Calculabilité et Complexité

Quelques résultats que je connais

Etienne Grandjean
GREYC, Normandie Université
CNRS / Université de Caen / ENSICAEN

Journée des Fédérations NorMath et NormaSTIC
20 octobre 2017

Quelques épisodes de l'histoire

- En 1879, Gottlob Frege met en place la *logique moderne* – calcul des prédicats ou logique du premier ordre – en inventant les *quantificateurs* $\forall, \exists$ et leur écriture formelle (qu'il appelle Idéographie).
- A partir de 1874, Georg Cantor invente la *théorie des ensembles* avec, entre autre, la *méthode de diagonalisation*.
- Puis, au 2ème Congrès international des Mathématiciens en 1900, David Hilbert présente ses 23 problèmes...

Questions choisies de David Hilbert

- *Le 10ème problème de Hilbert* (1900) : Trouver un algorithme qui décide pour toute équation diophantienne, c'est-à-dire toute équation polynomiale sur les entiers :

$$p(x_1, \ldots, x_n) = 0$$

  si elle a des solutions entières.
- *Entscheidungsproblem* (1928) : Trouver un algorithme qui, pour toute formule F en logique du premier ordre (calcul des prédicats), donc construite avec les quantificateurs $\forall, \exists$ et les connecteurs logiques, décide si F est un théorème, c'est-à-dire, est valide pour toutes les interprétations possibles des prédicats.

Puis vient Kurt Gödel

- avec son *Théorème de Complétude* (1929) :
  "Une formule logique du premier ordre F est *valide* ($=$ vraie pour toutes les interprétations) si et seulement si F est *démontrable* dans un certain système axiomatique."
- et son *Théorème d'Incomplétude* (prouvé par *diagonalisation* en 1931) :
  "Toute théorie logique axiomatisable T de *l'arithmétique des entiers* $(\mathbb{N}, +, \times, 0, 1, =)$ ou qui intègre l'arithmétique des entiers est *incomplète* ; c'est-à-dire, il existe des théorèmes F (formules valides) *non démontrables* dans T."

A l'inverse, Presburger a construit en 1928 un système axiomatique pour la théorie logique de la structure $(\mathbb{N}, +, 0, 1, =)$, c'est-à-dire l'arithmétique avec addition mais sans multiplication...

Enfin, J. Herbrand, A. Church, A. Turing et S. Kleene font naître la Calculabilité

en inventant

- le λ-*calcul* de Church (1935)
- les *fonctions récursives* de Herbrand-Gödel-Kleene (1931-1934)
- la *machine de Turing* (1936)

et en prouvant l'équivalence de ces trois modèles : tous trois définissent exactement la *même notion de fonction calculable*.

D'où la *Thèse de Church-Turing* (implicite dans les articles de 1935-1936, mais explicitée et appelée "Thèse de Church" par Kleene en 1943) :

"Toute fonction calculable par un algorithme,
au sens intuitif,
est calculable par machine de Turing (ou par...)"

Church et Turing démontrent l'indécidabilité des problèmes...

- Validité d'une formule logique du premier ordre : c'est-à-dire, *Entscheidungsproblem* de Hilbert (Church 1935)
- Arrêt d'une machine de Turing (Turing 1936)

toujours par la méthode de *diagonalisation*
et en *admettant* la Thèse de Church-Turing.

Le problème de l'arrêt d'une machine de Turing

Outil technique : Tout programme ou toute machine de Turing M est représentable par un mot binaire appelé ici $\texttt{code}(M)$.

On s'intéresse au problème suivant, sous-problème du problème de l'arrêt :

ARRET-DIAGONAL
Donnée : Une machine de Turing M donnée par $\texttt{code}(M)$.
Question : Partant de la donnée $\texttt{code}(M)$, la machine M s'arrête-t-elle ? ce qu'on note alors : $M(\texttt{code}(M))$ *s'arrête*.

Indécidabilité de l'arrêt

Supposons qu'il existe une machine de Turing M_1 qui décide le problème ARRET-DIAGONAL, c'est-à-dire, telle que

- si $M(\texttt{code}(M))$ s'arrête (resp. ne s'arrête pas)
- alors, partant de la donnée $\texttt{code}(M)$, la machine M_1 répond OUI (resp. répond NON),
  ce qu'on note $M_1(\texttt{code}(M)) = \texttt{OUI}$ (resp. $M_1(\texttt{code}(M)) = \texttt{NON}$).

On modifie sur un point le programme de la machine M_1 qui devient la machine M_2 de programme suivant :

- faire agir M_1
- si M_1 répond OUI alors boucler.

Indécidabilité de l'arrêt (Fin)

On déduit immédiatement les implications suivantes, pour toute machine de Turing M :

- Si $M(\texttt{code}(M))$ s'arrête alors $M_1(\texttt{code}(M)) = OUI$, donc $M_2(\texttt{code}(M))$ boucle (ne s'arrête pas)
- Si $M(\texttt{code}(M))$ ne s'arrête pas alors $M_1(\texttt{code}(M)) = NON$, donc $M_2(\texttt{code}(M))$ s'arrête.

D'où on tire l'équivalence :

$$M(\texttt{code}(M)) \text{ s'arrête } \iff M_2(\texttt{code}(M)) \text{ ne s'arrête pas}$$

En prenant $M = M_2$, on obtient la contradiction.

Donc aucune machine de Turing ne décide le problème ARRET-DIAGONAL !

Suivirent une pléthore de problèmes indécidables

- Validité d'une formule dans la structure arithmétique usuelle $(\mathbb{N}, +, \times, 0, 1, =)$ (K. Gödel, 1930 ?)
- Problème de Correspondance de Post (E. Post, 1946)
- 10ème problème de Hilbert : Solution des équations diophantiennes (M. Davis, H. Putnam et J. Robinson, années 48-60, enfin Matijasevic, 1970)
- Problèmes de Pavage du plan (par tuiles de Wang, par polyominos, R. Berger, 1966), etc.

Essentiellement, l'indécidabilité de chacun de ces problèmes est démontrée par *réduction* :

- le problème de l'ARRET se *réduit* au problème en question,
- donc, si ce problème *était* décidable alors le problème de l'ARRET le *serait*.

Les années 40-70 voient la naissance des Ordinateurs et de l'Informatique

avec leur cortège de nouveaux concepts et problèmes :

- Systèmes d'exploitation
- Langages de programmation
- Automates, Grammaires formelles et Compilateurs
- Bases de données et leurs langages
- Sécurité et Cryptographie
- et le développement systématique des **Algorithmes**...

La naissance de la Complexité Algorithmique

En 1965, deux contributions précurseuses :

- Le *logicien* Cobham donne une caractérisation indépendante des machines de la classe des fonctions *calculables en temps polynomial*, *identifiant* ainsi la classe de complexité P ou `PTIME` ;
- Dans son article “Paths, Trees and Flowers” où il donne le premier algorithme qui résout en temps polynomial $O(n^4)$ le problème-phare du *couplage maximum* dans un graphe quelconque, *l'algorithmicien* Jack Edmonds *identifie* de même la classe des problèmes `PTIME` comme celle des problèmes *tractable*.

La naissance de la Complexité Algorithmique

La même année 1965, J. Hartmanis et R. Stearns publient un article fondateur où ils définissent les classes de complexité *en temps* $\texttt{DTIME}(T(n))$ et *en espace* $\texttt{DSPACE}(S(n))$ sur machines de Turing et y démontrent, toujours par *diagonalisation*, le premier théorème de *hiérarchie* sur le temps et l'espace :

- Pour toutes fonctions $T_1(n) \geq n$ et $T_2(n)$ constructibles en temps telles que $T_1(n) \log T_1(n) = o(T_2(n))$, on a

$$\texttt{DTIME}(T_1(n)) \subsetneq \texttt{DTIME}(T_2(n))$$
$$(\text{exemple : } \texttt{DTIME}(n^2) \subsetneq \texttt{DTIME}(n^3))$$

- Pour toutes fonctions $S_1(n) \geq \log n$ et $S_2(n)$ constructibles en espace telles que $S_1(n) = o(S_2(n))$, on a

$$\texttt{DSPACE}(S_1(n)) \subsetneq \texttt{DSPACE}(S_2(n)).$$

La consécration de la Complexité Algorithmique : Cook, Karp et Levin

Enfin, vinrent les célèbres articles

- de S. Cook "The complexity of theorem-proving procedures", 1971
- et de R. Karp "Reducibility about combinatorial problems", 1972

où

- est identifiée/nommée la classe **NP** ainsi que la notion de **NP-complétude** par réductions polynomiales
- et où les problèmes de référence SAT, CLIQUE, Problème du Voyageur de commerce, etc.
  sont démontrés NP-complets avec une méthodologie de preuve bien établie.

En Union Soviétique, L. Levin publiait indépendamment des résultats similaires dans "Universal sequential search problems" en 1973.

Les deux caractérisations équivalentes de NP

Un langage/problème $L \subseteq \Sigma^*$ est NP

- s'il est *reconnu par une machine de Turing non déterministe* en temps polynomial
- ou s'il est *vérifiable en temps polynomial*, c'est-à-dire, s'il existe un problème $V \in \texttt{PTIME}$ et un polynôme p tels que, pour tout $w \in \Sigma^*$,

$$w \in L \iff \exists v \, (|v| \leq p(|w|) \wedge (w, v) \in V).$$

V est appelé *problème de vérification* pour L et le mot v, de taille $|v|$ polynomiale en la taille $|w|$ de w, est appelé *témoin/certificat/solution polynomial* de w pour le problème L.

La notion fondamentale de NP-complétude (au sens de R. Karp)

Un problème/langage L est *NP-complet* si

1. L est NP et
2. tout autre problème $L' \subseteq \Sigma^*$ qui est NP *se réduit polynomialement à* L, c'est-à-dire, il existe une fonction r calculable en temps polynomial telle que, pour tout $w \in \Sigma^*$,

$$w \in L' \iff r(w) \in L$$

Conclusion des travaux sur la NP-complétude années 70-90 : une quasi-dichotomie

Au fil des années, les algorithmiciens ont su montrer que la quasi-totalité des problèmes "naturels" qui sont NP (= la grande majorité des problèmes de recherche en informatique)

- soit sont PTIME (= ont des algorithmes *efficaces*)
- soit sont NP-complets.

Point-clé :

- Si un jour *un seul* problème NP-complet était prouvé être PTIME
- alors *tous* les problèmes NP-complets le seraient aussi
- et on aurait P=NP.

Les exceptions : un problème NP qui a résisté longtemps, un autre qui résiste encore

- **La Primalité** : problème co-NP (son complémentaire, la non-primalité est évidemment NP)
  - a été prouvé être NP (Pratt, 1975)
  - et aussi RandomPolynomial, i.e. décidable en temps polynomial par un algorithme probabiliste avec erreur bornée (Miller-Rabin 1976)
  - et finalement être PTIME (Agrawal-Kayal-Saxena 2002)
- **L'Isomorphisme de graphes**...

Remarque : Au-delà de la primalité, Shor a donné un algorithme probabiliste de *factorisation des entiers* qui fonctionne en temps polynomial sur un *calculateur quantique*.

L'isomorphisme de graphes

Données : deux graphes G_1 et G_2 de même ensemble de sommets $\{1, 2, \ldots, n\}$, par exemple présentés par leurs matrices d'adjacence

$$M_1 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \text{ et } M_2 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Question : Les graphes G_1 et G_2 sont-ils isomorphes, c'est-à-dire, existe-t-il une permutation π de l'ensemble des sommets $\{1, 2, \ldots, n\}$ telle qu'on ait

$$M_1(\pi(i), \pi(j)) = M_2(i, j)$$

pour tous $i, j \in \{1, 2, \ldots, n\}$?

L'Isomorphisme de graphes : un problème NP très particulier

Première particularité : Son comptage est "facile"

- Si ce problème était PTIME alors le problème du comptage du nombre d'isomorphismes entre deux graphes serait aussi PTIME (Mathon, 1979)

A comparer avec le point suivant :

- La classe de complexité $\sharp P$ définie comme la classe des problèmes de comptage des solutions des problèmes NP-complets, SAT par exemple, est considérée comme beaucoup plus difficile que la classe NP.

Point-clé : La classe de comptage $\sharp P$ permet d'exprimer (contient) toute la *hiérarchie de complexité polynomiale* (Toda, 1989)

L'Isomorphisme de graphes : un problème NP très particulier

Seconde particularité : C'est un problème *sous-exponentiel*

- L'Isomorphisme de graphes est décidable en temps $2^{O((\log n)^3)}$ (Babai-Helfgot, 2015-2017), donc *quasi-polynomial*.

A comparer avec le point suivant :

- *Hypothèse ETH (Exponential Time Hypothesis)* communément admise : Tout algorithme qui résout un problème NP-complet, SAT par exemple, a une complexité au pire qui est au moins 2^{n^d}, pour une constante $d > 0$.

Conjecture : l'Isomorphisme de graphe est un problème PTIME...

Et pourtant...

Dès 1975, Richard Ladner publie le résultat suivant :
Théorème de Ladner : Si P$\neq$ NP alors il existe des problèmes NP qui ne sont

- ni PTIME
- ni NP-complets.

Remarque : C'est encore un résultat prouvé par une méthode de diagonalisation (sophistiquée).

Mais, comme pour le théorème de hiérarchie sur l'espace et le temps (ex : $\texttt{DTIME}(n^2) \subsetneq \texttt{DTIME}(n^3)$), les problèmes sont construits ad hoc !

On ne voit pas de problème NP "naturel" candidat à n'être ni PTIME ni NP-complet...

Un curieux phénomène...

Pratiquement tous les problèmes “naturels” difficiles ont pu être classés/prouvés :

- soit NP-complets
- soit PSPACE-complets (jeux : Othello, etc., Equivalence d'automates finis, Théorie du 1er ordre de presque tous les graphes)
- soit EXPTIME-complets (jeux : Dames, Echecs, Go)
- soit EXPSPACE-complets
- ....

via des réductions PTIME,

soit, pire, prouvés indécidables.

On a même plus...

Beaucoup de problèmes sont *équivalents* via des réductions calculables *en temps linéaire*.

- Par exemple, les problèmes NP-complets : SAT et les problèmes de graphes 3-COLORATION, NOYAU et VERTEX-COVER sont linéairement équivalents.

En conséquence,

- si l'un de ces problèmes était calculable en temps $O(n^d)$,
- alors tous les autres le seraient aussi *pour le même degré d*...

Le même phénomène est vrai... pour certains problèmes PTIME

Par exemple,

- le produit de deux matrices $n \times n$
- et l'inversion d'une matrice $n \times n$

ont et auront toujours *exactement* la même complexité : à ma connaissance, la borne supérieure est à l'heure actuelle $O(n^{2,376})$.

Mais quelles bornes inférieures de complexité sait-on prouver pour les problèmes “naturels” ?

Paradoxalement, on *sait* prouver des bornes inférieures pour les problèmes *les plus difficiles* :

- Prouver qu'un problème est *indécidable* : problème de pavage, ambiguïté d'une grammaire, équation diophantienne, etc.
- La preuve qu'un problème A est `EXPTIME`-*complet* (Go, Echecs, etc.) fournit une constante $d > 0$ telle que $A \notin \texttt{DTIME}(2^{n^d})$ par le théorème de hiérarchie en temps.
- De même, la preuve qu'un problème est `PSPACE`-*complet* fournit toujours aussi une borne inférieure de complexité en espace.

Un exemple

Il a été prouvé en 1983 que la théorie T des formules logiques (du 1er ordre) *vraies dans presque tous les graphes* est PSPACE-complète avec

- la borne supérieure $T \in \mathtt{DSPACE}((n/\log n)^2)$
- et les bornes inférieures $T \notin \mathtt{NSPACE}(o(n/\log n))$ et $T \notin \mathtt{NTIME}(o((n/\log n)^2))$.

Définitions :

- Une formule F portant sur les graphes est *vraie dans presque tous les graphes* si la proportion des graphes G à n sommets où F est vraie tend vers 1 quand n tend vers l'infini.
- $\mathtt{NSPACE}(S(n))$ et $\mathtt{NTIME}(T(n))$ sont les classes de problèmes décidables respectivement en espace $S(n)$ et en temps $T(n)$ sur une machine de Turing *non déterministe*.

Et pour les problèmes NP-complets ?

En 1982, dans son discours de réception du Prix Turing, Stephen Cook a dit/écrit :

"...the record for proving lower bounds on problems of small complexity is appalling. In fact there is

no nonlinear time lower bound known on a general purpose computation model

for any natural problem in NP, in particular, for any of the 300 problems listed in the reference book by Garey and Johnson, 1979..."

Pour quelles raisons, cette situation ?

Une remarque d'abord :

- La somme des longueurs des k premiers entiers en binaire : $1, 10, 11, 100, 101, 110, \ldots$ (ou en décimal, etc.) est

$$\Theta(k \log k)$$

Puis, une observation personnelle (facile à vérifier dans chaque cas) :

- Pour tout problème NP-complet naturel L, tout certificat/témoin/solution v de toute donnée $w \in L$ est de taille inférieure à celle de la donnée : $|v| \leq |w|$;
- Dans un grand nombre de cas, on a même $|v| = O(n/\log n)$ où $n = |w|$: voir ci-après.

Intuitivement, le *certificat est bien plus petit que la donnée*.

Certificat plus petit que la donnée
L'exemple du problème SAT : un cas particulier

La formule propositionnelle

$F : (p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_3) \wedge (p_2 \vee \neg p_1 \vee p_4) \wedge (\neg p_4 \vee \neg p_2)$

est satisfaite par l'interprétation

$$p_1 = 0,\ p_2 = 1,\ p_3 = 1,\ p_4 = 0$$

qu'on peut noter de façon concise par le vecteur/mot

$$0110$$

de longueur inférieure à la longueur $|F|$ de F.

Certificat plus petit que la donnée
Le cas général du problème SAT

Soit une formule propositionnelle F, portant sur k variables $p_1, p_2, \ldots, p_k$ et satisfaite par une interprétation $p_1 = i_1, \ldots, p_k = i_k$,
donc avec le mot certificat de longueur k

$$v = i_1 i_2 \ldots i_k \in \{0,1\}^k.$$

Soit n la longueur de F dans un alphabet Σ fixé, par exemple
$\Sigma = \{p, 0, 1, (,), \wedge, \vee, \neg\}$.

Du fait que les mots (les variables) $p_1, \ldots, p_k$ sont tous distincts, on tire les inégalités suivantes :

$$n = |F| \geq |p_1| + |p_2| + \ldots + |p_k| \geq c\ k \log k$$

pour une certaine constante c,
d'où la longueur *sous-linéaire* du certificat v :

$$|v| = k = O(n/\log n)$$

Difficultés à prouver des bornes inférieures de complexité pour les problèmes NP-complets

Constat : Les problèmes NP-complets “naturels”

- sont *trop facilement vérifiables* :
- ils ont des certificats *trop petits*,
- c’est-à-dire, utilisent *trop peu de non déterminisme*...

Ce qu'on sait :
La classe NLIN, analogue linéaire de la classe NP

C'est l'ensemble des problèmes L

- reconnus en *temps linéaire* par des machines RAM *non déterministes*
- ou, de façon équivalente, tels qu'il existe un problème de vérification V décidable en temps linéaire sur RAM et une constante c, tels que, pour toute donnée w,

$$w \in L \iff \exists v(|v| \leq c|w| \wedge (w, v) \in V)$$

c'est-à-dire, L est *vérifiable en temps linéaire* avec des *certificats de taille linéaires*.

Tous les problèmes NP-complets "naturels" (dont les 21 problèmes NP-complets de base étudiés dans l'article fondateur de Karp, 1972)

sont NLIN

La classe NLIN, analogue linéaire de la classe NP

- Tous les problèmes NP-complets "naturels" sont NLIN.
- Il existe des problèmes NLIN-complets "naturels".

Plus précisément, le problème RISA ("*Reduction of Incompletely Specified deterministic finite Automata*" : problème répertorié AL7 de la liste des problèmes NP-complets du livre de Garey et Johnson, 1979)

est NLIN-*complet* via des réductions calculables en *temps linéaire sur machine de Turing*.

Par ailleurs, il a été prouvé (Paul-Pippenger-Szemeredi-Trotter 1983) l'inclusion *stricte*

$$DTIME_{Turing}(O(n)) \subsetneq NTIME_{Turing}(O(n))$$

qui est l'équivalent de $P \neq NP$ pour le *temps linéaire des machines de Turing*...

Une borne inférieure modeste pour le problème RISA

La NLIN-complétude du problème RISA (prouvée en 1990) implique

- que RISA n'est pas décidable en temps linéaire sur machine de Turing :

$$RISA \notin DTIME_{Turing}(O(n))$$

à cause de l'inclusion stricte
$DTIME_{Turing}(O(n)) \subsetneq NTIME_{Turing}(O(n)) \subseteq NLIN$

Constat : depuis la conférence de Stephen Cook en 1982 il y a 35 ans...

- On n'a jamais su prouver aucune autre borne inférieure de complexité en temps *au-delà du linéaire*, par exemple le temps n^2, ou même $n^{1+\epsilon}$, pour aucun problème NP-complet "naturel", sur un *modèle de calcul de portée générale* (machine de Turing à plusieurs rubans, RAM...)
- Un comble pour des problèmes conjecturés de complexité en temps au moins *exponentiel* !

On a pourtant un théorème de hiérarchie du temps non déterministe...

- Soit $NTIME_{Turing}(T(n))$ la classe des problèmes reconnus par une machine de Turing non déterministe en temps $T(n)$.

Stephen Cook avait démontré dès 1973 le théorème de hiérarchie suivant (par une diagonalisation, complétée par une technique de point fixe et de rembourrage) :

- Pour toutes constantes $d \geq 1$ et $d' > d$, on a l'inclusion stricte

$$NTIME_{Turing}(n^d) \subsetneq NTIME_{Turing}(n^{d'})$$

Une situation surprenante au premier abord

- Pour toutes constantes $d \geq 1$ et $d' > d$, on a l'inclusion stricte

$$NTIME_{Turing}(n^d) \subsetneq NTIME_{Turing}(n^{d'})$$

et pourtant, on se sait prouver pour aucun problème NP-complet "naturel" L,

1. $L \notin NTIME_{Turing}(n^d)$, pour un degré $d > 1$, ni même
2. $L \notin DTIME_{Turing}(n^d)$, pour un degré $d > 1$.

Raison du point (1) :

- tout problème NP-complet "naturel" est dans NLIN (classe du temps linéaire des RAM non déterministes), classe elle-même incluse dans $NTIME_{Turing}(n \log n)$.

Tout problème NP "naturel" est toujours PTIME ou NP-complet ? Des résultats en faveur d'une telle dichotomie...

Soit $H = (V_H, E_H)$ un graphe et soit le problème NP suivant paramétré par H :

H-**COLORIAGE**

Donnée : un graphe $G = (V_G, E_G)$.

Question : Existe-t-il un *homomorphisme* f de G dans H ?

Définition : Un *homomorphisme* de $G = (V_G, E_G)$ dans $H = (V_H, E_H)$ est une application $f : V_G \to V_H$ telle que pour toute arête $(x, y) \in E_G$ on a $(f(x), f(y)) \in E_H$.

Exemple : Si on prend pour H la clique à 3 éléments (graphe-triangle) K_3, le problème du K_3-coloriage d'un graphe G est exactement le problème de l'existence d'un coloriage des sommets de G avec 3 couleurs de façon que deux sommets adjacents ne soient jamais de même couleur.

Tout problème NP "naturel" est toujours PTIME ou NP-complet ? Des résultats en faveur d'une telle dichotomie...

Théorème (Hell-Nesetril 1990) : Soit H un graphe quelconque. On a le résultat dichotomique suivant :

- Le problème H-coloriage est PTIME si le graphe H est biparti.
- Sinon, le problème H-coloriage est NP-complet.

Au-delà de la dichotomie du H-coloriage...

De façon encore bien plus significative, le résultat de Hell et Nesetril a été généralisé au problème du $\mathcal{S}$-coloriage pour une structure $\mathcal{S}$ quelconque : graphe orienté, relation ternaire, etc.

Théorème (prouvé indépendamment par Bulatov et Zhuk, 2017, réponse positive à une Conjecture fameuse de Feder-Vardi 1998) : Soit une structure $\mathcal{S}$ quelconque.

- Soit le problème $\mathcal{S}$-coloriage est PTIME
- Soit il est NP-complet.

De plus, on sait décider si le $\mathcal{S}$-coloriage est `PTIME` ou NP-complet.

Importance du Théorème de Bulatov et Zhuk
Une dichotomie de portée générale

- Tout problème de satisfaction de contraintes (en Anglais : CSP) sur des domaines finis s'exprime comme un problème de $\mathcal{S}$-coloriage...
- Or, une bonne partie des problèmes d'Optimisation et des problèmes de l'Intelligence Artificielle s'expriment comme des problèmes de satisfaction de contraintes sur domaines finis...

La dichotomie ainsi prouvée intègre donc tous ces problèmes.

Conclusion

- Les problèmes phare sont souvent plus faciles à décider/calculer qu'il semble au premier abord : voir exemple de la Primalité.
- On sait démontrer des bornes inférieures de complexité pour les problèmes les plus difficiles... Exemple : existence de stratégie gagnante dans un jeu.
- Par contre, on n'a obtenu aucune borne inférieure significative (au-delà du linéaire) pour aucun problème NP-complet "naturel" : SAT, CLIQUE.
- Un grand nombre de problèmes difficiles sont prouvés équivalents entre eux via des réductions polynomiales (problèmes NP-complets, PSPACE-complets) ou même des réductions linéaires.
- On a des résultats de dichotomie, typiquement entre PTIME et NP-complet.