LERIA

Raisonnement dans l'Incertain et Contraintes

LERIA/ RIC
Université d'Angers
www.leria.univ-angers.fr

David LESAINT
david.lesaint@univ-angers.fr

Eric MONFROY
eric.monfroy@univ-angers.fr

Membres

- Vincent BARICHARD, MCF
- Martín DIEGUEZ LODEIRO, MCF
- Laurent GARCIA, MCF
- Frédéric LARDEUX, MCF-HDR
- Claire LEFEVRE, MCF
- David LESAINT, PR
- Eric MONFROY, PR
- Igor STÉPHAN, MCF-HDR
- Marc LEGEAY, ECER
- Claudia VASCONCELLOS-GAETE, ECER
- Corentin BÉHUET, Doctorant
- Salah Eddine BOUTERFIF, Doctorant
- Bryan GARREAU, Doctorant

Autres participants

- Frédéric SAUBION, PR

Les thèmes de recherche du LERIA

Le Laboratoire d'Étude et de Recherche en Informatique d'Angers (LERIA) mène des recherches de nature fondamentale et appliquée dans deux domaines connexes de l'informatique : l'intelligence artificielle et l'optimisation combinatoire. Les sujets abordés forment un continuum que l'on peut décliner en quelques grandes thématiques :

- Méthodes de résolution approchée pour l'optimisation combinatoire issues du calcul évolutionnaire ;
- Techniques de modélisation, reformulation et résolution exacte fondées sur la programmation par contraintes et la logique propositionnelle ;
- Langages et algorithmes pour le raisonnement non-monotone fondées sur la programmation par ensembles réponses et autres formalismes de contraintes quantifiées ;
- Méthodes d'apprentissage artificiel en appoint de techniques de résolution ou portées à d'autres champs disciplinaires tels la bioinformatique et la chimie quantique ;
- Méthodes d'interrogation de bases de connaissances fondée sur des modèles de représentation graphiques.

À ces travaux théoriques s'ajoutent le développement d'approches à visée applicative. Ces travaux, souvent menés dans le cadre de collaborations industrielles ou de projets pluridisciplinaires, recouvrent des domaines variés. Citons entre autres exemples la résolution de problèmes logistiques par optimisation combinatoire, l'analyse logique de données fondée sur le raisonnement symbolique, et le développement de systèmes de recommandation par apprentissage artificiel.

Le LERIA est organisé en 3 thèmes d'activité :

- Thème **MOC** : Méta-heuristiques et Optimisation Combinatoire
- Thème **RIC** : Raisonnement dans l'Incertain et Contraintes
- Thème **ARC** : Apprentissage Artificiel et Représentation des Connaissances.

Le thème RIC

Le thème RIC fédère les travaux du laboratoire autour du raisonnement non-monotone

et des approches déclaratives pour la résolution de problèmes combinatoires. L'accent est mis sur la représentation des connaissances et les efforts se portent sur la conception de langages de modélisation et le développement de techniques de transformation ou d'encodage de modèles visant à concilier flexibilité et efficacité. Les travaux, qui abordent les fondements théoriques de ces outils mais aussi leur implantation et leurs applications, s'appuient sur des paradigmes de modélisation haut-niveau tels la programmation par ensembles-réponses (ASP) et la programmation par contraintes (PPC, CHR) ainsi que sur des formalismes plus élémentaires tels CSP et SAT :

- Extension des cadres ASP et CHR : intégration de variables existentielles en ASP pour le raisonnement ontologique, gestion de préférences en ASP possibilistes pour la révision de croyances, quantification de contraintes en CHR pour les problèmes à horizon non borné.
- Modélisation, transformation et encodage : CSP ensemblistes et encodages CSP/SAT, décomposition de problème fondée sur l'interprétation abstraite et la PPC, PPC pour la transformation de modèles en ingénierie des modèles, PPC pour la recherche de motifs en fouille de données.

Programmation par ensembles-réponses [14, 13, 3, 6, 7, 1, 5, 4] L'ASP (Answer Set Programming) est un langage déclaratif développé afin de traiter des connaissances en intelligence artificielle, lorsque les informations sont incomplètes, ou pour résoudre des problèmes combinatoires.

Nos travaux portent sur diverses extensions du cadre de l'ASP et allient théorie et pratique.

Nous développons depuis plusieurs années un solveur ASP, ASPeRiX, basé sur une approche originale guidée par les règles et ne nécessitant pas d'instanciation préalable du programme. Nos travaux récents montrent que l'algorithme sous-jacent à ASPeRiX peut s'exprimer de manière efficace et extensible en CHR. Nous travaillons à une mise à jour majeure d'ASPeRiX basée sur le langage CHR (et sa version CHR++ développée au sein du laboratoire) qui permettra de remplacer le backtrack chronologique d'ASPeRiX par un backtrack intelligent et, d'un autre côté, pourra intégrer aisément des contraintes définies par l'utilisateur ainsi que les extensions modernes d'ASP.

Par ailleurs, dans le cadre de l'étude de la fusion multi-sources et de l'interrogation d'informations issues du web, généralement exprimées en logique de description (projet ANR ASPIQ), nous nous sommes par exemple intéressés à l'intégration dans l'ASP des variables existentielles, inhérentes aux ontologies exprimées en logique de description, ainsi qu'à l'interrogation de ces ontologies en ASP.

D'autre part, nous nous intéressons à la révision de croyances lorsque les connaissances sont exprimées par un programme ASP. Cette problématique étant peu étudiée, nous proposons une sémantique, des postulats de révision et une étude de complexité de la révision en ASP. Enfin, l'intégration de préférences entre connaissances a amené à l'étude de la révision en ASP possibiliste. Nous étudions de nouveaux modes de révision tirant profit de la structure des règles ASP (révision par ajout, par exemple) et nous essayons de déterminer s'il est possible d'élaborer une caractérisation logique de l'ASP possibiliste en termes de logique d'équilibre ce qui pourrait donner un cadre pour l'étude de théories arbitraires dans l'ASP possibiliste.

Concernant l'ASP, nous nous intéressons plus généralement à l'explicabilité des résultats. Cette étude est nécessaire pour nos travaux sur le backjumping et pourra aussi être utilisée dans le cadre de l'ASP possibiliste.

LERIA

Toujours préoccupés par la mise en pratique de l'ASP sur des applications réelles, nous sommes partie prenante du projet Potassco Solutions, initié par nos collègues de Potsdam, dont nous sommes les correspondants français.

Contraintes ensemblistes [11, 9, 10, 15] Par rapport aux modélisations basées sur des matrices ou des entiers, les ensembles sont pratiques car ils réduisent naturellement le nombre de symétries. De plus, il est bien connu que de nombreux problèmes peuvent être facilement modélisés avec des contraintes ensemblistes. Plusieurs solveurs de contraintes spécialisés pour les contraintes ensemblistes existent déjà. Nous nous intéressons à divers aspects des contraintes ensemblistes : les modèles CSP avec contraintes ensemblistes, la réduction des variables à domaine fini et des variables ensemblistes ainsi qu'à l'encodage des contraintes ensemblistes en SAT.

Encodage CSP/SAT [16, 17] Nous travaillons sur les encodages CSP vers SAT afin de mieux comprendre les facteurs clés qui déterminent leurs caractéristiques ainsi que leurs effets sur la résolution. Dans un premier temps, nous analysons l'impact de la provenance des variables du modèle SAT sur l'heuristique de branchement les solveurs SAT. En effet, certaines variables du modèle SAT correspondent directement à des variables du modèle CSP alors que d'autres, dites auxiliaires, sont ajoutées lors de l'encodage des contraintes du modèle CSP. Nous travaillons également sur l'encodage de mécanismes de propagation propres aux solveurs CSP dans le modèle SAT. Afin d'encoder au mieux le maximum d'informations provenant du mécanisme de propagation CSP sans pour autant augmenter de manière démesurée la taille de l'instance SAT, nous proposons de nouveaux encodages SAT comme par exemple l'Abacus encoding.

Problèmes quantifiés sous contraintes [2] Il existe plusieurs formalismes et langages de modélisation issus de la programmation par contraintes comme CSP et CHR (Constraint Handling Rules). Les CSP ont donné lieu à plusieurs extensions dont les QCSP (Quanfitied Constraint Satisfaction Problem) qui adressent les problèmes quantifiés sous contraintes. Néanmoins les formalismes quantifiés de la littérature sont à horizon borné ce qui rend difficile voire impossible la modélisation de certains problèmes quantifiés qui nécessite une représentation en intention et non en extension de la combinatoire. C'est pourquoi nous proposons une approche étendant CHR pour intégrer la quantification. Ceci nous permet de modéliser des problèmes à horizon non borné.

Constraint Handling Rules [2] La programmation par contraintes tire ses racines de la programmation logique et peut être vue comme une forme de programmation logique qui incorpore des contraintes. Cette variante de la programmation logique, appelée communément programmation logique par contraintes, a été implémentée dans différents systèmes (`Prolog III`, `CLP (R)` et `CHIP`). Ces systèmes sont aujourd'hui supplantés par les solveurs PPC plus efficaces. Toutefois, lorsque l'on souhaite innover, il est nécessaire de sortir du carcan d'un solveur PPC, tâche difficile avec les solveurs actuels. C'est pourquoi depuis quelques années nous cherchons un langage de modélisation de contraintes d'un autre niveau d'abstraction. Notre choix s'est porté sur CHR (Constraint Handling Rules). CHR est un langage de modélisation et de programmation de haut niveau basé sur des règles de propagation descriptives et applicables. Les axiomes du langage sont de plus bas niveau que les contraintes habituelles d'un solveur PPC, mais elles permettent de redéfinir facilement et rapidement chacune d'elles. CHR est implémen-

tée comme extension d'un langage hôte et est disponible pour `Prolog`, `Haskell`, `Java` ... C'est pourquoi nous développons `CHR++` [1], notre propre implémentation de CHR au dessus de `C++`. `CHR++` implémente un mécanisme d'exploration arborescent avec retour arrière permettant la recherche d'une solution comme un solveur PPC. Il allie à la fois la puissance de modélisation et la simplicité d'utilisation d'un solveur PPC, la souplesse d'un langage de programmation logique et la performance d'un langage impératif. Nous avons utilisé la logique linéaire pour formaliser le système, et travaillons à étendre le langage par l'ajout de concepts tels que : la disjonction hiérarchique, les contraintes *bang* ou les contraintes permanentes. Nos travaux sont ensuite appliqués à divers domaines et applications comme les QCSP, l'ASP, la transformation de modèles et la conception d'emplois du temps.

Modélisation et résolution hybrides Actuellement, les travaux liés à la frontière entre la programmation logique non-monotone (et de son langage phare, ASP) et la résolution de problèmes combinatoires contraints sur les domaines finis ne considère qu'un unique point de vue consistant à faire coopérer à haut niveau un solveur ASP et un solveur de contraintes. Un axe que nous explorons est d'appliquer des mécanismes de la propagation de contraintes à l'ASP.

Applications en cours

- *Ingénierie dirigée par les modèles [12, 8]* Cette méthode consiste à abstraire les différentes parties d'un système sous la forme d'un ensemble de modèles, chaque modèle adoptant un niveau d'abstraction différent et adapté aux besoins de la personne les utilisant. Nous avons défini une approche permettant d'intégrer des contraintes dans des transformations de modèles, pour spécifier certaines contraintes spécifiques au domaine d'application (par exemple, la planification d'emploi du temps).
- *Emplois du temps universitaires (EDT)*. Le calcul d'EDT se traite généralement par optimisation combinatoire et des travaux récents se sont intéressés à la révision d'EDT pour mieux répondre au caractère dynamique du problème et aux exigences des différents acteurs en termes de contrôle et d'explicabilité du calcul. Notre approche vise à établir un cadre formel pour la révision d'EDT ciblant particulièrement le système universitaire français et qui se fonde sur la programmation par contraintes et possibles hybridations afin d'intégrer opérations et stratégies de réparation.
- L'*Inférence grammaticale* consiste à apprendre automatiquement une grammaire formelle (généralement sous forme de règles de productions ou d'un automate) à partir d'un ensemble d'observations. Nous nous intéressons actuellement à la modélisation SAT et INLP d'automates finis non déterministes devant accepter un ensemble de mots donnés et rejeter un ensemble de mots n'appartenant pas au langage.

1. http://chrpp.barichard.com/

Références

[1] Jean-François Baget, Laurent Garcia, Fabien Garreau, Claire Lefèvre, Swan Rocher, and Igor Stéphan. Bringing existential variables in answer set programming and bringing non-monotony in existential rules : two sides of the same coin. *Annals of Mathematics and Artificial Intelligence*, 82(1-3) :3–41, March 2018.

[2] Vincent Barichard and Igor Stéphan. Quantified Constraint Handling Rules. In

ICLP 2019, volume 306, pages 210–223, Las Cruces, United States, 2019.

[3] Christopher Beatrix, Claire Lefèvre, Laurent Garcia, Igor Stephan, Manuel Carro, Andy King, Neda Saeedloei, and Marina De Vos. Justifications and Blocking Sets in a Rule-Based Answer Set Computation. In *32nd International Conference on Logic Programming (ICLP 2016)*, volume 52, New york, United States, 2016.

[4] Laurent Garcia, Claire Lefevre, Odile Papini, Igor Stephan, and Eric Würbel. A Semantic characterization for ASP base revision. In *Scalable Uncertainty Management*, volume 10564 of *Scalable Uncertainty Management - 11th International Conference, SUM 2017, Granada, Spain, October 4-6, 2017, Proceedings*, pages 334–347, Granada, Spain, October 2017. Springer.

[5] Laurent Garcia, Claire Lefèvre, Odile Papini, Igor Stephan, and Eric Würbel. Possibilistic ASP Base Revision by Certain Input. In Jérome Lang, editor, *International Joint Conference on Artificial Intelligence*, Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 2018.

[6] Laurent Garcia, Claire Lefèvre, Igor Stéphan, and Fabien Garreau. Exists-ASP. In *1st Ontologies and Logic Programming for Query Answering workshop (ONTOLP 2015)*, Buenos Aires, Argentina, 2015.

[7] Laurent Garcia, Claire Lefèvre, Igor Stéphan, Odile Papini, and Eric Würbel. A Semantic Characterization for ASP Base Revision. *Journal of Artificial Intelligence Research*, 66 :989 – 1029, December 2019.

[8] Frédéric Jouault, Valentin Besnard, Théo Le Calvar, Ciprian Teodorov, Matthias Brun, and J. Delatour. Designing, Animating, and Verifying Partial UML Models. In *23rd International Conference on Model Driven Engineering Languages and Systems (MODELS 2020)*, pages 211–217, Virtual event, Canada, October 2020.

[9] Frédéric Lardeux and Eric Monfroy. From Declarative Set Constraint Models to "Good" SAT Instances. In *Artificial Intelligence and Symbolic Computation*, volume 8884 of *Lecture Notes in Computer Science*, pages 76–87, Seville, Spain, December 2014. Springer.

[10] Frédéric Lardeux and Eric Monfroy. Expressively Modeling the Social Golfer Problem into SAT. In *International Conference on Computational Science - ICCS 2015*, Procedia Computer Science, Reykjavík, Iceland, June 2015. Elsevier. In press.

[11] Frédéric Lardeux, Éric Monfroy, Eduardo Rodriguez-Tello, Broderick Crawford, and Ricardo Soto. Solving complex problems using model transformations : from set constraint modeling to SAT instance solving. *Expert Systems with Applications*, 149 :113243, July 2020.

[12] Théo Le Calvar, Fabien Chhel, Frédéric Jouault, and Frédéric Saubion. Toward a Declarative Language to Generate Explorable Sets of Models. In *34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)*, Limassol, Cyprus, April 2019.

[13] Claire Lefèvre, Christopher Beatrix, Igor Stephan, and Laurent Garcia. ASPeRiX, a first-order forward chaining approach for answer set computing. *Theory and Practice of Logic Programming*, 17(3) :266–310, 2017.

[14] Claire Lefevre and Pascal Nicolas. The First Version of a New ASP Solver : ASPeRiX. In *10th International Conference, LPNMR 2009*, volume 5753, pages 522 – 527, Potsdam, Germany, 2009. Springer.

[15] Eric Monfroy, Frédéric Lardeux, Broderick Crawford, and Ricardo Soto. Set constraint model and automated encoding into SAT : application to the social golfer problem. *Annals of Operations Research*, 2015.

[16] Claudia Vasconcellos-Gaete, Vincent Barichard, and Frédéric Lardeux. On the Use of CSP Semantic Information in SAT Models. In *18th Mexican International Conference on Artificial Intelligence (MICAI)*, volume 148 of *Research in Computing Science*, page 127, Xalapa, Mexico, October 2019.

[17] Claudia Vasconcellos-Gaete, Vincent Barichard, and Frédéric Lardeux. Abacus : A New Hybrid Encoding for SAT Problems. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 145–152, Baltimore, United States, November 2020. IEEE.