Échantillonnage de solutions pratiquement uniforme en programmation par contraintes

Gilles Pesant[1], Claude-Guy Quimper[2] and Hélène Verhaeghe[1]

[1]Polytechnique Montréal, Canada
gilles.pesant@polymtl.ca, helene.verhaeghe@polymtl.ca
[2]Université Laval, Canada
claude-guy.quimper@ift.ulaval.ca

Juin 2022

Résumé

La capacité d'échantillonner des solutions dans un espace de recherche combinatoire contraint a des applications dans des domaines tels que le raisonnement probabiliste et la vérification de matériel/logiciel. Une propriété souhaitable d'un tel échantillonnage est l'uniformité du tirage aléatoire parmi les solutions. Notre approche se base sur un hachage universel ayant des garanties sur l'uniformité du tirage. Nous résumons ici notre article "Practically Uniform Solution Sampling in Constraint Programming", accepté à CPAIOR2022 [3].

Mots-clés

Échantillonnage uniforme, hachage, dénombrement

Abstract

The ability to sample solutions of a constrained combinatorial space has important applications in areas such as probabilistic reasoning and hardware/software verification. A highly desirable property of such samples is that they should be drawn uniformly at random, or at least nearly so. Our approache approaches is based on universal hashing which provide probabilistic guarantees about sampling uniformity. We summarise here our paper "Practically Uniform Solution Sampling in Constraint Programming" accepted at CPAIOR2022 [3].

Keywords

Uniform sampling, haching

1 Introduction

De nombreux mais importants problèmes peuvent être exprimés sur un espace combinatoire contraint : trouver un élément *satisfaisant* (i.e. une solution) dans cet espace et chercher pour un *optimal* selon un critère sont probablement les tâches les plus communes. *Compter* combien de solutions il y a et *échantillonner* des solutions de manière uniforme aléatoirement ont également des applications importantes. Des exemples de telles tâches existent en planification, inférence Bayesienne, vérification de code, pour ne nommer que celles-ci.

Les compteurs de modèles qui suivent une approche basée sur le hachage universel bénéficient de garanties probabilisitiques concernant la qualité du dénombrement approximatif qu'ils donnent. Dans le contexte des modèles SAT, de tels hachages universels prennent la forme de contraintes de parité (XOR) générées de manière aléatoire [1]. Dans un contexte plus large de variables avec domaine fini et de modèles PPC, ces contraintes se généralisent par des contraintes d'égalités linéaires en arithmétique modulaire générées de manière aléatoire, telles qu'investiguées dans [2].

Récemment, un échantillonneur de solutions pour la PPC [4] s'inspire de l'idée de découpe de l'espace de recherche en cellules et se concentre sur une telle cellule. Cette découpe est faite en ajoutant des contraintes de tables aléatoires de faible arité au modèle PPC. Malheureusement cette méthode n'est ni supportée par des garanties théoriques garantissant la qualité de l'échantillonnage ni n'a montré un échantillonnage uniforme durant les expériences.

Notre méthode ajoute certaines contraintes linéaires modulaires, partageant les mêmes garanties probabiliste que les contraintes XOR pour SAT, afin d'échantillonner les solutions d'un modèle PPC. En pratique, nos expériences montrent que nous arrivons à un échantillonnage presque uniforme. Notre méthode est indépendante du modèle échantillonné et peut être appliquée tant que le solveur PPC supporte les contraintes requises.

2 Échantillonnage uniforme

Notre but est d'utiliser des fonctions de hachage indépendantes les unes des autres pour diviser l'espace de solutions en cellules contenant environ le même nombre de solutions chacune. L'échantillonnage se fait par énumération des solutions d'une cellule de la taille choisie. Un des avantages de telles cellules obtenues par hachage de l'espace est que l'énumération des solutions contenues est plus rapide dû à un espace plus petit. Un autre avantage est la garantie de ne pas avoir de doublon au sein d'une même cellule.

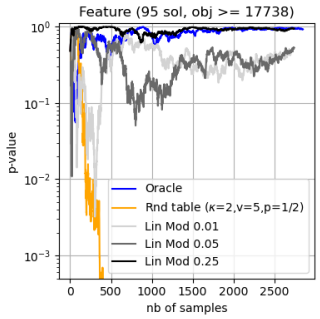

Figure 1: Problème des modèles de caractéristique avec contrainte additionnelle sur l'objectif ($obj \geq 17738$)

Le partitionnement presque uniforme des solutions peut être atteint en ajoutant le système de m inégalités linéaires modulaires sur n variables avec domaine entier fini

$$A\boldsymbol{x} \leq \boldsymbol{b} \pmod p$$

où $\boldsymbol{x}$ est le vecteur des n variables avec domaine entier fini, A est une matrice $m \times n$ dont les éléments appartiennent à $[p] \triangleq \{0, 1, \ldots, p-1\}$, $\boldsymbol{b}$ est un vecteur de m éléments de $[p]$, et p est le modulo. Les inégalités linéaires modulaires possèdent 2 propriétés si le modulo p est un nombre premier. Premièrement, un partitionnement uniforme des solutions, et, secondement, l'indépendance garantie entre deux solutions. En utilisant un nombre premier, on peut également appliquer une élimination de Gauss-Jordan sur le système d'inégalités linéaires afin de le réécrire sous forme paramétrique. Son ensemble de solutions, encodable comme une contrainte de table, peut être obtenu par simple énumération sur les domaines des variables. Cela permet d'arriver à la consistance des domaines par une contrainte de table ajoutée au modèle. La proportion de solutions (ou taille de la cellule) est contrôlé par le nombre m d'inégalités linéaires modulaires ajoutées ainsi que par le choix des les valeurs dans $\boldsymbol{b}$.

3 Expériences

Durant nos expériences, nous avons testé notre méthode sur quatre problèmes: le problème des N-Reines, le problème des modèles de caractéristique (problème de configuration de logiciel), un modèle synthétique_n_d (créé avec n variables de taille de domaine d, de telle sorte que le nombre total de solutions soit calculable analytiquement) et le problème Myciel (coloriage de graphe). Pour vérifier l'uniformité de l'échantillonnage sur les examplaires avec un nombre raisonnable de solutions, nos tests échantillonnent 30 fois le nombre de solutions. Des cellules contenant 1%, 5% et 25% des solutions sont utilisées pour effectuer l'échantillonnage. Pour les examplaires avec plus de solutions, une centaine de solutions sont échantillonnées via des cellules contenant 0.001% de solutions.
La qualité de l'échantillonnage est analysée en calculant l'évolution de la p-valeur. La figure 1 montre l'évolution de la p-valeur à chaque solution échantillonnée en plus sur l'examplaire du problème des modèles de caractéristiques auquel une contrainte à été ajoutée sur la valeur de l'objectif (afin de réduire le nombre de solution à 95). Plus celle-ci est proche de 1, plus l'échantillonnage est uniforme. Nous nous comparons à un oracle (ligne bleue) qui suppose toutes les solutions connues et qui tire une solution au hasard à chaque fois (basé sur le générateur aléatoire de `java`). Nous nous comparons également à la méthode de Vavrille et al. [4] (ligne orange). Nous pouvons voir sur le graphe que notre méthode obtient un échantillonnage le plus uniforme parmi les méthodes. L'uniformité est également meilleure plus les cellules considérées sont grandes, ce qui est attendu puisque les solutions d'une même cellule sont garanties distinctes. Des résultats similaires sont obtenus sur d'autre problèmes.
D'un point de vue du temps d'exécution, pour les examplaires avec beaucoup de solutions, pour une énumération d'une cellule contenant 0.005% des solutions, nous avons pu observer que notre méthode est beaucoup plus rapide que d'énumérer l'entièreté des solutions pour ensuite sélectionner le nombre de solutions équivalent au contenu de la cellule. Nous obtenons également une bien meilleure uniformité que [4].

4 Conclusion

Notre papier décrit une nouvelle approche pour échantillonner de manière uniforme un ensemble de solutions d'un modèle de PPC. Les résultats empiriques démontrent que notre méthode atteint presque l'uniformité.

References

[1] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A scalable approximate model counter. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, pages 200–216. Springer, 2013.

[2] Gilles Pesant, Kuldeep S. Meel, and Mahshid Mohammadalitajrishi. On the usefulness of linear modular arithmetic in constraint programming. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 18th International Conference, CPAIOR 2021, Vienna, Austria, July 5-8, 2021, Proceedings*, pages 248–265. Springer, 2021.

[3] Gilles Pesant, Claude-Guy Quimper, and Hélène Verhaeghe. Practically uniform solution sampling in constraint programming. In *CPAIOR22*, 2022.

[4] Mathieu Vavrille, Charlotte Truchet, and Charles Prud'homme. Solution sampling with random table constraints. In *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, pages 56:1–56:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.