HAL open science

Où sont les problèmes difficiles

Christine Solnon

▶ To cite this version:

Christine Solnon. Où sont les problèmes difficiles. Bibliothèque Tangente, 2021, Bibliothèque Tangente Num., 75, pp.50-53. hal-03350964

HAL Id: hal-03350964
https://hal.science/hal-03350964

Submitted on 21 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

OÙ SONT LES PROBLÈMES DIFFICILES ?

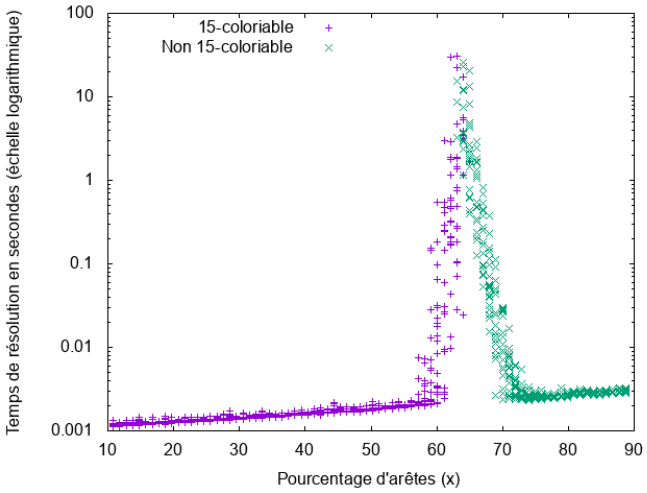
CHRISTINE SOLNON, CITI, INRIA, INSA LYON

Les problèmes NP-complets (introduits dans l'article "P est-il égal à NP ?") sont des problèmes fascinants pour lesquels personne n'a trouvé d'algorithme efficace dans *tous* les cas. Cependant, il existe généralement des algorithmes qui sont très efficaces dans *certains* cas. La question que nous étudions ici est : quels sont ces cas ? Nous allons notamment voir que la difficulté des problèmes NP-complets dépend beaucoup moins de leur taille que d'un paramètre permettant de contrôler leur faisabilité.

Nous allons illustrer cela avec le problème de k-coloration. Étant donnés un graphe et un nombre k de couleurs, l'objectif de ce problème est de décider si le graphe est k-coloriable, c'est-à-dire s'il est possible de colorier tous ses sommets de telle sorte que les sommets voisins soient de couleurs différentes. Ce problème a été montré NP-complet par Karp en 1972 et, par conséquent, nous ne connaissons pas d'algorithme efficace pour tous les graphes. Cependant, il existe des algorithmes très efficaces pour certains graphes, y compris des graphes ayant beaucoup de sommets. Par exemple, si chaque sommet a moins de k voisins, alors nous pouvons affirmer immédiatement que le graphe est k-coloriable. A l'inverse, si le graphe a une arête entre chaque paire de sommets et s'il y a plus de k sommets, alors nous pouvons facilement démontrer que le graphe n'est pas k-coloriable. Ainsi, le nombre d'arêtes semble avoir un impact sur le fait qu'un graphe est k-coloriable ou pas, et nous allons voir qu'une transition de phase apparaît lorsque nous faisons varier ce paramètre.

Qu'est-ce qu'une transition de phase ?

Pour illustrer ce phénomène, nous faisons une expérience simple. Nous générons des graphes en fixant le nombre de sommets n à 70 et en faisant varier le pourcentage d'arêtes x entre 10% et 90%. Plus précisément, notons $m_{max} = \frac{n(n-1)}{2}$ le nombre maximum d'arêtes, correspondant au cas où il y a une arête pour chaque paire de sommets. Pour chaque valeur de x, nous générons 20 graphes différents en sélectionnant aléatoirement $m = m_{max} * \frac{x}{100}$ arêtes différentes selon une distribution uniforme. Ensuite, nous résolvons le problème pour $k = 15$ couleurs en utilisant notre programme préféré [1]. Les résultats de cette expérience sont visualisés dans le graphique ci-dessous : chaque point (x, y) correspond à un graphe ayant $m_{max} * \frac{x}{100}$ arêtes ; la valeur de y correspond au temps de résolution, et la couleur du point indique si le graphe est 15-coloriable (+ violet) ou pas (× vert).



De façon prévisible, nous observons deux régions : une région à gauche où tous les graphes générés sont 15-coloriables (car ils ont peu d'arêtes), et une région à droite où aucun des graphes générés n'est 15-coloriable (car ils ont trop d'arêtes). De façon un peu moins prévisible, la transition entre ces deux régions est étroite : la région où les 20 graphes générés sont 15-coloriables va jusque $x = 64\%$ alors que la région où aucun n'est

1. Nous avons utilisé le programme Color6 disponible sur `https://home.mis.u-picardie.fr/~cli/EnglishPage.html`.

15-coloriable commence à $x = 67\%$. C'est ce phénomène qui est appelé transition de phase par analogie avec la transformation physique d'un système d'un état vers un autre en fonction d'un paramètre externe (par exemple, le passage de l'état solide à l'état liquide en fonction de la température). Dans notre cas, le paramètre est le pourcentage d'arêtes dont l'augmentation fait passer de l'état 15-coloriable à l'état non 15-coloriable.

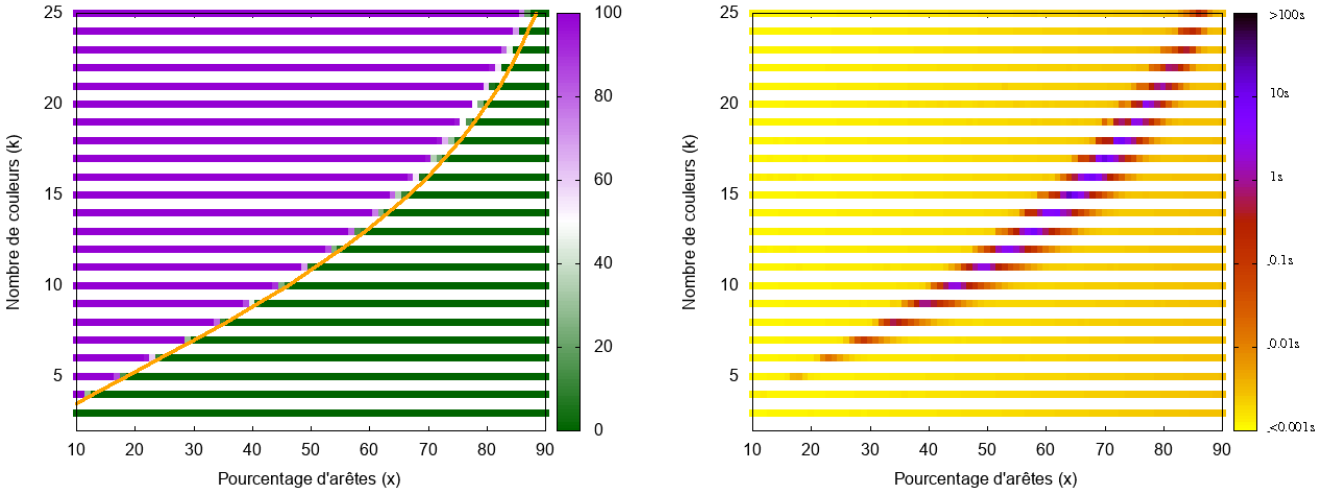
Ce phénomène de transition de phase est d'autant plus intéressant qu'il est lié à un pic de complexité : quand x est suffisamment loin de la transition ($x \leq 55\%$ ou $x \geq 75\%$), le problème est résolu en quelques millièmes de secondes, tandis que quand x est au cœur de la transition, la résolution peut nécessiter plus de 10 secondes.

Peut-on prédire où se trouve la transition de phase ?

Oui ! Il suffit de rechercher la valeur de x pour laquelle la moitié des graphes sont k-coloriables. Étant donnés n et k, considérons la variable aléatoire S_x correspondant au nombre de solutions du problème consistant à k-colorier un graphe ayant $m_{max} * \frac{x}{100}$ arêtes choisies aléatoirement selon une distribution uniforme. Nous devons chercher la valeur de x pour laquelle la probabilité que $S_x \geq 1$ est égale à 0,5. Si on fait l'hypothèse que S_x a une loi normale, cela revient à chercher la valeur de x pour laquelle l'espérance de S_x est égale à 1. Cette valeur peut être calculée en résolvant un problème de dénombrement que nous laissons à titre d'exercice. Pour $n = 70$ et $k = 15$, l'espérance de S_x est égale à 1 quand $x = 66\%$, ce qui correspond à peu près au pic de la figure 1.

Que se passe-t-il si on change le nombre k de couleurs ?

Pour répondre à cette question, nous faisons varier k entre 3 et 25, et le pourcentage d'arêtes x entre 10% et 90%. Pour chaque valeur de k et x, nous générons aléatoirement 20 graphes ayant $m_{max} * \frac{x}{100}$ arêtes et nous essayons de colorier ces graphes avec k couleurs. Dans le graphique ci-dessous à gauche, la couleur de chaque point (x, y) donne le pourcentage de ces graphes pour lesquels une solution a été trouvée quand $k = y$, tandis que la courbe orange correspond à la prévision théorique de la transition (lorsque l'espérance de S_x est égale à 1). Dans le graphique de droite, la couleur de chaque point correspond au temps moyen de résolution pour ces mêmes graphes (jaune si moins d'un millième de seconde, et noir si plus de 100 secondes).



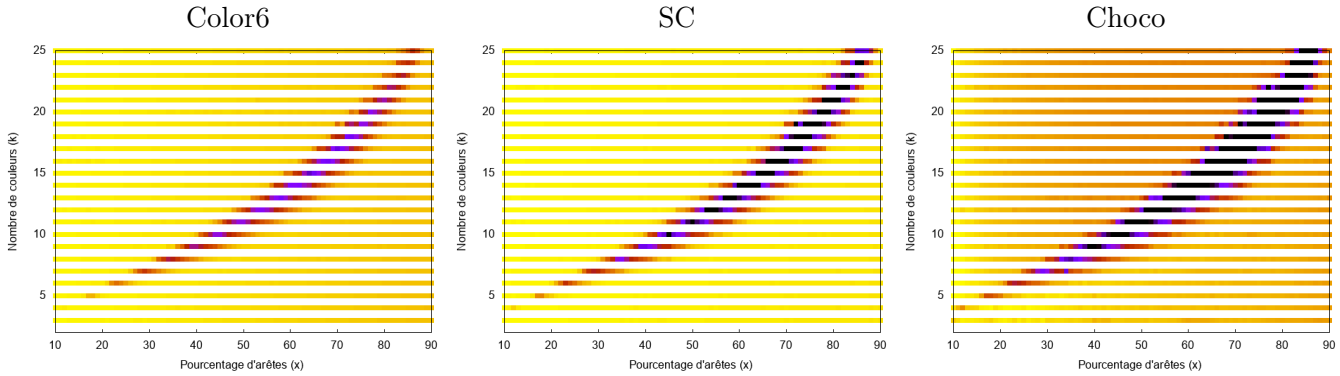
Sur le graphique de gauche, nous observons une transition de phase entre une région où les 20 graphes générés sont k-coloriables (points violet en haut à gauche, correspondant aux cas où il y a beaucoup de couleurs par rapport au nombre d'arêtes), et une région où aucun des graphes n'est k-coloriable (points verts en bas à droite, correspondant aux cas où il y a peu de couleurs par rapport au nombre d'arêtes). Les temps de résolution pour les graphes de ces deux régions sont de l'ordre du millième de seconde. La transition entre ces deux régions est très étroite, et correspond aux graphes pour lesquels les temps de résolution sont les plus importants (supérieurs à 10 secondes dans certains cas).

Observe-t-on le même phénomène si on change de programme ?

Le programme considéré dans les expériences précédentes (Color6) utilise un principe de résolution appelé *Conflict Driven Clause Learning* (CDCL), souvent utilisé pour résoudre le problème de la satisfiabilité de formules booléennes (SAT). Nous pouvons nous demander si d'autres programmes, utilisant d'autres principes

de résolution, rencontrent les mêmes difficultés sur les mêmes graphes. Afin d'apporter quelques éléments de réponse, considérons deux autres programmes : SC, qui parcourt de façon exhaustive l'espace des colorations en utilisant l'heuristique DSATUR proposée par Brélaz en 1979 pour choisir le prochain sommet à colorier [2], et Choco, qui utilise un principe de programmation par contraintes.

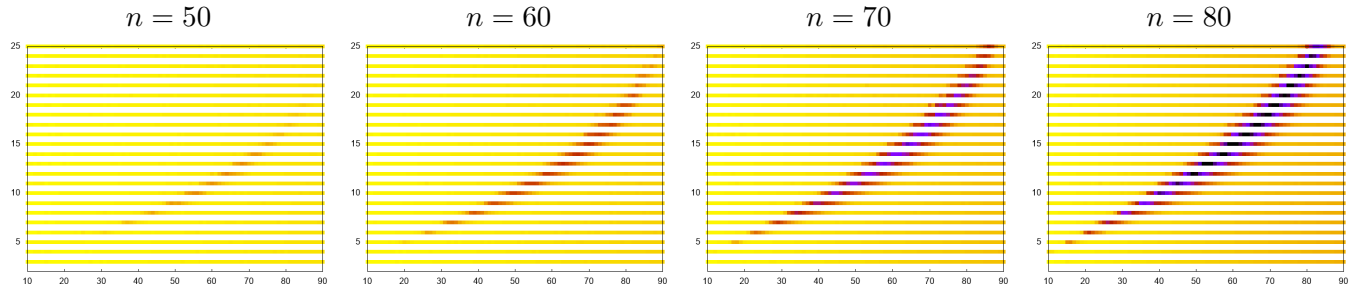
Les graphiques ci-dessous comparent les temps d'exécution de ces trois programmes quand $n = 70$ et $3 \leq k \leq 25$: la couleur de chaque point (x, y) correspond au temps moyen de résolution pour k-colorier 20 graphes ayant $m_{max} * \frac{x}{100}$ arêtes quand $k = y$ (même échelle de couleurs que précédemment).



Pour les trois programmes, les problèmes les plus difficiles sont situés dans la région de la transition de phase, même si on observe des différences de performance : SC est le plus efficace pour les problèmes faciles, Color6 est le plus efficace pour les problèmes difficiles, et Choco est toujours moins efficace que SC et Color6 (ce qui n'est pas étonnant d'une part parce qu'il est implémenté en Java alors que SC et Color6 sont en C, et d'autre part parce qu'il s'agit d'un moteur générique de résolution alors que SC et Color6 sont dédiés au problème de k-coloration).

Comment évoluent les temps de résolution quand on augmente le nombre n de sommets ?

La réponse à cette question dépend de si on est proche ou non de la transition de phase, comme cela est illustré dans les graphiques suivants, où le nombre de sommets augmente de 50 à 80 (pour chaque valeur de n, la couleur du point (x, y) correspond au temps moyen de résolution de Color6 pour 20 graphes ayant $m_{max} * \frac{x}{100}$ arêtes quand le nombre de couleurs est égal à y).



Quand on est loin de la transition de phase, les temps de résolution augmentent doucement et sont toujours inférieurs à un centième de seconde. En revanche, quand on est au cœur de la transition de phase, on observe un phénomène d'explosion combinatoire où une légère augmentation du nombre de sommets se traduit par une très forte augmentation des temps d'exécution. Par exemple, le temps de résolution des problèmes les plus difficiles passe de 6 millisecondes quand $n = 50$ à 2 secondes quand $n = 60$, 4 minutes quand $n = 70$, et 3/4 d'heure quand $n = 80$.

À quoi ça sert ?

Ce que nous avons observé ici sur le problème de k-coloration se généralise à tous les problèmes NP-complets : pour tous ces problèmes, il existe un ou plusieurs paramètres permettant de contrôler l'existence d'une solution. En général, la zone de transition entre la région où il existe des solutions et celle où il n'en existe pas est très étroite, et cette zone contient les problèmes qui sont effectivement difficiles à résoudre, les autres problèmes étant généralement très faciles.

2. Ce programme est disponible sur `http://perso.citi-lab.fr/csolnon/simpleColour.tgz`.

Cette connaissance est très utile pour évaluer les performances d'un programme résolvant un problème NP-complet car elle permet de contrôler la difficulté des jeux d'essai : nous pouvons facilement générer des problèmes de difficulté croissante et évaluer ainsi la capacité du programme à passer à l'échelle en fonction de cette difficulté.

En pratique, les problèmes que nous devons résoudre "en vrai" ne sont pas générés aléatoirement, mais sont construits à partir de données réelles. Par exemple, le problème de k-coloration peut être utilisé pour allouer des fréquences radio à des transmetteurs tout en évitant les interférences : les sommets correspondent aux transmetteurs, les couleurs aux fréquences, et les arêtes à des contraintes d'incompatibilité (dues aux interférences) empêchant deux transmetteurs d'utiliser une même fréquence. Pour ces problèmes réels, on observe également que la difficulté de résolution n'est pas nécessairement liée à la taille, et qu'il existe un très grand nombre de ces problèmes qui sont faciles à résoudre en pratique. Malheureusement, la distribution de probabilité des données réelles n'est généralement pas connue, de sorte que nous ne pouvons pas construire des modèles permettant de prévoir la difficulté de ces problèmes comme nous l'avons fait ici dans le cas où les arêtes de nos graphes sont générées selon une distribution uniforme.