Opinions Libres
le blog d'Olivier Ezratty

# Comprendre l'informatique quantique – complexité

Dans la **partie précédente** de cette série estivale sur l'informatique quantique, nous avons fait le tour des principaux algorithmes quantiques connus, de leurs domaines d'applications et de leur performance relative.

Le calcul quantique est parfois présenté comme étant une solution miracle aux limites du calcul sur supercalculateurs. Il permettrait de résoudre des problèmes dits "intractables" sur des ordinateurs classiques. Mais au juste, quelle est la nature des problèmes qui peuvent être résolus avec un ordinateur quantique et qui ne peuvent pas l'être avec des ordinateurs classiques ? Et surtout, quelles sont les limites des ordinateurs quantiques ? Comment se situent-elles par rapport aux limites de l'intelligence artificielle ?

Nous allons voir que ces limites sont plutôt floues et mouvantes. Elles sont traitées dans un champ complet et méconnu de la science, celui des **théories de la complexité**. C'est un monde on ne peut plus abstrait où les spécialistes parlent un langage abscons fait de P, NP, BQP et autres complétudes. Ils gambergent depuis près d'un demi-siècle pour déterminer si **P = NP ou pas**, une question aussi importante que le rôle exact du nombre 42 dans le fonctionnement de l'Univers. C'est la science des classes de complexité de problèmes. Derrière ces mathématiques de la complexité se cachent des considérations techniques mais aussi philosophiques fondamentales pour l'Homme et son désir de toute puissance.

Les classes de complexité de problèmes sont des poupées russes plus ou moins emboitées les unes dans les autres. Elles tournent surtout autour de la question de la montée en charge du temps de résolution des problèmes en fonction de leur taille.

On ne sait résoudre dans un temps raisonnable que les problèmes dits polynomiaux ou plus simples que les polynomiaux. Un temps polynomial est proportionnel à une puissance donnée de N, N étant la dimension du problème à résoudre. L'informatique quantique permet dans certaines conditions de résoudre certains problèmes dits exponentiels, qui croissent de manière exponentielle avec leur taille. Au-delà se situent divers problèmes inaccessibles qui relèvent souvent de simulations complexes ou de résolutions par force brute. Bref, les ordinateurs quantiques ne pourront pas résoudre tous les problèmes qui nous passeront par la tête, même le jour où l'on pourra aligner des gazillions de qubits avec un taux d'erreur infinitésimal.

Ces limites ont un impact indirect sur les prévisions concernant la création d'intelligences artificielles omniscientes capables de transcender le raisonnement humain et de résoudre tous les problèmes. Ces hypothétiques AGI (Artificial General Intelligence) seront limitées par les données et concepts qui les alimentent et par l'impossibilité de résoudre certains problèmes complexes, notamment ceux qui relèvent de la prévision et de la simulation et qui reposent sur la force brute plutôt sur des astuces algorithmiques permettant d'aboutir rapidement à la solution.

Le calcul ultime n'existe donc pas encore et l'Homme continuera à faire face à l'impossible et ne pourra pas résoudre tous les problèmes complexes qu'il rencontrera ! Bref, le calcul quantique ne permet pas de dominer la nature et de mettre en équation l'Univers et d'en prévoir le fonctionnement au quantum près. Le hasard,

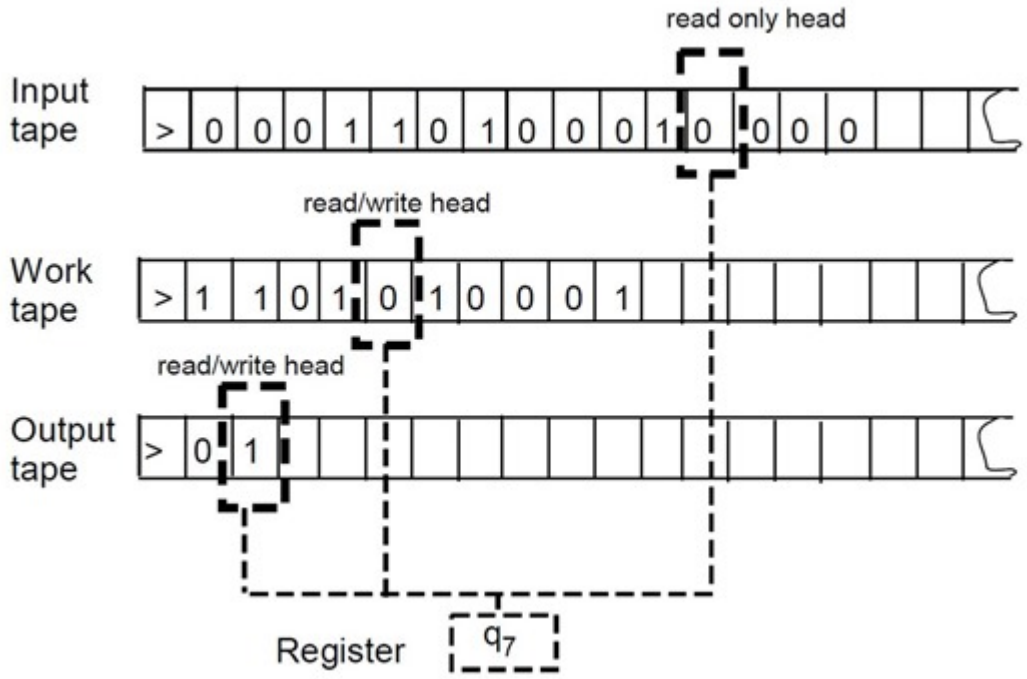
l'imprévu et le libre arbitre continueront de jouer un rôle dans un monde très indéterministe et c'est tant mieux comme cela. C'est une petite leçon d'humilité pour l'Homme que de passer son temps à découvrir des limites scientifiques à ses besoins de contrôle !

**Classes de complexité de problèmes**

Pour rentrer dans ce sujet, il faut en passer par définir les grandes classes de problèmes par niveau de complexité. Elles font partie d'un champ entier de la logique et des mathématiques qui agite un petit monde de spécialistes. Me voilà donc une fois encore amené à devoir simplifier la complexité, et cette fois-ci, au sens littéral du terme.

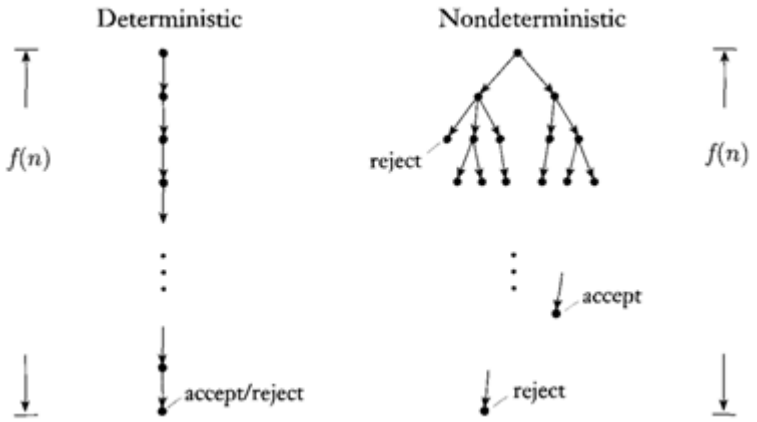
Dans la pratique, les classes de complexité décrivent des problèmes que l'on résout par force brute en testant plusieurs combinatoires. Ils ne relèvent pas d'équations mathématiques simples que l'on résout avec des formules permettant d'aboutir directement à la solution comme on le fait pour prédire la position des planètes en s'appuyant sur les lois de la mécanique newtonienne.

Les classes de problèmes font appel à une notion de machines déterministes et non déterministes de Turing. De quoi s'agit-il ? Les machines de Turing sont les modèles conceptuels d'ordinateurs créés par Alan Turing avant la seconde guerre mondiale. Elles modélisent les traitements informatiques en s'appuyant sur la notion de programmes et de données, incarnées par des rouleaux de papier continus, le premier pour le programme, le second pour les données en entrée et le troisième pour générer les résultats (schéma *ci-dessous*, **source**). Des étudiants d'un master de l'ENS Lyon ont réalisé une machine de Turing en Lego en 2012 à l'occasion du centenaire de la naissance d'Alan Turing (**vidéo**) et ce n'était pas la seule du genre (**vidéo**) ! Un Anglais en a aussi réalisé une en bois en 2015 (**vidéo**). Voilà de quoi s'éduquer ludiquement !




Le modèle théorique de Turing est utilisé depuis longtemps pour définir les classes de problèmes que l'on peut résoudre ou pas avec un ordinateur. Les ordinateurs sont tous métaphoriquement des machines de Turing, reproduisent cette logique en lisant des instructions de programmes et en gérant les données en mémoire vive (RAM) ou en stockage persistent (disque dur, SSD, …). Est associée à la notion de machine de Turing celle de la **thèse de Church-Turing**, des noms d'Alonzo Church et Alan Turing selon laquelle il existe une équivalence entre problèmes de calcul réalisable à la main et avec des ressources non limitées, ceux qui sont traitables avec une machine de Turing et les ceux qui peuvent être résolus avec des fonctions dites récursives.

Dans une machine déterministe, la séquence des actions à réaliser est prédéfinie et séquentielle. Dans le modèle conceptuel de machine de Turing non déterministe, les règles de calcul peuvent imposer de réaliser plusieurs opérations différentes pour chaque situation évaluée. En gros, en explorant plusieurs voies en parallèle et en cherchant une réponse positive à une composante d'algorithme et en fermant des boucles de tests parallèles une fois les sous-solutions trouvées.

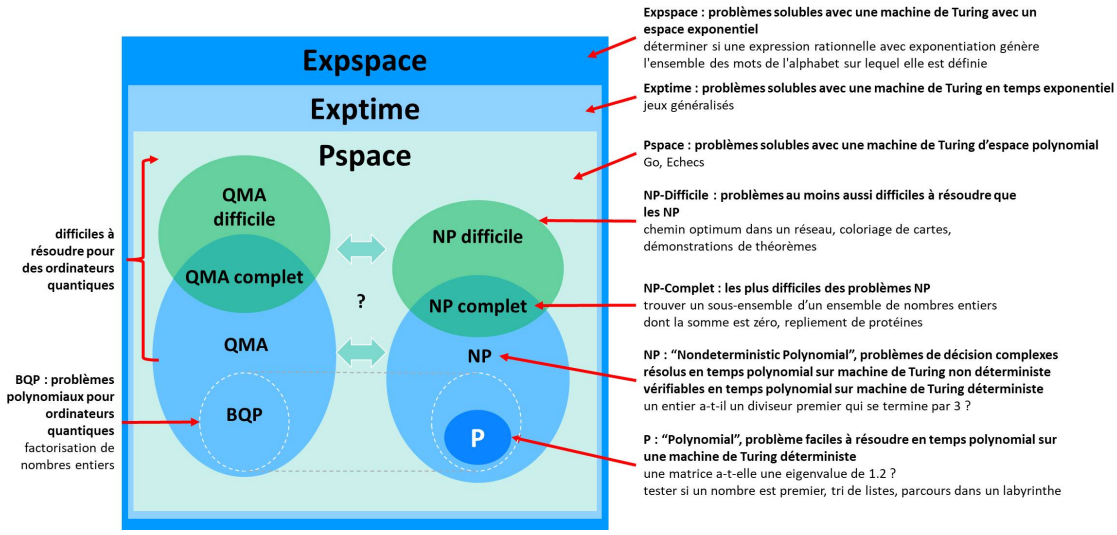



C'est plus ou moins le modèle de navigation dans l'arbre de décision de la version 2017 d'AlphaGo, dite **AlphaGo Zero**, qui évalue dans un réseau de neurones différents scénarios de jeu. Bref, une machine non déterministe augmente la combinatoire de calcul par rapport à une machine déterministe. Et cette combinatoire passe de polynomiale à exponentielle. La force d'AlphaGo Zero est d'utiliser des réseaux de neurones en quelque sorte récursif pour réduire le nombre de branches de l'arbre de décision à tester pour sortir partiellement de la fatalité exponentielle de ce jeu.

**Les classes de complexité génériques**

Le niveau de complexité s'entend vis à vis du temps de calcul nécessaire et de l'espace mémoire nécessaire pour ces calculs. En règle générale, on est bloqué par le temps de calcul avant de l'être par la capacité mémoire. L'association d'un problème à une classe de complexité est liée à la performance du meilleur algorithme connu pour résoudre ledit problème.

Les niveaux de classes de problèmes dans les théories de la complexité reposent souvent sur des modèles de boite noire ou d'oracles à qui un système pose des questions et obtient des réponses en fonctions des données fournies. C'est une logique de "force brute" et de scan d'hypothèses. La combinatoire à tester est plus ou moins grande selon les classes de problèmes.



Voici donc ces classes par niveau croissant de complexité sachant que nous passerons le plus de temps sur les classes NP et NP Complet.

**L** : ou LSPACE, ou DLOGSPACE, qui définit la classe des problèmes que l'on peut résoudre à une échelle logarithmique de mémoire consommée et sur une machine de Turing déterministe. Bref, sur un ordinateur traditionnel. C'est le genre de classe idéale ! La complexité de calcul diminue rapidement avec la taille du problème. Malheureusement, très peu de problèmes complexes sont dans cette classe là. On y trouve notamment les requêtes dans des bases de données relationnelles préalablement indexées, les recherches de séquences d'ADN et d'une manière générale les techniques de recherche utilisant des pointeurs et qui optimisent l'utilisation de la mémoire des ordinateurs.

**NL** : la classe des problèmes résolus à une échelle logarithmique sur une machine non-déterministe. Les logiciens cherchent toujours à savoir si si L=NL ! Cela pourrait durer quelque temps. Cela les occupe moins que P=NP.

**P** : qui définit les problèmes que l'on peut résoudre avec un temps qui croit de manière polynomiale avec le nombre de données à traiter et sur une machine déterministe. Si N est la taille du problème, la durée de traitement est proportionnelle à N puissance M, M étant un entier, si possible 2. C'est un problème facile à résoudre. Il est dit "tractable". Cela comprend le tri de listes, la validation de l'existence d'un chemin dans un graphe, la recherche d'un chemin minimum dans un graphe, la multiplication de matrices ou l'évaluation d'un nombre pour savoir s'il est premier.

**BPP** : est une classe de problèmes qui peuvent être résolus par des approches aléatoires ("Bounded-Error Probabilistic Polynomial-Time"). Il semblerait que BPP=P mais ce n'est pas encore démontré.

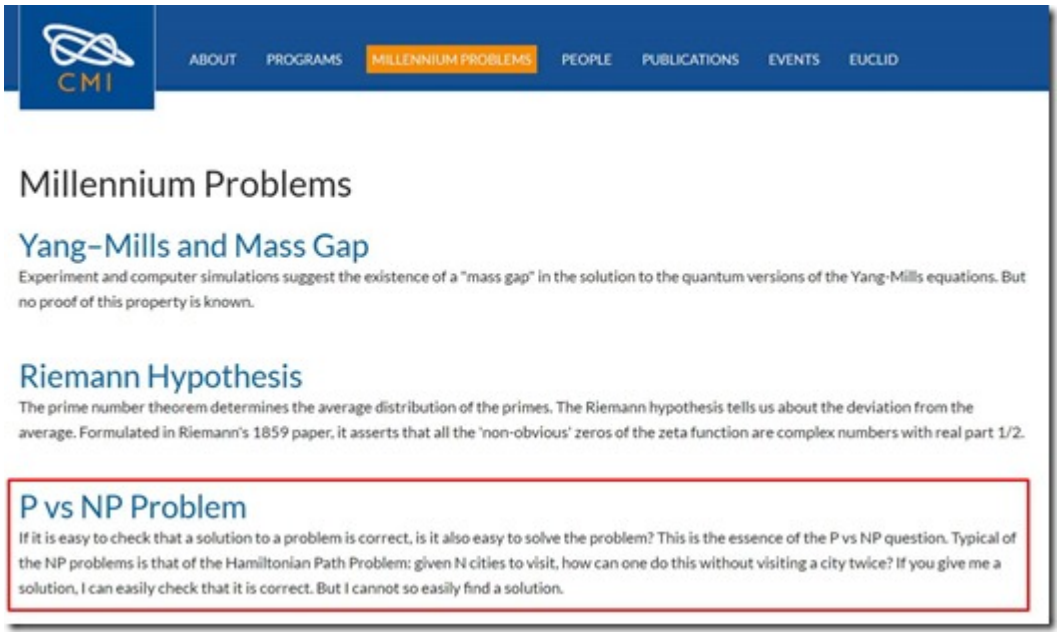
**NP** : décrit la classe des problèmes dont il est facile de vérifier la validité d'une solution, à savoir que celle-ci peut être réalisée en un temps polynomial par une machine déterministe. L'autre définition de la classe est qu'elle contient les problèmes dont le temps de résolution est polynomial sur une machine non déterministe. Ces problèmes plus complexes on un temps de calcul au minimum exponentiel lorsque la méthode utilisée est dite naïve, pour tester toutes les hypothèses possibles. Ils sont dits intractables. En pratique, ce sont des problèmes particulièrement adaptés aux ordinateurs quantiques du fait de leur capacité à évaluer en parallèle 2 puissance N combinaisons.

Quelques exemples de problèmes NP : l'arbre de Stein pour déterminer si un réseau électrique permet de relier un nombre de maison à un certain prix, vérifier qu'une séquence d'ADN se retrouve dans plusieurs gênes et la distribution de tâches à différents agents pour minimiser le temps de leur réalisation. Les exemples théoriques des cours de complexité ont l'air de relever de problèmes futiles, nous en verrons quelques-uns plus tard. Mais côté métiers, ces problèmes ont des équivalents très concrets dans la logistique, la planification, la production, les transports, les télécoms, les utilities, la finance ainsi que dans la cryptographie.

A noter qu'un problème "décidable", c'est à dire qui requiert d'explorer un espace fini d'options, n'est pas forcément faisable d'un point de vue pratique. Même s'il peut être résolu en un temps fini, sa résolution peut prendre un temps trop long. Un problème exponentiel a une solution élégante si on peut en trouver une qui ait une durée polynomiale voir, dans le meilleur des cas, linéaire. Les temps polynomiaux *scalent* mieux que les temps exponentiels !

A noter qu'un problème décidable n'est pas forcément faisable d'un point de vue pratique. Un problème décidable requiert d'explorer un espace fini d'options. Cela peut prendre un temps très long mais c'est un temps fini. Mais s'il est trop long à traiter, il n'est pas faisable, avec les techniques de calcul à un moment donné. Un problème exponentiel a une solution élégante si on peut en trouver une qui ait une durée polynomiale voir, dans

le meilleur des cas, linéaire. Les temps polynomiaux scalent mieux que les temps exponentiels !



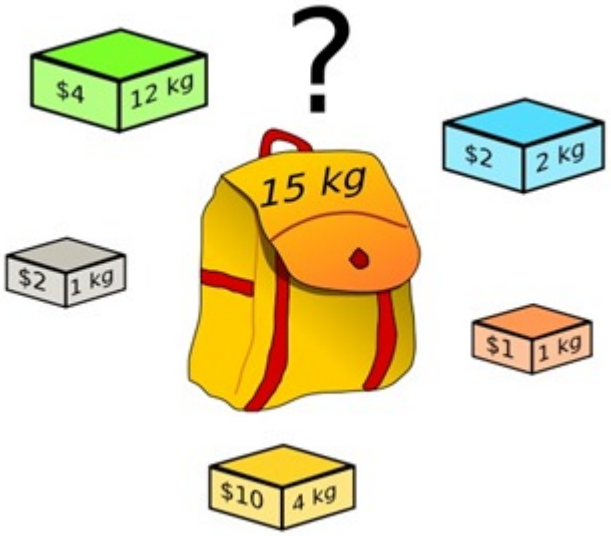

Un gros débat a cours depuis 1956 (Kurt Gödel) pour savoir si la classe P égale la classe NP. Si P = NP, il serait aussi simple de trouver un résultat quand on sait aussi le vérifier simplement. Le consensus général est que P n'est pas égal à NP. La démonstration de P<>NP ou de son contraire **fait partie** de l'un des **sept défis mathématiques** du Clay Mathematics Institute lancés en 2000 chacun dotés d'un prix de $1M (*ci-dessus*). Parmi ces défis, on trouve la démonstration des équations de Navier-Stokes sur la mécanique des fluides et celle de l'hypothèse de Riemann sur la distribution des nombres premiers. Voilà de beaux problèmes à résoudre pour une hypothétique AGI (Artificial General Intelligence) capable de dépasser l'Homme dans sa capacité de conceptualisation. Et $7M à la clé, si le principe de l'AGI était vérifié, à savoir sa capacité à résoudre n'importe quel problème, à supposer que celui-ci soit décidable ! Le chercheur brésilien André Luiz Barbosa a publié en 2010 **P ? NP Proof** (25 pages) tout comme un papier invalidant le théorème de Cook selon lequel un problème booléen SAT est NP-Complet, **The Cook-Levin Theorem is False**, 2010 (11 pages). Il ne fait visiblement pas l'unanimité, ses travaux n'étant ni cités, ni repris.

Du côté P vs NP, la **formulation du défi à relever** donne un exemple d'un tel problème : vous devez allouer 50 chambres de deux étudiants à 400 candidats mais certains candidats ne doivent pas cohabiter dans la même chambre. La combinatoire de choix des 100 étudiants parmi 400 est monstrueusement énorme, donc le problème n'est pas traitable facilement avec un supercalculateur et avec de la force brute. C'est bien un problème NP car une solution donnée est facile à vérifier car il suffit de vérifier qu'aucune des chambres ne contient une paire d'individus interdite. C'est un peu la théorie du tout ou du rien car si P = NP, tous les problèmes NP ont une solution efficace polynomiale. Si P ? NP, aucun des problèmes NP n'a de solution efficace.

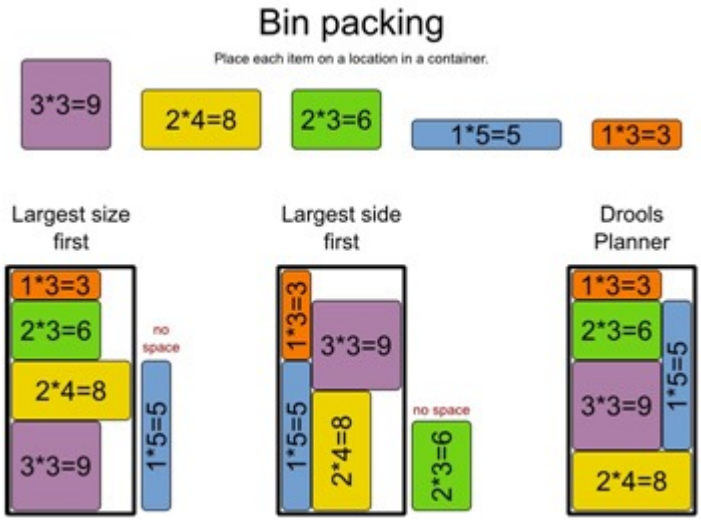
La définition des classes de problèmes NP et NP-Complet est relativement récente. Elle est issue de **The complexity of theorem-proving procedures** de Stephen Cook de l'Université de Toronto, 1971 (8 pages), mieux vulgarisé dans **An overview of computational complexity** (8 pages) et **Reducibility about combinatorial problems**, de Richard Karp, 1972 (19 pages) ainsi que dans **Complexité et calculabilité** d'Anca Muscholl du LaBRI, 2017 (128 slides). J'ai parcouru un grand nombre d'autres publications pour m'y retrouver mais elles étaient assez redondantes dans l'ensemble.

**NP Complet** : ils se définissent selon Richard Karp comme les problèmes dans lesquels les autres problèmes NP peuvent être réduits de manière polynomiale. A fortiori, ils n'ont pas de solution P (polynomiale) connue.

Ils sont encore non accessibles aux ordinateurs quantiques. C'est dans cette classe que l'on trouve les problèmes de logique booléenne de type SAT ou 3SAT dont je vous passe les détails car je risquerai de vous perdre et de me perdre par la même occasion ! Plus de 3000 problèmes NP-Complets sont identifiés à ce jour (**liste**).

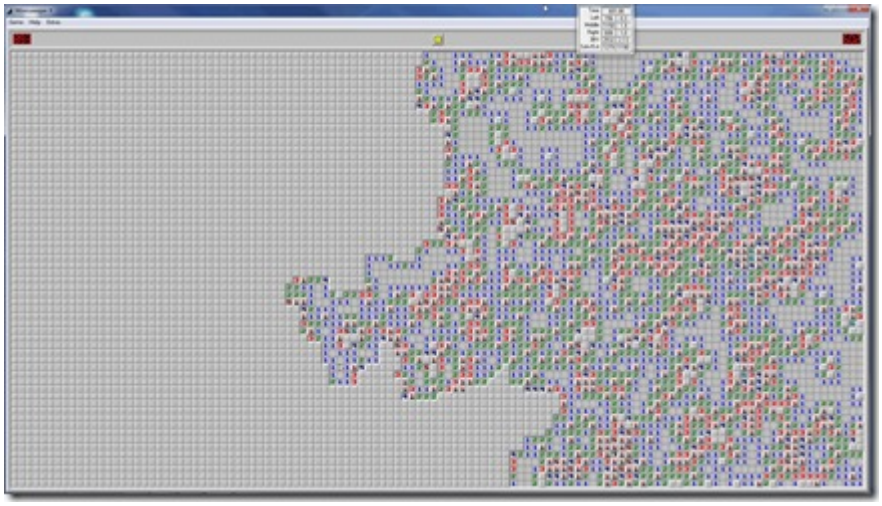



On y trouve notamment le problème du remplissage optimum du coffre de la voiture lorsque l'on part en vacances ou lorsque l'on revient du Noël dans sa famille avec une flopée de cadeaux. Et puis le problème du sac à dos consistant à le remplir de manière optimale avec un jeu d'objets, pour obtenir la plus grande charge et sans dépasser un poids maximum ("Bin packing").




Il comprend aussi le problème de la somme de sous-ensemble consistant à trouver un sous-ensemble d'un ensemble de nombres entiers dont la somme est égale à un entier arbitraire.

Il y a aussi le problème du démineur consistant à localiser des mines cachées dans un terrain avec pour seules indications le nombre de mines dans les zones adjacentes et le nombre de mines total dans le champs. Le tout évidemment, sans les faire exploser. C'est un jeu bien connu des utilisateurs de Windows, lancé en 1989 !



Il semblerait enfin que la simulation du repliement de protéines complexes soit un problème NP-Complet. Cf **Is protein folding problem really a NP-complete one ? First investigations**, 2013 (31 pages). Ce serait donc un problème potentiellement très difficile à résoudre avec un ordinateur quantique avec de grandes protéines.

Il est démontré que si l'on trouvait une solution optimale à un problème NP-Complet, on trouverait toutes les solutions aux problèmes de cette classe. C'est la notion importante de réduction de problèmes.

Le coloriage de graphes avec des couleurs différentes pour les nœuds, les branches ou les surfaces fait partie des problèmes NP, NP-Complet et NP-Difficile selon les cas. Les deux premiers cas nécessitant un nombre de couleurs dépendant du nombre maximum de connexions entre éléments du graphe et le dernier cas, relevant du coloriage de cartes dans des couleurs adjacentes différentes qui n'en nécessite que quatre au maximum, grâce à la démonstration informatique du théorème des quatre couleurs en 1976 par Kenneth Appel et Wolfgang Haken.

- Le coloriage de **nœuds** de graphes a des applications dans le placement d'antennes mobiles et dans l'allocation de registres mémoires pour un compilateur. Le problème est NP-Complet pour sa résolution et NP-Difficile pour trouver sa solution optimale.

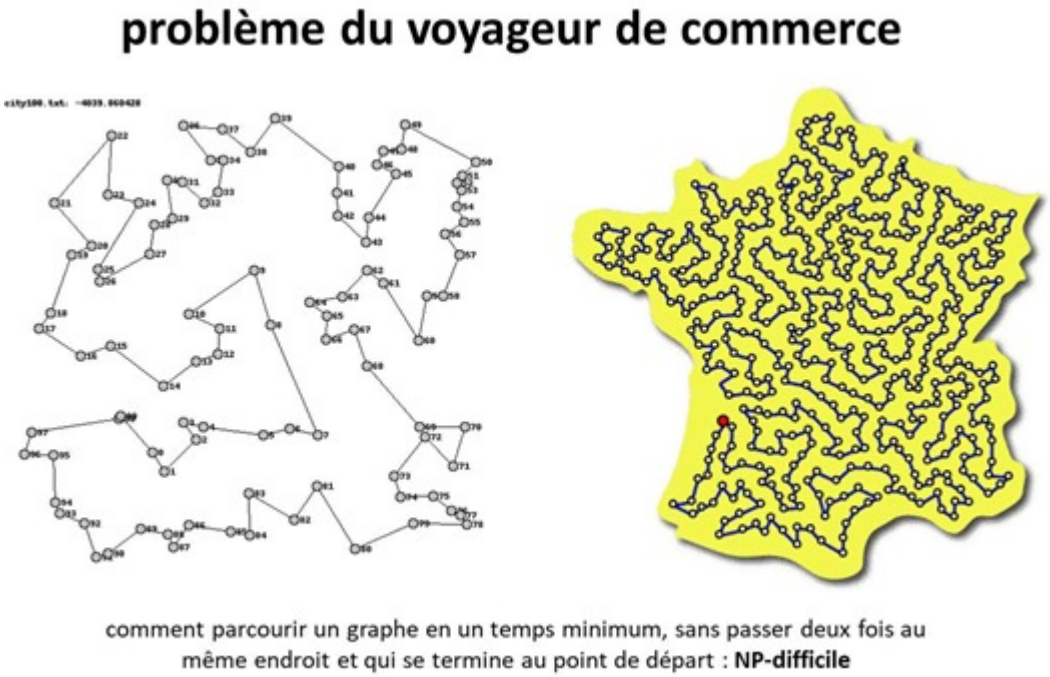
- Celui des **branches** a des applications dans l'allocation de fréquences de réseaux de fibres optiques multimodes. Il permet aussi d'optimiser le placement d'objets ou personnes en fonction de leur compatibilité ou incompatibilités. Le coloriage optimum est un problème NP-Difficile.

- Celui des **zones** peut servir à définir les zones de couvertures d'antennes radio mobiles ou de satellites de télécommunications. Il peut même servir à allouer les fréquences micro-ondes d'activation de qubits supraconducteurs. Le coloriage avec trois couleurs est un problème NP-Complet.

D'une manière générale, de nombreuses classes de problèmes C ont une sous-classe C-Complet et C-Difficile. Un problème est C-Difficile s'il existe un type de réduction des problèmes de la classe C vers ce problème. Si le problème C-Difficile fait partie de la classe C, alors il est dit "C-Complet".

Pour en savoir plus, voir notamment Complexity Theory **Part I** (81 slides) et **part II** (83 slides), qui fait partie d'un **cours de Stanford sur les théories de la complexité**, **Calculabilite et Complexite – Quelques resultats que je connais** d'Etienne Grandjean de l'Université de Caen, 2017 (43 slides) ainsi que cette **vidéo** d'Olivier Bailleux (2017, 20 minutes). Elle est très claire mais vous serez très forts si vous arrivez à suivre et à tout assimiler ! L'auteur précise qu'il faut la regarder plusieurs fois en faisant souvent une pause !

**NP Difficile** : concerne les problèmes d'optimisation où l'on recherche un minimum ou un maximum avec une grande combinatoire. Un problème est NP-Difficile si tous les problèmes NP-Complets peuvent se réduire par simplification polynomiale à ce problème. C'est le cas de la résolution du problème du voyageur du commerce où l'on doit tester une grande combinatoire de parcours pour trouver celui qui est réalisé le plus rapidement pour passer via un nombre déterminé de villes. Il faut alors tester toutes les solutions.




comment parcourir un graphe en un temps minimum, sans passer deux fois au même endroit et qui se termine au point de départ : **NP-difficile**

Si un voyageur de commerce doit traverser 125 villes en moins de 30 jours, s'il existe une solution qui fonctionne dans ce laps de temps, alors le problème est NP. Mais rien ne dit que l'on a trouvé toutes les solutions ! La résolution du problème en-dessous d'un temps de parcourt arbitraire avec un retour au point de départ est un problème NP-complet. C'est ce que l'on appelle un circuit hamiltonien : un chemin parcourant un graphe passant une fois et une seule par chacun des nœuds et revenant à son point de départ. La détermination du temps de parcours le plus court est NP-difficile. L'algorithme de force brute pour le résoudre a un temps qui dépend de N! , N étant le nombre de nœuds du réseau. Le temps optimum connu est N^2 * 2^N. Le problème est difficile à résoudre au-delà de 20 étapes ! Cf le site **The Traveling Salesman Problem** qui donne quelques exemples de tels problèmes comme le parcours de tous les 49 687 pubs anglais ou des 49 603 lieux touristiques aux USA.

La classe des problèmes NP-Difficile contient aussi nombre de jeux de **Nintendo** comme Super Mario Bros, La Légende de Zelda et Pokemon. Cf **Classic Nintendo Games are (Computationally) Hard**, 2012 (36 pages).

**PSPACE** : est la classe des problèmes qui peuvent être résolus en espace polynomial sur une machine déterministe. NPSPACE est la classe des problèmes pouvant être résolus en espace polynomial sur une machine non déterministe. NPSPACE = PSPACE selon le **théorème de Savitch**.

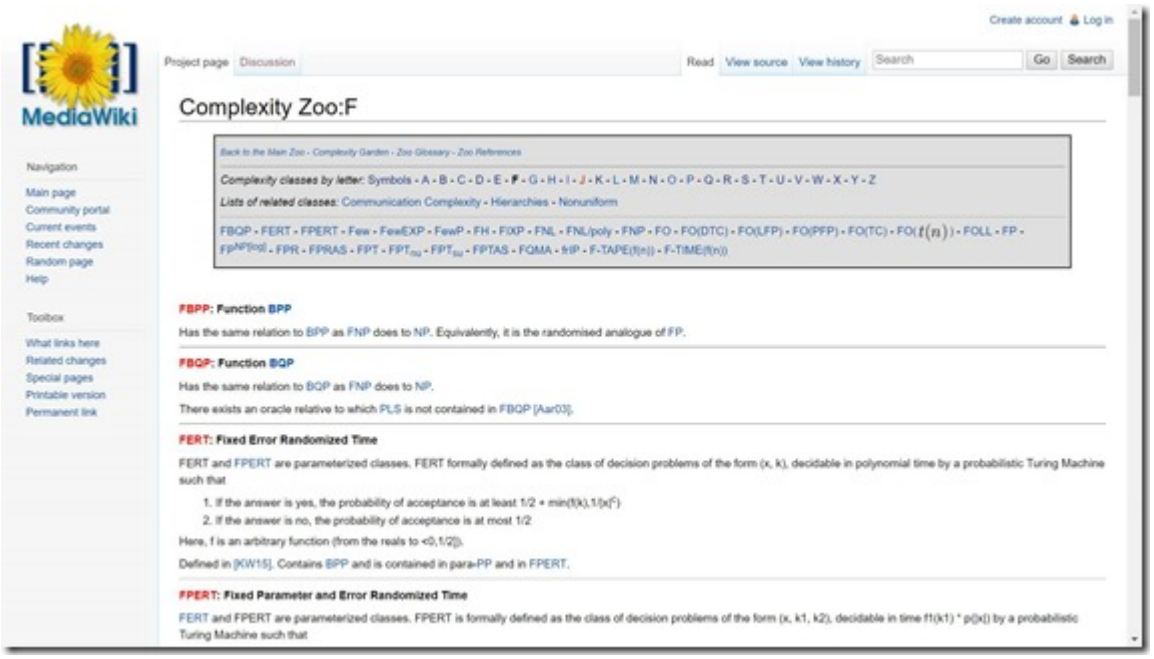
**EXPTIME** : est la classe des problèmes décidés en temps exponentiel par une machine déterministe. Précisément, le temps de calcul de ces problèmes est une puissance de 2 exprimée sous forme d'un polynome de N, N étant le niveau de complexité du problème. Ils sont intractables avec des machines traditionnelles. Certains de ces problèmes peuvent être convertis en problèmes traitables de manière polynomiale par des ordinateurs quantiques. Les jeux d'échecs et de Go sur grille de taille arbitraire font partie de cette catégorie. Dans les grilles à taille limitée, l'effet exponentiel a des limites. Celles-ci ont été dépassées pour les échecs par Deep Blue en 1996 et pour le jeu de Go par AlphaGo de DeepMind en 2016 et 2017.

**NEXPTIME** : est la classe des problèmes décidés en temps exponentiel par une machine non-déterministe et avec un espace mémoire illimité.

**EXPSPACE** : est la classe des problèmes qui peuvent être résolus en espace exponentiel. Autant dire qu'ils sont difficiles d'accès aux machines d'aujourd'hui et même de demain.

Les quatre classes précédentes (PSPACE, EXPTIME, NEXPTIME, EXPSPACE) ne correspondent pas à des problèmes pratiques faciles à identifier dans la vie courante. En tout cas, je n'en ai pas trouvé dans la littérature. Ils couvrent dans l'ensemble les problèmes de prévision de comportement de systèmes ultra-complexes avec de fortes interactions. S'il est possible que la modélisation du repliement d'une protéine soit un problème NP, quelle serait la classe du problème de simulation du fonctionnement d'une cellule vivante entière, voir d'un organisme multicellulaire ? Les interactions sont tellement nombreuses au niveau atomique, moléculaire et cellulaire que la classe de ce genre de problème est probablement située bien au-delà de NP-Difficile.

Il existe bien d'autres classes de complexité de problèmes que je ne vais pas décrire ici : EXP, IP, MIP, BPP, RP, ZPP, SL, NC, AC0, TC0, MA, AM et SZK ! Elles sont listées dans un **site web** qui inventorie le zoo des classes de complexité de problèmes. Il semble il y en avoir plus d'une centaine.




Pour en savoir plus sur le sujet des théorie de la complexité, vous pouvez notamment parcourir le très documenté **Computational Complexity A Modern Approach**, de Sanjeev Arora et Boaz Barak de l'Université de Princeton, 2007 (489 pages).

**Classes de complexité quantiques**

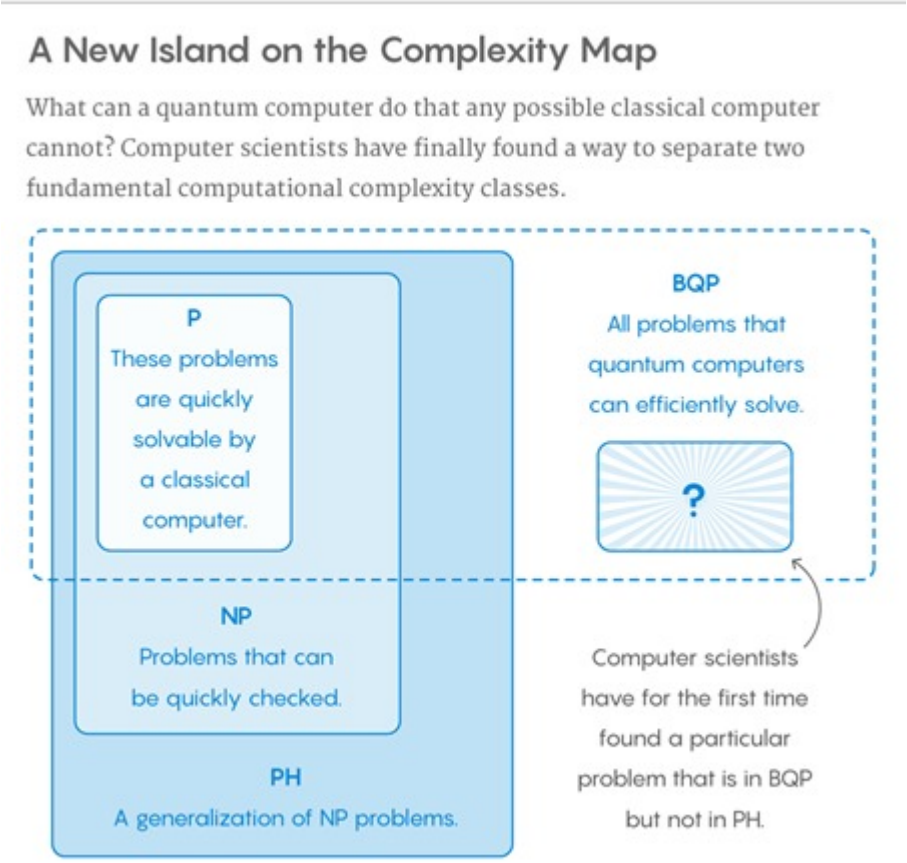
On peut y ajouter une classification de problèmes par niveau de difficulté pour les ordinateurs quantiques, la correspondance avec les classes *ci-dessus* étant encore un problème… non entièrement résolu ! La classification est différente car les ordinateurs quantiques peuvent paralléliser les traitements comme les ordinateurs classiques assimilables à des machines de Turing ne peuvent le faire.

Il faudrait y ajouter une contrainte connue des ordinateurs quantiques : leur temps de cohérence qui est non seulement fini mais relativement court. Il contraint par la durée le nombre de portes quantiques que l'on peut enchaîner pour résoudre un problème. Et ce temps est inférieur au dixième de seconde pour un ordinateur

quantique à base de supraconducteurs. C'est une contrainte que n'ont pas les ordinateurs traditionnels. On peut y faire tourner un algorithme jusqu'à plus soif. Un problème trop complexe pour un ordinateur quantique serait donc aussi un problème qui nécessiterait d'enquiller un nombre de portes quantiques trop grand pour s'exécuter plus rapidement que le temps de cohérence.

**PH** : est la classe des problèmes qui peuvent être résolus par des ordinateurs traditionnels du présent et, surtout, par tout ordinateur traditionnel du futur ! PH signifie "Polynomial Hierarchy".

**BQP** : définit une classe de problème qui est traitable en temps polynomial sur un ordinateur quantique avec un taux d'erreur contraint. Cela peut dans certains cas correspondre à des problèmes P. La classe a été définie en 1993, au moment où apparaissaient les premiers algorithmes quantiques. En débat, le fait de savoir sur la classe BQP est véritablement différente de la classe P ! Il est déjà démontré que la classe P des problèmes polynomiaux est dans BQP. Mais est-ce que NP est dans BQP ? A priori non. Mais c'est difficile à prouver de manière générique. Mais on sait déjà que BQP est dans NP. C'est là que l'on trouve une bonne part des meilleurs algorithmes quantiques comme ceux de Grover et de Shor.




Le point clé est de trouver des algorithmes qui font partie de BQP (traitables en quantique) et qui ne sont pas dans PH (traitables avec n'importe quel architecture traditionnelle du présent et du futur). Cette incertitude a été levée très récemment. Cf **Finally, a Problem That Only Quantum Computers Will Ever Be Able to Solve de Kevin Hartnett**, 2018, qui fait référence à **Oracle Separation of BQP and PH** des Israéliens Ran Raz et Avishay Tal, mai 2018 (22 pages), présenté dans la conférence Electronic Colloquium on Computational Complexity. Les deux logiciens ont trouvé des algorithmes à base d'oracles (boites noires) qui sont dans BQP mais pas dans PH. Donc, qui ont un temps de résolution polynomial sur ordinateur quantique mais qui reste exponentiel sur ordinateur classique. Alea jacta est ! Mais je ne vais pas prétendre avoir compris cette belle démonstration !

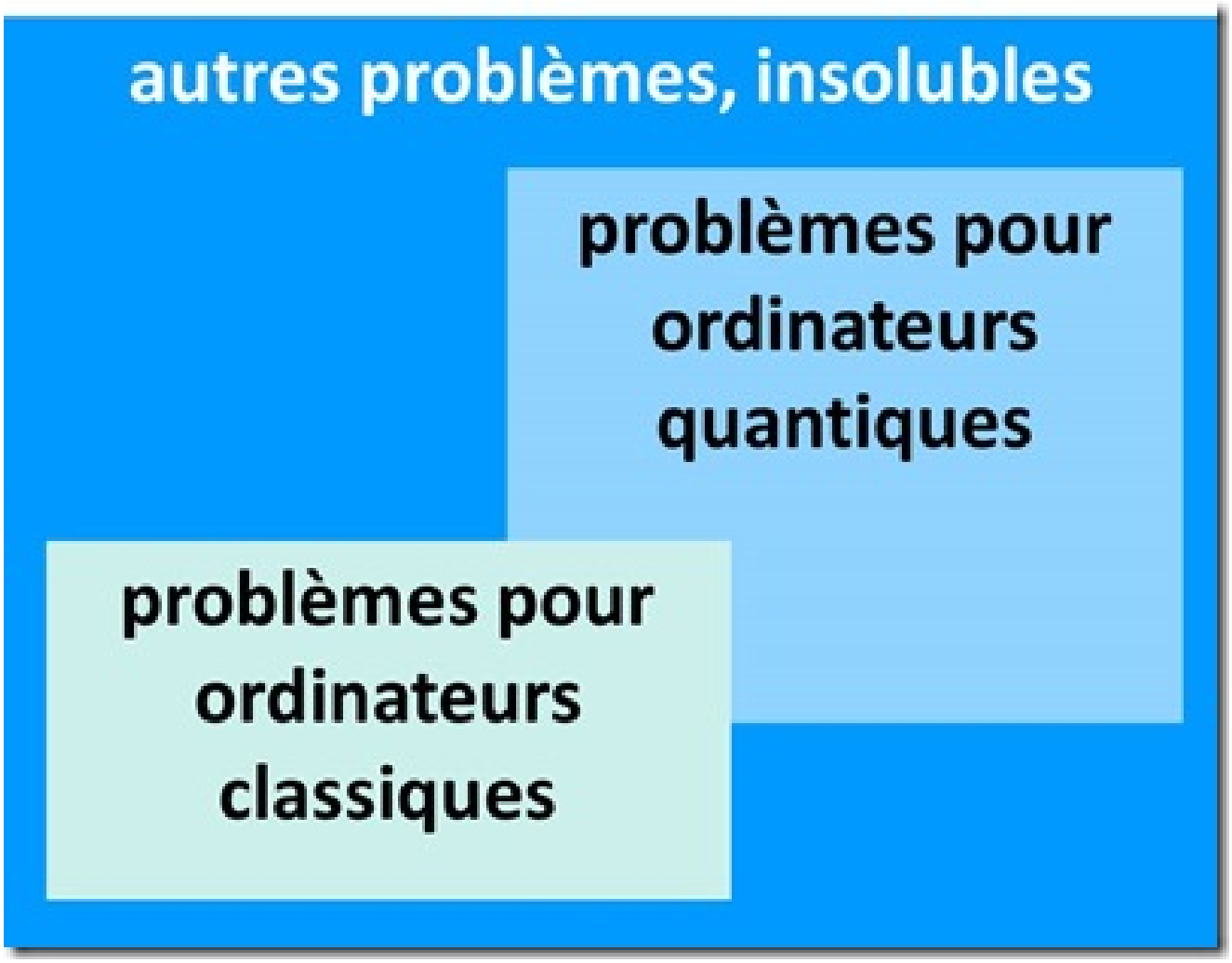
Qu'en est-il au passage d'une éventuelle différence de complexité de problèmes gérable avec des ordinateurs

quantiques à portes universelles vs les ordinateurs à recuit quantique de style D-Wave ? D'après **Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation** de Dorit Aharonov, Wim van Dam et Julia Kempe (du CNRS), 2008 (30 pages), il n'y en aurait pas. Divers théorèmes montrent qu'un problème qui peut être résolu avec des portes quantiques universelles peut aussi l'être avec une architecture de recuit quantique à la D-Wave et réciproquement et dans un ordre de grandeur de temps équivalent.

**QMA** (pour Quantum Merlin Arthur) définit une classe de problèmes qui est vérifiable en temps polynomial sur un ordinateur quantique avec une probabilité supérieure aux 2/3. C'est l'analogue quantique de la classe de complexité "traditionnelle" NP. La classe QMA contient les classes P, BQP et NP. Cf **QMA-complete problems** de Adam Bookatz, 2013 (18 pages). Comme la classe NP, la classe QMA a deux sous-classes, QMA Complet et QMA Difficile. En pratique, ce sont des problèmes difficiles à résoudre avec des ordinateurs quantiques. Malheureusement, la littérature sur le sujet n'en décrit pas la nature ou ne fournit pas d'exemples. C'est bien dommage pour ceux qui apprécient d'adopter un sens pratique des choses !

**QCMA** est une classe hybride entre QMA et NP. La preuve est fournie en temps classique polynomial mais la résolution relève du niveau QMA et est réalisée de manière quantique. Je n'ai pas bien compris et ce n'est pas bien grave.

Nombre de publications relèvent les limitations des algorithmes et ordinateurs quantiques. Un problème BQP qui n'est pas dans PH donne l'avantage au quantique. Mais des problèmes intractables exponentiels pour lesquels l'amélioration apportée par le quantique n'est que racinaire (racine carrée du temps classique) ne modifie pas leur nature exponentielle.




C'est ce que relève Scott Aaronson dans **The Limits of Quantum Computers** (16 pages) ainsi que dans **NP-**

**complete Problems and Physical Reality** (23 pages). Les problèmes NP Complets et au-delà restent inaccessibles aux ordinateurs quantiques. Bref, la force brute a des limites que même le calcul quantique ne permet pas de dépasser en théorie ! Cela explique en partie la difficulté à créer des algorithmes quantiques efficaces.

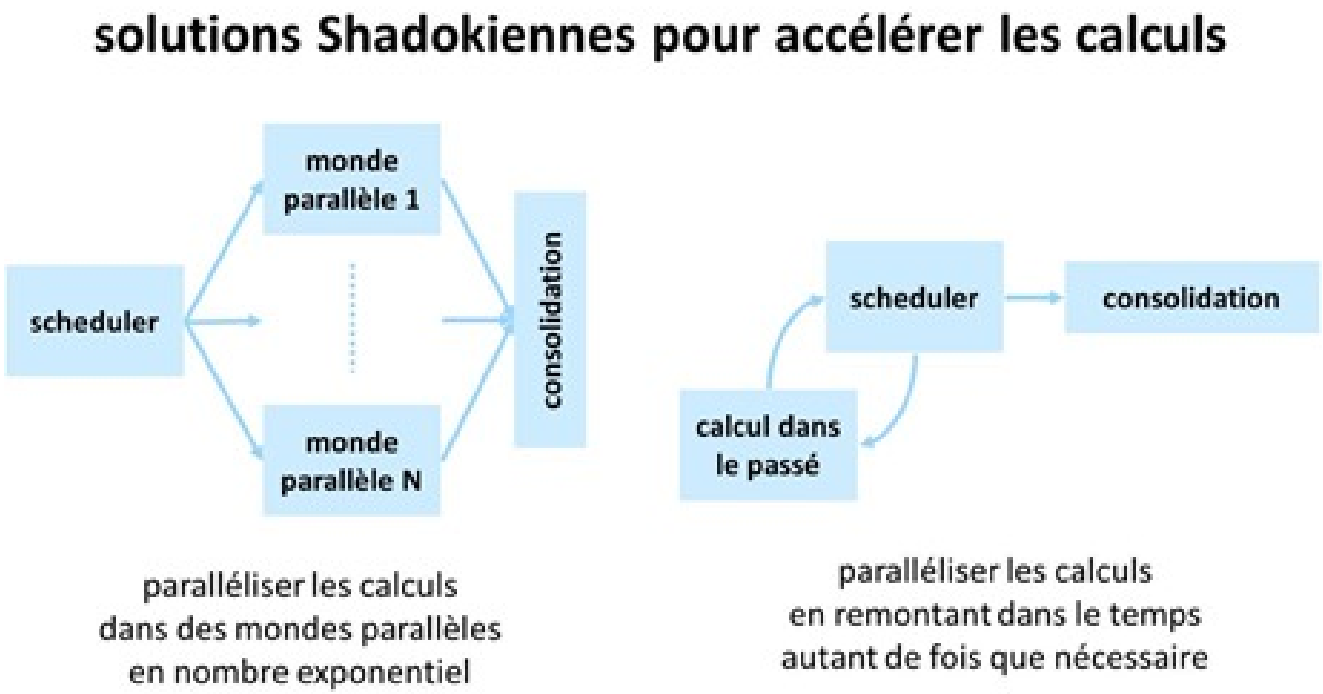
Comme vu dans **Complexite algorithmique** de SylvainPerifel, 2014 (430 pages), il faut aussi prendre en compte le fait que certains problèmes sont indécidables, à savoir qu'ils ne peuvent pas être résolus par un algorithme, quel que soit le temps dont on dispose. Il en va ainsi de la détermination de l'arrêt d'un programme dans une machine de Turing. En d'autres termes, il n'existe pas de programme permettant de savoir si n'importe quel programme écrit dans un langage de programmation courant va s'arrêter ou boucler pendant une durée infinie.

Dans le même ordre, le **théorème de Rice** démontre qu'aucune propriété non triviale d'un programme ne peut être décidée de manière algorithmique. Tout cela pour dire qu'il n'existerait pas de méthode automatique de détection de bugs dans un programme ou de certification qu'il s'exécute bien. Il existe cependant des méthodes de preuve formelle permettant de certifier l'exécution de programmes spécifiques. Cela passe par l'utilisation de spécifications formelles des programmes qui servent de référence pour l'évaluation de leur bon fonctionnement. C'est déjà très utilisé, sans quantique, dans l'informatique industrielle et dans les systèmes critiques tels que ceux de l'aérospatial.

**Contournements de science-fiction**

Une autre barrière semble infranchissable aux ordinateurs de tout type : la barrière du temps de Planck, qui est de 10 puissance –43 secondes. Il serait impossible de réaliser un calcul élémentaire en moins de temps que ce temps de Planck. C'est une limite de la loi de Moore en plus de la barrière de la chaleur de Landauer qui définit la quantité d'énergie minimale nécessaire pour modifier une information. Mais cette barrière temporelle de Planck est loin d'être atteinte. Les ordinateurs à base de processeurs CMOS sont limités à une fréquence d'horloge de 4 à 6 GHz, donc 2*10 puissance –10 secondes. Avec un processeur photonique pouvant atteindre théoriquement 100 Ghz, on en serait à 10 puissance – 11 secondes mais ceux-ci sont très difficiles à réaliser. Et cela donne de la marge, 10 puissance 32, avant d'atteindre la barrière temporelle de Planck.

Autre solution pour transcender les calculateurs quantique d'architecture connue et qui relève de la science-fiction : faire des calculs dans des mondes parallèles pour tester toutes les solutions à un problème complexe et consolider les résultats obtenus. C'est l'application du scénario de la série TV "Fringe" au calcul quantique. Il va sans dire que pour que cela ait un intérêt, il faudrait pouvoir faire cela dans un nombre exponentiel d'univers parallèles. Il faudrait donc aussi que la méthode soit "scalable". Si on ne pouvait l'utiliser que dans un nombre limité de mondes parallèles, cela n'aurait pas un grand intérêt !




On peut aussi imaginer remonter dans le temps pour refaire les calculs plusieurs fois et consolider également les résultats. Dans ces deux scénarios shadokiens, l'une des difficultés parmi d'autres serait de créer un goulot d'étranglement dans la consolidation des résultats. Il faudrait un bus de données suffisamment rapide pour absorber une grande quantité d'information. On pourrait d'ailleurs se demander s'il n'est pas possible de réduire l'un des cas à l'autre voir de les combiner. Ainsi, si on calcule dans le passé, on peut choisir de le faire dans des passés simultanés ou dans des passés différents, histoire d'éviter de trop encombrer les passés.

Les deux méthodes pourraient peut-être aussi être opérantes avec des ordinateurs traditionnels. Mais l'ajout du quantique rend la chose plus crédible, si l'on peut dire. Revenir dans le passé ou se déplacer instantanément dans des mondes parallèles est probablement bien plus aisé avec des quantums.

Ce sont évidemment des hypothèses qui ne relèvent pas du domaine du possible, même en tortillant les lois de la physique au gré de ses désirs les plus fous. Scott Aaronson évoque pourtant quelques-uns de ces scénarios dans **NP-complete Problems and Physical Reality**, 2005 (23 pages). En pratique, au mieux peut-on tirer partie de la vitesse supraluminique de connexion entre qubits intriqués, qui peut servir dans certaines circonstances que nous aurons l'occasion d'explorer, mais qui n'accélèrent pas les calculs pour autant.

Bref, l'intractabilité des problèmes NP Complets et QMA pourrait faire partie des principes de base de la physique et rester des limites infranchissables. Jusqu'à ce que l'on trouve une parade astucieuse insoupçonnée ! Une AGI ou intelligence artificielle générale pourrait-t-elle le faire ? Cela devient un problème récursif… !

En attendant, les recherches vont surtout bon train pour permettre la création d'ordinateurs quantiques avec un grand nombre de qubits dotés de caractéristiques opérationnelles clés comme un long temps de cohérence et des taux d'erreurs aussi faibles que possibles. C'est le point de passage obligé pour pouvoir traiter des problèmes exponentiels de grande taille dans des applications métier courantes.

---

Dans la **partie suivante**, il sera temps de revenir dans l'espace du possible et d'examiner les outils de développements disponibles pour créer des applications quantiques. Les ordinateurs quantiques n'étant pas légion, on pourra être surpris par l'abondance d'outils de développement quantiques qui sont déjà disponibles.

Cet article a été publié le 26 juillet 2018 et édité en PDF le 19 mars 2024.
(cc) Olivier Ezratty – “Opinions Libres” – **https://www.oezratty.net**