la programmation avec Scratch*. https://lig-membres.imag.fr/tchounikine/PenseeInformatiqueEcole.html

Tikva, C. & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, 162, 104083.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.

Yang, S., Tandon, P., Lelong, N., Pry, R., Charles, M.-A., & Heude, B. (2024). Associations of screen use with cognitive development in early childhood: The ELFE birth cohort. *Journal of Child Psychology and Psychiatry*, *65*(5), 680-693.