Article 36. Problème de Satisfaction de Contraintes : Structure et Complexité en Temps

T. A. Ralijaona, H. Faliharimalala
École du Génie du Management d'Entreprises et du Commerce,
Institut Supérieur de Technologie d'Antananarivo,
tiana.ralijaona@ist-tana.mg
heritianamihanta@yahoo.fr

Mots clés : Problème de satisfaction de contraintes, Complexité en temps, Backtracking, Intelligence artificielle, N-Reines.

Résumé

Le backtracking est l'algorithme générique et classique qui permet de résoudre exactement les problèmes de satisfaction de contraintes. Mais sa complexité en temps pour des problèmes non triviaux est d'ordre exponentiel. Dans la pratique, une combinaison d'heuristiques et de méthodes d'optimisation combinatoire est utilisée pour résoudre ces problèmes en un temps raisonnable. Mais rien ne garantit l'exactitude de la solution. À travers l'exemple des N-reines (vu ici comme un cas particulier de problème de satisfaction de contraintes), les résultats de nos travaux démontrent qu'un lien existe bien entre structure du problème et complexité en temps de l'algorithme de résolution. En effet, après avoir mené une simulation et recueilli les temps d'exécution pour différentes tailles du problème des N-reines, les résidus de la régression du temps d'exécution en fonction de la taille du problème exhibent une matrice de corrélation non diagonale. Ces résultats corroborent donc l'idée qu'une étude de la structure mathématique d'un problème permettra d'atteindre du moins une amélioration significative des méthodes de résolution exacte.

I. Introduction

En optimisation combinatoire, les problèmes de satisfaction de contraintes sont des problèmes où l'on cherche à déterminer l'état ou la valeur de variables de décision satisfaisant certaines contraintes ou critères. Ces problèmes sont au cœur même de la recherche en intelligence artificielle et en recherche opérationnelle car elles fournissent un modèle générique pour résoudre une multitude de problème dans ces domaines tel que les problèmes de planification, d'ordonnancement et d'allocation de ressources [3].

Pour résoudre ces problèmes, l'algorithme de backtracking et ses variantes permettent de trouver une solution exacte. Cependant, son temps d'exécution pour des problèmes non triviaux est d'ordre exponentiel. Ce qui est une pénalité certaine pour la résolution des problèmes de grande taille ou même de taille moyenne [1, 4]. Inversement, les algorithmes probabilistes et génétiques sont quant à eux plus efficients mais ils ne garantissent pas toujours une solution exacte au problème à résoudre [2, 5]. D'un autre point de vue, des techniques hybrides essayent d'exploiter la structure du graphe sous-jacent au problème pour améliorer le temps de résolution du problème en question [6]. D'où l'interrogation : existe-t-il un lien entre structure intrinsèque du problème et complexité en temps de l'algorithme de résolution ?

Pour répondre à cette question, nous avons effectué une simulation dans laquelle nous avons résolu le problème des N-reines (vu ici comme un cas particulier du problème de satisfaction de contraintes) pour différentes tailles. Ensuite, nous avons cherché à déterminer si il existait bien un lien entre la structure du problème, représentée ici par la taille du problème, et le temps d'exécution de l'algorithme de résolution.

II. 2 Matériels et méthodes

Tout d'abord, nous allons définir le problème de satisfaction de contraintes. Puis, nous allons décrire l'algorithme de backtracking qui permettra de le résoudre. Ensuite, nous décrirons le problème des N-reines. Et finalement, nous donnerons la méthodologie utilisée pour tester l'existence ou non du lien entre la taille du problème des N-reines et le temps d'exécution de l'algorithme de backtracking.

2.1 Problème de satisfaction de contraintes

Un *problème de satisfaction de contraintes* (ou CSP) est un triplé $P = \langle X, D, C \rangle$ [1] où :
$X = \{X_1, X_2, ..., X_n\}$ est un ensemble de variables,
$D = \{D_1, D_2, ..., D_n\}$ est l'ensemble des domaines de valeurs respectifs des variables,
$C = \{C_1, C_2, ..., C_m\}$ est l'ensemble des contraintes qui vont restreindre les valeurs que peuvent prendre les variables.
Pour un CSP, chaque contrainte $C_j \in C$ est un couple $\langle t_j, R_j \rangle$ où :
$t_j \subseteq X$ est un sous-ensemble de k variables,
R_j est une relation k-aire sur les domaines correspondants aux variables.
Soulignons que c'est bien chaque relation R_j qui va restreindre les valeurs que peuvent prendre simultanément les k variables de t_j.
Une *solution* pour un CSP est une affectation de valeurs pour toutes les variables provenant de leurs domaines respectifs telle que toutes les contraintes soient satisfaites. Dans ce cas, le problème est dit *satisfiable*. Dans le cas contraire, il est *insatisfiable*.

2.2 Algorithme de backtracking

Une méthode pour résoudre un CSP est l'*algorithme de backtracking*. Elle consiste à affecter à la suite aux variables du problème des valeurs. La validité d'une contrainte est alors vérifié aussitôt que les variables correspondantes à cette contrainte sont toutes instanciées (c'est à dire, affectées de valeurs). Si une instanciation partielle viole une des contraintes alors la valeur de la dernière variable instanciée est changée par la prochaine valeur disponible. Au cas où toutes les valeurs ont été épuisées pour cette dernière alors on remonte à l'avant dernière variable et ainsi de suite [4].

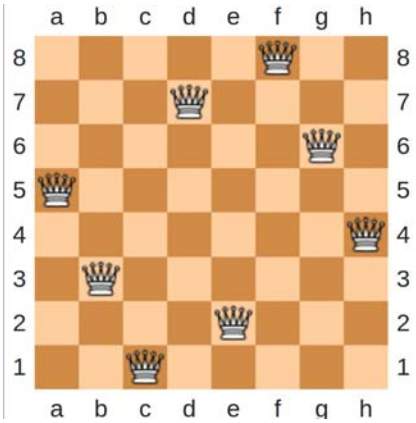
Un pseudocode de l'algorithme de backtracking est donné ci-dessous :

Procédure Backtrack(candidat):
Si Rejeter(candidat) alors retourner
Si Accepter(candidat) alors retourner candidat
extension ⟵ PremierExtension(candidat)
Tant que extension ¬ = ∅:
Backtrack(extension)
extension ⟵ ProchainExtension(extension)
FinProcédure
où

Rejeter(candidat) : retourne Vrai si le candidat partiel ne peut pas être complété,
Accepter(candidat) : retourne Vrai si le candidat est une solution,
PremierExtension(candidat) : génère la première extension de candidat,
ProchainExtension(extension) : génère la prochaine extension.

2.3 Problème des N-reines

Le problème des N-reines consiste à placer N reines sur un échiquier à N lignes et N colonnes de manière à ce qu'aucune des reines ne s'attaquent. Autrement dit, il faudra placer les N reines de telle façon à ce qu'aucune reine ne se trouve ni sur la même ligne ni sur la même colonne ni sur la même diagonale.



Le CSP fournit une structure générique pour modéliser mathématiquement le problème des N-reines. Ainsi, nous pouvons résoudre ce dernier en utilisant l'algorithme de backtracking décrit précédemment.

2.4 Méthodologie

La méthodologie utilisée pour tester le lien entre la structure du problème et le temps d'exécution de l'algorithme de résolution est la suivante. Dans un premier temps, nous avons exécuté l'algorithme de backtracking pour résoudre le problème des N-reines pour différentes tailles du problème (le nombre de reines à placer). Après avoir recueilli les temps d'exécution pour chaque taille, nous avons effectué une régression linéaire du logarithme du temps d'exécution en fonction de la taille du problème. Nous avons ensuite calculer les résidus de ce problème afin de tester si il existait un schéma entre eux ; ce qui sera prouvé par le calcul du coefficient de corrélation linéaire entre les résidus originaux et les résidus décalés. En effet, une forte corrélation indiquerait un lien entre structure du problème et complexité en temps de l'algorithme de résolution.

III. 3 Résultats et discussions

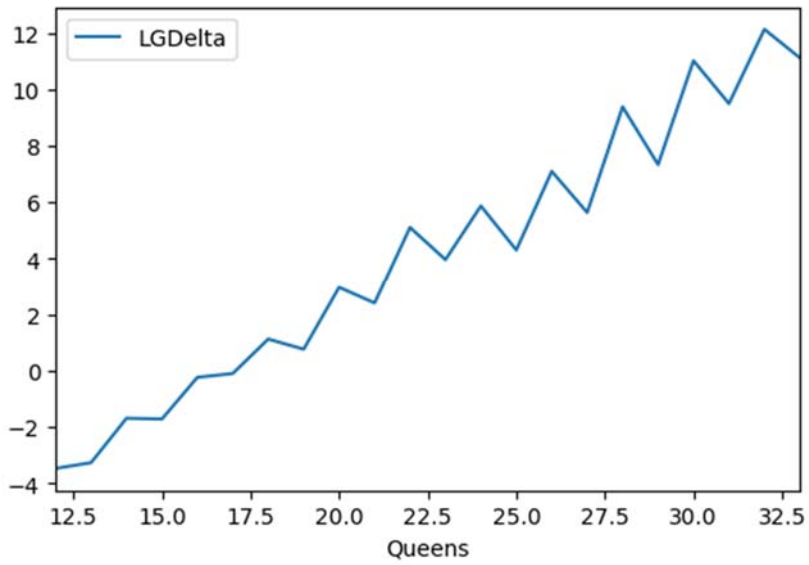
Nous présentons ci-après les résultats de notre étude.

3.1 Résultats

Après avoir résolu le problème des N-reines pour $N = 12, ..., 33$ par l'algorithme de backtracking, nous avons obtenu les temps d'exécution dont un extrait est donné dans le tableau suivant :

N	T (en s)	$\log(T)$
12	0,031	-3,473768
13	0,038	-3,270169
14	0,184	-1,692820
⋮	⋮	⋮
32	183.176,716 ≃ 51 heures	12,118207
33	66.907,952 ≃ 19 heures	11,111073

Et le graphe ci-dessous nous montre le logarithme du temps d'exécution en fonction du nombre de reines. Ce graphe exhibe bien le fait que le temps d'exécution de l'algorithme est d'ordre exponentiel en nombre de reines.



Nous remarquerons ici la structure en dents de scie qui suggère déjà une corrélation entre les résidus de la régression.

La significativité des paramètres de la droite de régression ci-dessous confirme de surcroît le temps d'exécution d'ordre exponentiel.

$$\log(T) = 0,7247N - 12,2533$$

Param.	Coeff.	P-values	R^2
N	0,7247	$< 10^{-3}$	0,961
Intercept	-12,2533	$< 10^{-3}$	

De plus, le coefficient de détermination R^2 démontre qu'on a un bon modèle.

Finalement, le résultat ci-dessous nous donne les coefficients de corrélation linéaire entre les résidus originaux e, les résidus décalés d'ordre 1 $L(e)$ et les résidus décalés d'ordre 2 $L^2(e)$.

	Coeff.	P-values
$\mathrm{cor}(e, L(e))$	-0,8228	$< 10^{-3}$
$\mathrm{cor}(e, L^2(e))$	0,8787	$< 10^{-3}$

Ces résultats nous montrent qu'on a une corrélation forte entre les résidus originaux et les résidus décalés. Ce qui démontrent bien le fait qu'il existe une relation entre la structure du problème et la complexité en temps de la résolution du problème.

3.2 Discussions

Au vue des résultats précédents, on peut alors affirmer que l'étude de la structure du problème permettra d'améliorer la complexité en temps de la résolution du problème. Ce que suggère en effet les méthodes hybrides avancées par [6]. Et dans le cas des problèmes de satisfaction de contraintes, cette structure est capté par le graphe ou l'hypergraphe sous-jacent au problème [4]. Une étude de la structure des hypergraphes nous permettra alors entre autres d'améliorer l'efficacité de la résolution des problèmes de satisfaction de contraintes.

IV. Conclusion

Nous avons donc pu voir que la performance des algorithmes de résolution du problème de satisfaction de contraintes dépendait de l'exploitation de la structure sous-jacente du problème. Ceci a été bien vérifié dans l'exemple des N-reines présenté ici. Et afin d'atteindre cet objectif, il est alors nécessaire d'étudier la structure des hypergraphes modélisant ces problèmes.

Références

[1] S. C. Brailsford, C. N. Potts, B. M. Smith, «Constraint Satisfaction Problems: Algorithms and Applications», European Journal of Operational Research, vol. 119, 1999.

[2] A. E. Eiben, P.-E. Raue, Z. Ruttkay, «Solving Constraint Satisfaction Problems Using Genetic Algorithms», Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 2, 1994.

[3] K. Ghédira, «Constraint Satisfaction Problems : CSP Formalisms and Techniques», Wiley, 2013.

[4] V. Kumar, «Algorithms for Constraint Satisfaction Problems: A Survey», AI MAGAZINE, vol. 13, 1992.

[5] T. Schoning, «A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems», 40th Annual Symposium on Foundations of Computer Science, 1999.

[6] C. Terrioux, P. Jégou, «Bounded Backtracking for the Valued Constraint Satisfaction Problems», Proceedings of the 9th ICPPCP, 2003.