Une approche SAT incrémentale pour raisonner efficacement sur les réseaux de contraintes qualitatives

Gael Glorian, Jean-Marie Lagniez, Valentin Montmirail, Michael Sioutis

Une approche SAT incrémentale pour raisonner efficacement sur les réseaux de contraintes qualitatives

Gaël Glorian[1*] **Jean-Marie Lagniez**[1] **Valentin Montmirail**[2] **Michael Sioutis**[3]

[1]CRIL, Université d'Artois et CNRS, F62300, Lens, France
[2]I3S, Université Côte d'Azur et CNRS, Nice Sophia-Antipolis, France
[3]Département d'Informatique, Université d'Aalto, Espoo, Finlande
{glorian,lagniez}@cril.fr vmontmirail@i3s.unice.fr michael.sioutis@aalto.fi

Résumé

Le langage $\mathcal{RCC}8$ est un formalisme largement utilisé pour décrire des arrangements topologiques de régions dans l'espace. Deux problèmes fondamentaux sont associés au langage $\mathcal{RCC}8$: la *satisfiabilité* et la *réalisation*. Soit un réseau de contraintes qualitatives (QCN) de $\mathcal{RCC}8$, le problème de satisfiabilité est de décider s'il est possible d'assigner des régions aux variables du QCN de telle sorte que les contraintes soient satisfaites (*solution*). Le problème de réalisation est le fait de produire un modèle spatial qui peut servir de solution. Dans cet article, nous combinons les deux lignes de recherches mentionnés au-dessus et nous explorons l'idée de relier le problème de satisfiabilité et de réalisation. Nous nous limitons aux QCN qui, quand ils sont satisfiables, sont réalisables avec des rectangles. En effet, nous proposons une approche SAT incrémentale afin d'être capable de raisonner sur le langage $\mathcal{RCC}8$ en nous laissant guider par les contre-exemples. Nous avons expérimentalement évalué notre approche et étudié ses performances face aux solveurs de l'état de l'art pour raisonner en $\mathcal{RCC}8$ en utilisant de nombreux ensembles d'instances. Notre approche tient la charge et est compétitive face aux solveurs de l'état de l'art sur les instances considérées.

Abstract

The $\mathcal{RCC}8$ language is a widely-studied formalism for describing topological arrangements of spatial regions. Two fundamental reasoning problems that are associated with $\mathcal{RCC}8$ are the problems of *satisfiability* and *realization*. Given a qualitative constraint network (QCN) of $\mathcal{RCC}8$, the satisfiability problem is deciding whether it is possible to assign regions to the spatial variables of the QCN in such a way that all of its constraints are satisfied (*solution*). The realization problem is producing an actual spatial model that can serve as a solution. In this article, we combine the two aforementioned lines of research and explore the opportunities that surface by interrelating the corresponding reasoning problems, viz., the problems of satisfiability and realization. We restrict ourselves to QCNs that, when satisfiable, are realizable with rectangles. In particular, we propose an *incremental* SAT-based approach for providing a framework that reasons about the $\mathcal{RCC}8$ language in a counterexample-guided manner. We experimentally evaluated our approach and studied its scalability against state-of-the-art solvers for reasoning about $\mathcal{RCC}8$ relations using a varied dataset of instances. The approach scales up and is competitive with the state of the art for the considered benchmarks.

1 Introduction

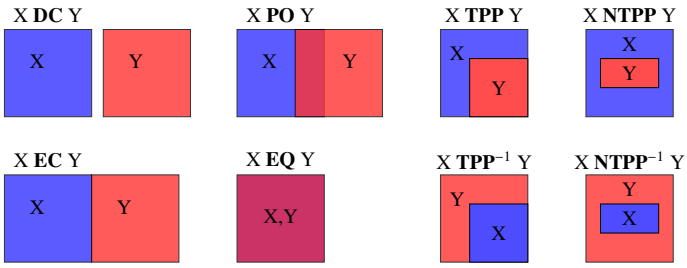
Le raisonnement qualitatif spatial et temporel (QSTR) est un domaine majeur de l'intelligence artificielle, en particulier dans la représentation des connaissances (KR), qui gère les concepts fondamentaux de l'espace et du temps dans un cadre qualitatif, abstrait, comme le fait un humain.

Pour illustrer cela, dans le langage naturel, nous utilisons les expressions telles que *dedans*, *avant*, *au nord de* pour spatialement et temporellement relier un objet à un autre ou à lui même, sans recourir à des informations quantitatives sur ces objets.

Formellement, QSTR limite le vocabulaire très riche des théories mathématiques qui gère l'espace et le temps d'objets à un simple langage de contraintes qualitatives. Ainsi, QSTR fourni un cadre concis qui permet des raisonnements peu coûteux au sujet d'objets localisés dans l'espace et le

*Papier doctorant : Gaël Glorian[1] est auteur principal.

FIGURE 1 – Illustration des relations de base $\mathcal{RCC}8$



temps. Ce cadre améliore la recherche et les applications à une pléthore de domaines incluant (sans être limité à) l'intelligence ambiante, le GIS dynamique, la robotique cognitive, le design spatio-temporel, et la génération de modèles qualitatifs depuis des sources vidéos [3, 25].

Au sujet du raisonnement spatial qualitatif, *Randell et al.* ont développé dans [21] le cadre le plus connu pour les calculs spatiaux dans QSTR, à savoir : Le calcul de connections de régions ($\mathcal{RCC}$). Il étudie les différentes relations qui peuvent être définies entre des régions dans un espace topologique ; ces régions sont basées sur la relation primitive de *connexion*. Par exemple, la relation *déconnectée* entre deux régions X et Y implique qu'aucun des points de la région X n'est connecté à un point de la région Y et réciproquement.

Deux fragments de $\mathcal{RCC}$, à savoir $\mathcal{RCC}8$ et $\mathcal{RCC}5$ (un sous-langage de $\mathcal{RCC}8$) où aucune importance n'est attachée aux frontières des régions), ont été utilisés dans de nombreuses applications de la vie réelle. En particulier, *Bouzy* dans [4] utilise $\mathcal{RCC}8$ dans la programmation du jeu de Go, *Lattner et al.* dans [3] utilise $\mathcal{RCC}5$ pour mettre en place des systèmes d'assistance dans les véhicules intelligents, et *Heintz et al.* dans [12] utilise $\mathcal{RCC}8$ dans le domaine des drones.

$\mathcal{RCC}8$ (qui sera l'objet de cet article) est basé sur les huit relations suivantes : égale (**EQ**), chevauchement partiel (**PO**), connecté à l'extérieur (**EC**), déconnecté (**DC**), la partie propre tangente (**TPP**) et son inverse (**TPP**$^{-1}$), et la partie propre non-tangente (**NTPP**) et son inverse (**NTPP**$^{-1}$). Ces relations spatiales sont illustrées sur la Figure 1.

Soit un réseau de contraintes qualitatives (QCN) défini sur $\mathcal{RCC}8$, nous nous sommes intéressés en particulier au problème de *satisfiabilité*, c'est-à-dire décider s'il existe une interprétation spatiale des variables du QCN qui satisfont ses contraintes. Le problème de satisfiabilité pour $\mathcal{RCC}8$ (et pour $\mathcal{RCC}5$) est NP-complet [23].

Une fois qu'un QCN de $\mathcal{RCC}8$ est connu comme étant satisfiable, c'est-à-dire n'ayant qu'une relation possible pour chacune des arêtes (sans aucun choix possible), nous devons typiquement s'attaquer au *problème de réalisation* (qui est un problème traitable [16]) afin de produire un véritable modèle spatial qui peut servir de solution.

D'autres problèmes de raisonnement fondamentaux que nous pouvons citer sont le problème de *l'étiquetage minimal* (autrement appelé la *fermeture déductive*) et le problème de la *redondance* [22]. Le problème de l'étiquetage minimal consiste à trouver la plus forte contrainte impliquée par le QCN, et le problème de la redondance est de déterminer si une contrainte donnée dans un QCN est impliqué par le reste du réseau (cette contrainte sera appelé *redondante* et sa suppression ne change pas l'ensemble des solutions du QCN). Le problème de la redondance, de l'étiquetage minimale et de satisfiabilité sont tous équivalents calculatoirement sous l'hypothèse de trouver une réduction polynomiale [11].

Les recherches en $\mathcal{RCC}8$ se focalisent habituellement soit sur une vérification symbolique de la satisfiabilité d'un QCN, soit en présentant une méthode pour *réaliser* un QCN satisfiable. À notre connaissance, combiner ces deux lignes de recherche d'une manière inter-reliée n'a pas été considéré dans la littérature. En effet, la première ligne correspond à des méthodes basées sur les contraintes, la seconde correspond à des structures mathématiques beaucoup plus complexes à implémenter.

Dans cet article, nous proposons une approche unifiée et homogène en utilisant une technique SAT incrémentale connue sous le nom de CEGAR, qui signifie **C**ounter-**E**xample **G**uided **A**bstraction **R**efinement [6]. L'idée est la suivante : au lieux de générer une formule propositionnelle équisatisfiable comme le fait l'état-de-l'art [13], nous générons une *sous-abstraction* (une formule qui est sous-contraintes, aussi appelée *relaxation* dans d'autres domaines). Se faisant, si la sous-abstraction n'est pas satisfiable, alors par construction, la formule originale non plus ; sinon, le solveur SAT va fournir en sortie un modèle qui va pouvoir être vérifié. Il se pourrait que l'approche soit chanceuse et que le modèle de la sous-abstraction soit aussi un modèle de la formule originale, et dans ce cas, le problème est décidé. En général, la sous-abstraction est raffinée, *c-à-d*, elle va se rapprocher de la formule originale et, dans le pire cas, devenir équisatisfiable après un nombre fini d'étapes de raffinement.

CEGAR a notamment été proposé dans de nombreux problèmes tel que la vérification bornée de modèles (BMC) [6], la satisfiabilité modulo des théories (SMT) [5], la planification [24] ou plus récemment pour la satisfiabilité minimale en logique modale S5 [14]. CEGAR, de manière globale, peut être vu comme une approche Lazy-SMT [7], où la

connaissance du problème qui est extraite de l'abstraction est utilisée pour guider les étapes des raffinements, au lieu d'un solveur de théorie.

2 Préliminaires

Dans cette section, nous supposons que le lecteur est familier avec les notions de la théorie des graphes et des topologies.

2.1 Le calcul de connexion de régions

Le calcul de connexion de régions ($\mathcal{RCC}$) [21] est une théorie du premier ordre pour représenter et raisonner au sujet d'informations méréotopologiques entre deux régions d'un même espace topologique. Ces relations sont basées sur la relation de connectivité (**C**). En particulier, en utilisant **C**, un ensemble de relations binaires est défini.

De cet ensemble, le fragment $\mathcal{RCC}8$ peut être extrait : {**DC**, **EC**, **PO**, **EQ**, **TPP**, **NTPP**, **TPP**$^{-1}$, **NTPP**$^{-1}$ }. Ces huit relations sont conjointement exhaustives et disjointes par paires, signifiant qu'une seule de ces relations peut être vraie entre deux régions. Comme indiqué dans l'introduction, ce fragment (illustré par la Figure 1), sera référé en tant que $\mathcal{RCC}8$ pour des raisons de facilité.

Nous pouvons voir les régions de $\mathcal{RCC}8$ comme des sous-ensembles non-vides réguliers d'un certain espace topologique qui ne doit pas être intérieurement connecté et n'a pas de dimension particulière, mais elles sont habituellement exigées *close* (*c-à-d*, que les sous-ensembles égalent la fermeture de leurs intérieurs respectifs).

Soit $R(\mathcal{X})$ qui désigne l'ensemble de toutes les régions d'un espace topologique $\mathcal{X}$. Alors, nous pouvons avoir la représentation suivante des relations de $\mathcal{RCC}8$, où $\mathbf{R}_i$ dénote les interprétations de **R** pour deux régions-variables instanciées. Sémantiquement, la relation binaire **R** contient toutes les instanciations possibles de ces paires de régions-variables.

Définition 1 (Notations ensemblistes de $\mathcal{RCC}8$)**.** *Soient deux régions X et Y dans $R(\mathcal{X})$, alors :* [1]

$$\begin{aligned}
\boldsymbol{EQ}_i(X,Y) \quad & ssi \quad X = Y \\
\boldsymbol{DC}_i(X,Y) \quad & ssi \quad X \cap Y = \emptyset \\
\boldsymbol{EC}_i(X,Y) \quad & ssi \quad \mathring{X} \cap \mathring{Y} = \emptyset, X \cap Y \neq \emptyset \\
\boldsymbol{PO}_i(X,Y) \quad & ssi \quad \mathring{X} \cap \mathring{Y} \neq \emptyset, X \not\subseteq Y, Y \not\subseteq X \\
\boldsymbol{TPP}_i(X,Y) \quad & ssi \quad X \subset Y, X \not\subseteq \mathring{Y} \\
\boldsymbol{TPP}_i^{-1}(X,Y) \quad & ssi \quad Y \subset X, Y \not\subseteq \mathring{X} \\
\boldsymbol{NTPP}_i(X,Y) \quad & ssi \quad X \subset \mathring{Y} \\
\boldsymbol{NTPP}_i^{-1}(X,Y) \quad & ssi \quad Y \subset \mathring{X}
\end{aligned}$$

1. $\mathring{A}$ dénote l'intérieur de A

Soient deux relations de base **R** et **S** dans $\mathcal{RCC}8$ qui implique des paires de variables (i, j) et (j, k) respectivement, la *faible composition* de **R** et **S**, dénotée par *CT(**R**,**S**)*, produit la plus forte relation de $\mathcal{RCC}8$ qui contient $\mathbf{R} \circ \mathbf{S}$, *c-à-d*, qui produit le plus petit ensemble de relations de bases tel que, chacun d'entre eux peut être satisfait par une instanciation des variables i et k pour une instanciation possible des variables i, j, k en fonction des relations **R** et **S**. Nous rappelons la définition de la composition faible :

Définition 2 (Composition Faible *CT*)**.** *Soit deux relations de base **R**, **S** de $\mathcal{RCC}8$, leur composition faible CT(**R**,**S**) est défini comme le plus petit sous-ensemble $\{\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_n\}$ de $2^{\mathcal{RCC}8}$ telle que $\boldsymbol{T}_i \cap (\boldsymbol{R} \circ \boldsymbol{S}) \neq \emptyset \; \forall i \in \{1, \ldots, n\}$.*

Le résultat de l'opérateur de composition faible pour chaque paire de relations de base de $\mathcal{RCC}8$ est fourni dans une table, appelée la *la table des compositions faibles* [17] ($\mathcal{RCC}8$ *CT* en plus court), illustrée dans la Table 1. Afin de capturer les informations spatiales qualitatives qui sont impliquées par la base de connaissances des relations $\mathcal{RCC}8$, nous allons utiliser la notion de réseaux de contraintes qualitatives (QCN) défini comme suit :

Définition 3 (Réseaux de contraintes qualitatives (**QCN**))**.** *Un* QCN *de $\mathcal{RCC}8$ est une paire $\mathcal{N} = (V, C)$ où V est un ensemble non-vide fini de variables (chacune correspondant à une région), et C est une application associant une relation $C(v, v') \in 2^{\mathcal{RCC}8}$ avec chaque paire (v, v') of $V \times V$. De plus, C est tel que $C(v, v) \subseteq \{\boldsymbol{EQ}\}$ et $C(v, v') = (C(v', v))^{-1}$.*

Concernant un QCN $\mathcal{N} = (V, C)$, nous avons les définitions suivantes : Une instanciation de V est une application σ de V dans $R(\mathcal{X})$. Une solution (*réalisation*) σ de $\mathcal{N}$ est une instanciation de V telle que pour chaque paire (v, v') des variable dans V, $(\sigma(v), \sigma(v'))$ satisfait $C(v, v')$, c'est-à-dire, qu'il existe une relation de base $b \in C(v, v')$ telle que $(\sigma(v), \sigma(v')) \in b$. $\mathcal{N}$ est satisfiable si et seulement si il admet une solution. Le graphe de contraintes d'un QCN $\mathcal{N}$ est le graphe (V, E), dénoté $G_{\mathcal{N}}$, pour lequel nous avons que $\{v, v'\} \in E$ si et seulement si $C(v, v') \neq \mathcal{RCC}8$ (c'est-à-dire $C(v, v')$ correspond à une relation non-universelle) et $v \neq v'$.

Dans cet article, nous nous restreignons aux QCN qui, quand ils sont satisfiables, sont réalisables avec des rectangles. Comme indiqué dans [19, Exemple 1], il existe des QCN pour lesquels il n'est pas possible de trouver une réalisation rectangulaire en utilisant un seul rectangle par région (cela est toujours possible si plusieurs rectangles par régions sont utilisés). En revanche, pour les instances difficiles à résoudre (ce qui nous concerne ici), qui contiennent des connaissances non-définies et donc, sont plus *flexibles* en terme de réalisations, il est rarement (voire même jamais) le cas qu'une réalisation rectangulaire ne soit pas atteignable pour un QCN satisfiable (voir Section 5).

TABLE 1 – La $\mathcal{RCC}8$ *CT*, où * défini la relation universelle

CT	**DC**	**EC**	**PO**	**TPP**	**NTPP**	**TPP**$^{-1}$	**NTPP**$^{-1}$	**EQ**
DC	*	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC	DC	DC
EC	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP TPP^{-1} EQ	DC EC PO TPP NTPP	EC PO TPP NTPP	PO TPP NTPP	DC EC	DC	EC
PO	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	*	PO TPP NTPP	PO TPP NTPP	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	PO
TPP	DC	DC EC	DC EC PO TPP NTPP	TPP NTPP	NTPP	DC EC PO TPP TPP^{-1} EQ	DC EC PO TPP^{-1} NTPP^{-1}	TPP
NTPP	DC	DC	DC EC PO TPP NTPP	NTPP	NTPP	DC EC PO TPP NTPP	*	NTPP
TPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	EC PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP TPP^{-1} EQ	PO TPP NTPP	TPP^{-1} NTPP^{-1}	NTPP^{-1}	TPP^{-1}
NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP NTPP TPP^{-1} NTPP^{-1} EQ	NTPP^{-1}	NTPP^{-1}	NTPP^{-1}
EQ	DC	EC	PO	TPP	NTPP	TPP^{-1}	NTPP^{-1}	EQ

2.2 Logique Propositionnelle

La logique propositionnelle, dénotée *CPL* permet de raisonner avec ce qui est vrai (`True`) et ce qui est faux (`False`). La syntaxe de *CPL* peut-être formellement défini comme suit :

Définition 4 (Langage de la logique propositionnelle)**.** *Soit $\mathbb{P}$ un ensemble infini dénombrable de variables propositionnelles. Le langage de la logique propositionnelle est l'ensemble des formules contenant $\mathbb{P}$, fermé sous l'ensemble des connecteurs logiques $\{\neg, \wedge\}$.*

Sans perte de généralité, nous supposons que toutes les formules de *CPL* sont en forme normale conjonctive (CNF) car n'importe quelle formule peut être transformée en une CNF équisatisfiable en utilisant l'algorithme de *Tseitin*. Sur les aspects sémantiques de la logique propositionnelle, la notion d'*interprétation* est importante. Elle est défini comme suit :

Définition 5 (Interprétation)**.** *Une interprétation est un ensemble de valuations des variables propositionnelles. Formellement, c'est une application $\mathbb{P} \rightarrow \{\texttt{True}, \texttt{False}\}$.*

Une interprétation est un *modèle* de ϕ si ϕ est vraie pour cette interprétation. Si une formule a *au moins un* modèle $\mathcal{M}$, nous dirons que cette formule est *satisfiable* ; $\mathcal{M} \models \phi$ indiquera que $\mathcal{M}$ satisfait ϕ. Formellement, la relation de satisfiabilité est défini comme suit :

Définition 6 (Relation de satisfiabilité dans *CPL*)**.** *La relation $\models$ entre les interprétations $\mathcal{M}$ et les formules ϕ dans CPL est défini récursivement comme suit :*

$$\begin{array}{lll} \mathcal{M} \models p & ssi & p \in \mathcal{M} \\ \mathcal{M} \models \neg\phi & ssi & \mathcal{M} \not\models \phi \\ \mathcal{M} \models \phi_1 \wedge \phi_2 & ssi & \mathcal{M} \models \phi_1 \textit{ et } \mathcal{M} \models \phi_2 \\ \mathcal{M} \models \phi_1 \vee \phi_2 & ssi & \mathcal{M} \models \phi_1 \textit{ ou } \mathcal{M} \models \phi_2 \end{array}$$

Si une formule est satisfiable par toutes les interprétations, nous dirons que cette formule est *valide* ; dans ce cas, la formule est une *tautologie* et nous l'indiquerons par $\models \phi$. Si une formule est fausse pour toutes les interprétations, nous dirons que cette formule est *insatisfiable*.

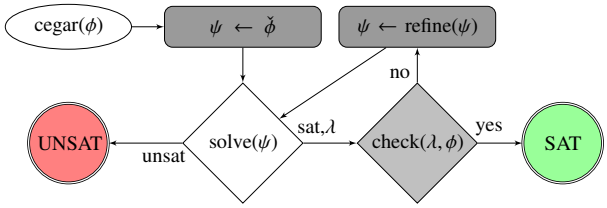
2.3 Préliminaires sur CEGAR

Counter-Example-Guided Abstraction Refinement (CEGAR) est une manière incrémentale de décider la satisfiabilité de formules en logique propositionnelle. Elle a été proposée initialement pour le problème de la vérification de modèles [6], c'est-à-dire, pour répondre à des questions du type "Est-ce que $S \models P$ est vrai ?" ou dit autrement, "Est-ce que $S \wedge \neg P$ est insatisfiable ?" où S décrit un système et P une propriété. Très souvent, dans des problèmes très structurés, seule une petite partie de la formule est nécessaire pour répondre à la question.

La pierre angulaire de CEGAR est de remplacer $\phi = S \wedge \neg P$ par une abstraction ϕ', où ϕ' doit être plus facile à résoudre en pratique que ϕ. Il y a deux types d'abstractions : une sur-abstraction (resp. sous-abstraction) de ϕ est une formule $\hat{\phi}$ (resp. $\check{\phi}$) tel que $\hat{\phi} \models \phi$ (resp. $\phi \models \check{\phi}$) est vraie. $\hat{\phi}$ a au plus, autant de modèles que ϕ et $\check{\phi}$ a au moins autant de modèle que ϕ.

Une illustration de l'approche CEGAR utilisant des sous-abstractions est donnée en Figure 2. Pour être correct, complet et pour terminer, une approche CEGAR doit vérifier

FIGURE 2 – Le cadre CEGAR avec des sous-abstractions



certaines hypothèses (les preuves peuvent être obtenus dans [15, Theo. 1,2 and 3])) :

1. "solve" est correcte, complete, et termine ;
2. si $\check{\phi}$ est insatisfiable, alors ϕ est insatisfiable ;
3. "check(λ, ϕ)" retourne vrai si λ est un modèle de ϕ ;
4. $\exists$ n $\in \mathbb{N}$ tel que refine$^n(\check{\phi})$ est équisatisfiable avec ϕ.

Comme nous utilisons un solveur SAT dans notre approche, nous supposons que le solveur SAT intégré est bien codé, qu'il est correct, complet et qu'il termine. Ainsi l'hypothèse (1) de CEGAR est satisfaite.

Maintenant que nous avons introduit les notions nécessaires à la compréhensions des contributions, la section suivante détaille la première d'entre elles, qui est l'encodage de $\mathcal{RCC}8$ en logique propositionnelle d'une telle manière qu'elle va nous permettre de vérifier facilement les hypothèses (2) et (4) de CEGAR.

3 Encoder RCC8 en SAT

Pour obtenir un encodage SAT du problème de satisfiabilité en $\mathcal{RCC}8$, nous devons définir comment traduire chacune des relations possibles. Nous allons représenter une région i comme un ensemble de quatre variables $\{x_i^-, y_i^-, x_i^+, y_i^+\}$ comme illustré sur la Figure 3.

Tous les cas possibles pour chacune des relations peuvent être trouvés et sont prouvés, ainsi que leurs liens avec l'algèbre des points, dans [18, Table 6.2]. De cette table, nous pouvons proposer l'encodage SAT suivant pour toutes les arrêtes possibles :

Définition 7 (Encodage SAT – tr)**.** *Pour toutes les relations R dans toutes les arrêtes (i, j) du problème $\mathcal{N}$ en entrée, nous avons :*

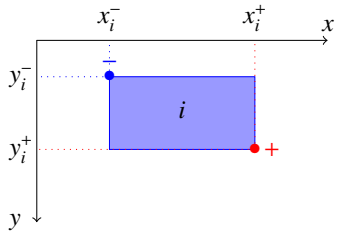
$$\texttt{tr}(\mathcal{N}) := \bigwedge_{\forall (R,i,j) \in \mathcal{N}} \texttt{tr}(R, i, j)$$

Ainsi, depuis [18, Table 6.2], si nous voulons traduire par exemple, la relation **EC** entre les nœuds i et j (la procédure est similaire pour les autres relations $\mathcal{RCC}8$), nous allons avoir l'encodage SAT suivant comme indiqué dans la Définition 7 :

Définition 8 (Encodage SAT de **EC** sur l'arête i-j)**.**

$$\texttt{tr}(\boldsymbol{EC}, i, j) := \boldsymbol{EC}(i, j) \rightarrow (\boldsymbol{EC}_r(i, j) \vee \boldsymbol{EC}_l(i, j) \vee \boldsymbol{EC}_u(i, j) \vee \boldsymbol{EC}_d(i, j))$$

FIGURE 3 – Illustration de la représentation d'une région



De cette définition, nous pouvons voir que la relation **EC** pour l'arête (i, j) ne peut être satisfaite que de quatre manière différentes, à savoir : *par la gauche*, *par la droite*, *par le haut* et *par le bas*. Chacun des cas est défini de la manière suivante :

$$\begin{aligned}
\mathbf{EC}_r(i,j) \rightarrow & ((x_i^- < x_j^-) \wedge (x_i^- < x_j^+)) \quad \wedge \\
& ((x_i^- = x_j^+) \wedge (x_i^+ < x_j^+)) \quad \wedge \\
& ((y_i^- < y_j^+) \vee (y_i^- < y_j^+)) \quad \wedge \\
& ((y_i^+ < y_j^-) \vee (y_i^+ < y_j^-))
\end{aligned}$$

$$\begin{aligned}
\mathbf{EC}_u(i,j) \rightarrow & ((x_i^- < x_j^-) \vee (x_i^- = x_j^-)) \quad \wedge \\
& ((x_i^- > x_j^+) \vee (x_i^- = x_j^+)) \quad \wedge \\
& ((y_i^- < y_j^-) \wedge (y_i^- < y_i^+)) \quad \wedge \\
& ((y_i^+ = y_j^-) \wedge (y_i^+ < y_j^+))
\end{aligned}$$

$$\begin{aligned}
\mathbf{EC}_l(i,j) \rightarrow & ((x_i^- > x_j^-) \wedge (x_i^- = x_j^+)) \quad \wedge \\
& ((x_i^- > x_j^+) \wedge (x_i^+ > x_j^+)) \quad \wedge \\
& ((y_i^- < y_j^+) \vee (y_i^- < y_j^+)) \quad \wedge \\
& ((y_i^+ < y_j^-) \vee (y_i^+ < y_j^-))
\end{aligned}$$

$$\begin{aligned}
\mathbf{EC}_d(i,j) \rightarrow & ((x_i^- < x_j^-) \vee (x_i^- = x_j^-)) \quad \wedge \\
& ((x_i^- > x_j^+) \vee (x_i^- = x_j^+)) \quad \wedge \\
& ((y_i^- > y_j^-) \wedge (y_i^- = y_i^+)) \quad \wedge \\
& ((y_i^+ > y_j^-) \wedge (y_i^+ > y_j^+))
\end{aligned}$$

Les relations inverses sont définies comme d'habitude : $\mathbf{TPP}^{-1}(i, j) = \mathbf{TPP}(j, i)$ et $\mathbf{NTPP}^{-1}(i, j) = \mathbf{NTPP}(j, i)$. Pour chaque nœud dans le QCN avec N nœuds que nous essayons de résoudre, nous allons ajouter la contrainte suivante qui permet de garantir que les coordonnées des points sont dans le bon ordre :

$$\bigwedge_{i=1}^{N} ((x_i^- < x_i^+) \wedge (y_i^- < y_i^+))$$

Nous voulons aussi indiquer que, si la variable propositionnelle $(A < B)$ est vraie, alors les variables $(A > B)$ et

$(A = B)$ sont fausses. Pour exprimer cela, nous ajoutons les clauses suivantes :

$$\texttt{AMO} := \bigwedge_{a\in\{x,y\}} \bigwedge_{c_1\in\{-,+\}} \bigwedge_{c_2\in\{-,+\}} \bigwedge_{i=1}^{N} \bigwedge_{j=1}^{N} \begin{pmatrix} ((a_i^{c_1} < a_j^{c_2}) \vee (a_i^{c_1} = a_j^{c_2}) \vee (a_i^{c_1} > a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} < a_j^{c_2}) \vee \neg(a_i^{c_1} = a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} < a_j^{c_2}) \vee \neg(a_i^{c_1} > a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} = a_j^{c_2}) \vee \neg(a_i^{c_1} > a_j^{c_2})) \wedge \end{pmatrix} \quad (1)$$

Grâce à l'équation 1 (AMO – At Most One), nous pouvons ainsi remplacer, par exemple, dans **EC** (i,j) (u), $(x_i^- < x_j^+) \vee (x_i^- = x_j^+)$ par $\neg(x_i^- > x_j^+)$. De la même manière, pour toutes les disjonctions dans [18, Table 6.2].

Enfin, nous voulons aussi assurer la transitivité des relations sur toutes les coordonnées possibles. Cela aura l'impact le plus important sur la taille de la CNF générée. Pour chaque triplet (i, j, k) dans la triangulation du graphe de contraintes de l'entrée QCN, nous devons ajouter les règles suivantes pour toutes combinaisons de $(c1, c2)$ qui assurent la transitivité $\in \{(-,-),(-,+),(+,-),(+,+)\}$ et pour les deux axes $a \in \{x, y\}$:

$$\texttt{transitivity}(i,j,k) := \bigwedge \begin{pmatrix} ((a_i^{c1} = a_j^{c1}) \wedge (a_j^{c1} = a_k^{c2})) \rightarrow (a_i^{c1} = a_k^{c2}) \\ ((a_i^{c1} < a_j^{c1}) \wedge \neg(a_j^{c1} > a_k^{c2})) \rightarrow (a_i^{c1} < a_k^{c2}) \\ ((a_i^{c1} > a_j^{c1}) \wedge \neg(a_j^{c1} < a_k^{c2})) \rightarrow (a_i^{c1} > a_k^{c2}) \\ ((a_j^{c1} > a_k^{c2}) \wedge \neg(a_i^{c1} < a_j^{c1})) \rightarrow (a_i^{c1} > a_k^{c2}) \\ ((a_j^{c1} < a_k^{c2}) \wedge \neg(a_i^{c1} > a_j^{c1})) \rightarrow (a_i^{c1} < a_k^{c2}) \end{pmatrix}$$

Nous ne rentrerons pas dans le détail sur la manière de trianguler un graphe, c'est une procédure standard. Il est à noter que trianguler un graphe prend un temps linéaire dans la taille du graphe triangulé en sortie. Avant de passer à la partie CEGAR, nous devons prouver que notre encodage est correct et complet.

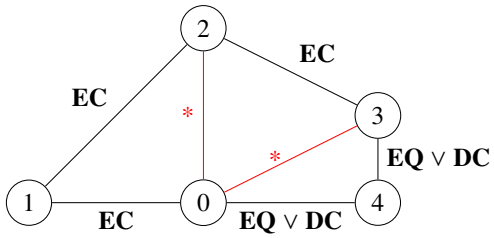
Théorème 1 (Preuve dans [10])**.** *Soit $\mathcal{N} = (V, C)$ un* QCN *de $\mathcal{RCC}8$, et G un super-graphe, du graphe de contraintes de $\mathcal{N}$, triangulé. Si* $\texttt{toSAT}(\mathcal{N})$ *est défini comme suit :*

$$\texttt{toSAT}(\mathcal{N}) := \texttt{tr}(\mathcal{N}) \wedge \texttt{AMO} \wedge \bigwedge_{i=1}^{N} ((x_i^- < x_i^+) \wedge (y_i^- < y_i^+)) \wedge \bigwedge_{(i,j,k)\in G} \texttt{transitivity}(i,j,k)$$

alors $\texttt{toSAT}(\mathcal{N})$ *est équisatisfiable à $\mathcal{N}$.*

Nous venons juste de voir que si nous encodons chacune des transitivités pour chaque triangle dans le graphe triangulé, nous obtenons une formule propositionnelle équisatisfiable. Cependant, quand nous regardons de plus près,

FIGURE 4 – Un exemple de graphe de contraintes d'un problème $\mathcal{RCC}8$ ϕ, (les labels sur les arrêtes dénotent les relations $\mathcal{RCC}8$ correspondantes), en rouge, les arrêtes ajoutées lors de la triangulation du graphe.



nous pouvons voir que cela est très coûteux en temps, la fonction `transitivity` est exactement ce que nous voulons éviter à tout prix. C'est donc pourquoi nous proposons une approche CEGAR pour l'éviter.

4 Traduire parcimonieusement les contraintes de transitivité

Comme expliqué précédemment, la fonction `transitivity` peut être très coûteuse et par conséquent, doit être évitée pour avoir une approche compétitive. C'est exactement l'hypothèse sur laquelle notre approche CEGAR repose. Prenons l'exemple illustré sur la Figure 4, un problème $\mathcal{RCC}8$ donné en entrée avec 5 nœuds et 5 relations (les relations sont mises en caractères **gras** sur la Figure 4)). Ainsi, quand nous traduisons ce problème en logique propositionnelle, nous obtenons les règles suivantes :

$$\begin{aligned} \texttt{under}(\phi) = \bigwedge_{i=0}^{4} ((x_i^- < x_i^+) \wedge (y_i^- < y_i^+)) \\ \wedge \texttt{AMO} \wedge EC_{0,1} \wedge EC_{1,2} \wedge EC_{2,3} \\ \wedge (EQ_{3,4} \vee DC_{3,4}) \wedge (EQ_{4,0} \vee DC_{4,0}) \\ \wedge \texttt{tr}(\mathbf{EC}, 0, 1) \wedge \texttt{tr}(\mathbf{EC}, 1, 2) \\ \wedge \texttt{tr}(\mathbf{EC}, 2, 3) \wedge \texttt{tr}(\mathbf{EC}, 0, 1) \\ \wedge \texttt{tr}(\mathbf{EQ}, 3, 4) \wedge \texttt{tr}(\mathbf{DC}, 3, 4) \\ \wedge \texttt{tr}(\mathbf{EQ}, 4, 0) \wedge \texttt{tr}(\mathbf{DC}, 4, 0) \end{aligned}$$

Théorème 2. *Soit ϕ un* QCN*, alors* $\texttt{under}(\phi)$ *est une sous-abstraction de ϕ (c-à-d, il a au moins autant de modèles).*

Démonstration. $\texttt{under}(\phi)$ est un sous-ensemble des clauses de l'encodage équisatisfiable (Théorème 1). Ainsi si under(ϕ) est insatisfiable, alors ϕ est aussi insatisfiable par définition de la conjonction logique. □

À ce point, la traduction est une sous-abstraction du problème original, c-à-d, si elle est insatisfiable, alors il est sûr que le problème est insatisfiable, mais un modèle de cette

traduction n'implique pas forcément être un modèle du problème original. L'hypothèse CEGAR (2) est respectée par construction de cette traduction, pour obtenir l'équisatisfiabilité, nous avons besoin de :

$$\begin{aligned}\text{toSAT}(\phi) = \ & \texttt{under}(\phi) \\ & \wedge \texttt{transitivity}(0,1,2) \\ & \wedge \texttt{transitivity}(0,2,3) \\ & \wedge \texttt{transitivity}(0,3,4)\end{aligned}$$

Comme le nombre de triangles dans le graphe triangulé est borné par un nombre N (pire des cas : $N = |V|^3$ quand le graphe est complet), nous pouvons facilement voir qu'après la traduction de la transitivité pour chaque triangle (une opération que nous appellerons maintenant un raffinement), nous pouvons raffiner le problème N fois et obtenir une formule équisatisfiable. Ceci permet de respecter l'hypothèse CEGAR (4).

À propos de l'hypothèse CEGAR (4), nous avons besoin d'une manière efficace pour vérifier si le modèle de la sous-abstraction retournée est aussi un modèle de la formule originale. Pour cela, nous utilisons l'algorithme *Directional Path Consistency* (DPC) présenté dans [8, 20, 27]. Notre fonction check réalise une vérification de modèle et retourne les triangles qui ne peuvent pas être validés par le modèle (qui retourne l'ensemble vide sur leurs relations). À partir de là, si le vérificateur retourne le triangle (i, j, k) et que nous ajoutons ensuite les contraintes de transitivité `transitivity`(i, j, k) dans la formule propositionnelle, alors il est impossible pour le vérificateur de retourner une fois de plus le même triangle. Comme discuté plus tôt, l'ensemble maximal de contraintes de transitivité que nous pouvons ajouter est borné, nous obtenons donc dans le pire des cas, une formule équisatisfiable.

Nous avons maintenant toutes les pièces du puzzle pour créer deux manières différentes pour résoudre le problème de satisfiabilité et de réalisation en $\mathcal{RCC}8$. La première consiste à utiliser un encodage direct (avec la fonction `toSAT`$(\mathcal{N})$). La seconde est d'utiliser une approche CEGAR, comme celle présenté dans l'Algorithme 1, qui dans le pire des cas (le cas où toutes les transitivités ont été ajoutées) va finir comme une version plus lente de l'encodage direct ; Cependant, cela ne s'est jamais produit lors de nos expérimentations (voir Section 5). De plus, à chaque fois qu'une instance a été satisfiable, nous avons pu obtenir une réalisation. En d'autres termes, nous avons résolu le problème de satisfiabilité et de réalisation ensemble.

Algorithme 1 : CEGAR-$\mathcal{RCC}8(\mathcal{N})$

Data : $\mathcal{N}$=(V,C) with n variables
Result : Une réalisation de $\mathcal{N}$ s'il est possible d'en obtenir une, UNSAT sinon

```
1  G ← (V, E ← E(G_N));
2  setOfTriangles ← Triangule(G);
3  transitivity ← ⊤;
4  ψ ← under(N) ;  // étape de sous-abstraction
5  while (setOfTriangle ≠ ∅) do
6      λ ← SAT-Solver(ψ ∧ transitivity) ;  // étape de résolution
7      if (λ = ⊥) then return UNSAT ;
8      res ← check(λ, N) ;                // étape de vérification
9      if (res = null) then return interpret(λ) ;
10     else
11         setOfTriangle.remove(res);
12         transitivity ← transitivity
               ∧transitivity(res) ;       // étape de raffinement
13 λ ← SAT-Solver(ψ ∧ transitivity) ;      // pire cas: équisatisfiable
14 if (λ = ⊥) then return UNSAT ;
15 else return interpret(λ) ;
```

5 Résultats Expérimentaux

Maintenant que nous avons notre nouvel encodage SAT et une approche CEGAR pour résoudre les problèmes de satisfiabilité et de réalisation en $\mathcal{RCC}8$, nous voulons comparer cela à l'état de l'art. Pour cela, nous avons implémenté nos approches dans le solveur Churchill [2] et nous avons utilisé Glucose [1, 9] en solveur SAT interne.

Nous comparons Churchill en encodage direct et en mode CEGAR face aux solveurs de l'état de l'art pour le raisonnement spatio-qualitatif en $\mathcal{RCC}8$, que sont : GQR , Renz-Nebel01 , RCC8SAT, PPyRCC8, et Chordal-Phalanx. Chaque solveur est utilisé dans son mode par défaut, à partir GQR pour lequel nous utilisons le flag "-c horn". En utilisant ce flag, GQR décompose une relation $\mathcal{RCC}8$ en sous-relations de Horn (qui est le comportement standard pour le reste des solveurs) ; Cela change son facteur de branchement de 4 à $\sim$ 1.4. De plus, PPyRCC8 et Chordal-Phalanx utilisent `PyPy`, comme recommandé par leurs auteurs, afin d'améliorer les performances globales. Nous comparons ces solveurs sur quatre catégories de benchmarks générés aléatoirement de différentes manières (voir [10] pour en savoir plus).

À propos des ensembles 3 et 4, *scale-free networks* sont des réseaux dont la distribution des degrés suit une loi de puissance [2], ces réseaux structurés ont été utilisés récemment [26]. Les expérimentations ont été réalisées sur un cluster de Xeon 4 cœurs cadencés à 3.3 GHz sous CentOS 7.0 avec une limite mémoire de 32 Go et une limite de temps de 900 secondes par solveur par instance. En vérifiant les

2. Le nom vient du personnage historique qui prenait beaucoup de CEGAR

FIGURE 5 – Distribution des temps d'exécution sur le premier ensemble

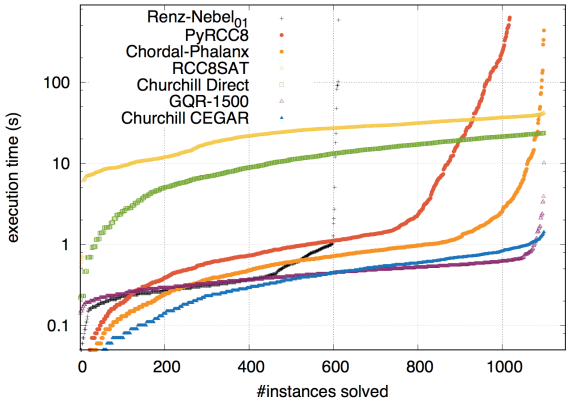
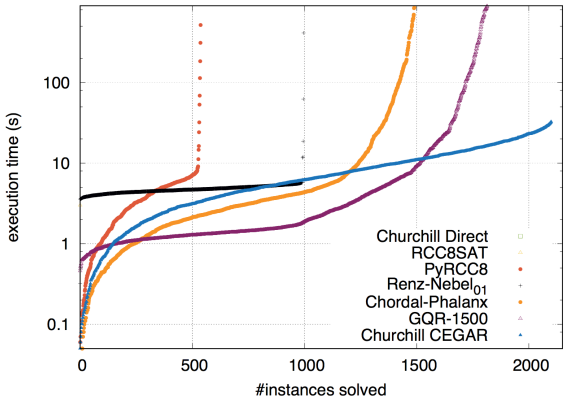


FIGURE 6 – Distribution des temps d'exécution sur le deuxième ensemble



réponses, comme aucune contradiction n'est apparue (tous les solveurs ont répondu la même chose), nous avons conclu que le cas où un QCN est satisfiable mais pas réalisable avec des rectangles n'est pas apparu dans nos ensembles de tests.

En regardant les Figures 5 et 6 qui indiquent les distributions des temps de résolutions des différents solveurs sur le premier et le deuxième ensemble d'instances, nous pouvons constater que Churchill et GQR sont extrêmement rapides pour résoudre ces ensembles. En effet, cela prend au maximum 1.42 seconde à Churchill pour résoudre la plus dure des instances du premier ensemble, 10.10 secondes pour GQR et 32.70 secondes pour le deuxième ensemble pour Churchill. Pour Churchill, cette rapidité vient du fait que nous effectuons en moyenne qu'un petit nombre de boucles CEGAR (moy : 8.90 pour le premier ensemble et 11.87 pour le deuxième ensemble). Cependant quand nous regardons les traductions directes (RCC8SAT et Churchill Direct), tenir dans la mémoire autorisée est difficile à partir de 500 nœuds.

La table 3 montre le nombre d'instances résolues pour les ensembles 3 et 4. Les meilleurs résultats d'une ligne sont présentés en gras et le nombre d'instances qui ne peuvent pas être résolues dû à la limite mémoire est indiqué entre parenthèses (si aucun dépassement de mémoire, un trait est affiché). La ligne VBS représente le *Virtual Best Solver* (une borne supérieure pratique représentant les performances atteignables en choisissant à chaque fois les résultats du meilleur solveur pour chaque instance).

Sur le troisième ensemble, nous pouvons voir la mise à l'échelle de l'approche CEGAR face à l'encodage direct (Churchill Direct) ou via une représentation CP. Les résultats sont clairs, quand le nombre de nœuds devient trop important, les approches SAT demandent trop de mémoire ou trop de temps pour effectuer la traduction et donc, deviennent inefficaces. Plus le réseau est gros, plus Churchill CEGAR prend de temps pour vérifier le modèle retourné par le solveur SAT et plus cela prend de mémoire pour ajouter les contraintes de transitivités. Dans certains cas, nous atteignons la limite de mémoire et sommes incapable de résoudre l'instance.

Sur le quatrième ensemble, nous étudions la mise à l'échelle sur des instances très difficiles mais de taille raisonnable. Nous pouvons voir ici que les approches basées sur SAT sont en difficulté avec la taille de l'entrée et qu'utiliser une approche CEGAR au lieu d'un encodage direct permet d'améliorer grandement les performances. En effet, Churchill CEGAR a réussi à résoudre toutes les instances, mais, malheureusement, il prend plus de temps que Chordal-Phalanx dans beaucoup de cas (médian : 37.97s pour Churchill contre 16.78s pour Chordal-Phalanx) ; Cependant il est plus rapide dans le pire cas (max : 163.10s pour Churchill contre 714.12s pour Chordal-Phalanx). Cela est clairement dû au fait que vérifier les modèles plusieurs fois, ce qui est typiquement le cas quand le réseau a une taille entre 2 000 et 3 500 nœuds, prend énormément de temps. En effet, dans la Table 2, un résumé du découpage en trois étapes du temps de résolution de Churchill : la triangulation, la vérification des modèles et la résolution des problèmes SAT est donné. Quand nous analysons les résultats donnés dans la Table 2, les résultats sont clairs : quand le réseau est petit (1er et 2ème ensemble) la plupart du temps est passé dans la triangulation du graphe. Quand le réseau est gros (3ème et 4ème ensemble) la plupart du temps est passé dans la vérification des modèles. Dans tous les cas, le solveur SAT n'est jamais coûteux. Pour chaque résultat, il est important de se rappeler que nous résolvons les problèmes de satisfiabilité et de réalisation, pas simplement la satisfiabilité comme les autres solveurs.

6 Conclusion

Dans ce papier, une nouvelle approche pour résoudre les problèmes de satisfiabilité et de réalisation en $\mathcal{RCC}8$

TABLE 2 – Résumé des temps des trois étapes dans Churchill

Temps (s)	Triangulation			Vérification			Résolution		
	min	med	max	min	med	max	min	med	max
Premier Ensemble	0.220	0.390	0.630	0.015	0.030	0.151	0.002	0.003	0.010
Deuxième Ensemble	3.910	12.910	20.153	0.430	1.170	2.057	0.015	0.030	0.370
Troisième Ensemble	8.708	55.96	128.96	7.900	222.3	668.57	0.015	0.860	16.38
Quatrième Ensemble	3.765	21.530	68.230	0.560	24.410	58.950	0.012	0.250	15.73

TABLE 3 – Résultats sur les ensembles 3 (gauche) et 4 (droit)

	random-scale-free-like-instances						*random-scale-free-like-np8-instances*						
#Nœuds (x1000)	< 6	6	7	8	9	10	0.5	1	1.5	2	2.5	3	3.5
#Instances	150	30	30	30	30	30	10	10	10	10	10	10	10
Renz-Nebel01	0 -	0 -	0 -	0 -	0 -	0 -	0 -	0 -	0 -	0 -	0 -	0 -	0 -
RCC8SAT	0 (150)	0 (30)	0 (30)	0 (30)	0 (30)	0 (30)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)
PPyRCC8	134 (14)	19 (8)	20 (7)	21 (7)	23 (7)	23 (4)	**10** -	9 -	**10** -	8 -	7 (1)	3 (5)	3 (7)
Chordal-Phalanx	**150** -	**30** -	**30** -	**30** -	**30** -	**30** -	**10** -	**10** -	**10** -	**10** -	**10** -	**10** -	**10** -
GQR-1500	**150** -	**30** -	**30** -	**30** -	**30** -	**30** -	**10** -	**10** -	9 -	6 -	**10** -	8 -	9 -
Churchill Direct	0 (150)	0 (30)	0 (30)	0 (30)	0 (30)	0 (30)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)	0 (10)
Churchill CEGAR	**150** -	**30** -	**30** -	18 (10)	8 (20)	6 (24)	**10** -	**10** -	**10** -	**10** -	**10** -	**10** -	**10** -
VBS	150	30	30	30	30	30	10	10	10	10	10	10	10

en utilisant une approche basée sur les sous-abstractions dans le cadre CEGAR a été proposée. Nous avons démontré que notre encodage est correct et complet pour les QCN satisfiables qui sont réalisables avec des rectangles et nous l'avons instancié au travers du solveur Churchill.

Nous avons comparé notre approche face aux solveurs représentant, à notre connaissance, l'état de l'art pour la résolution pratique en $\mathcal{RCC}8$ dans un large éventail d'instances de différentes tailles et difficultés. Nous avons conclu qu'un encodage basique de notre approche n'est pas du tout compétitif, beaucoup des instances disponibles sont de taille importante et demande un grand nombre de contraintes de transitivités dans l'encodage SAT. Cependant, notre approche CEGAR mixant à la fois des courts-circuits SAT et UNSAT surpasse les autres solveurs sur les benchmarks considérés.

Comme travaux futurs, en considérant la table 2, nous pouvons améliorer notre vérificateur de modèle. En effet, éviter de vérifier les sous-graphes non modifiés en mettant des marqueurs sur certains nœuds, c'est-à-dire, en ne vérifiant que les parties qui ont été modifiées par la précédente assignation. De plus, afin de rendre notre approche correcte, même en dehors de réalisation rectangulaire, nous pouvons étendre l'approche CEGAR en une approche RECAR [15] où la sur-abstraction serait de considérer des formes de plus en plus complexes (d'abord des points, ensuite des rectangles, ensuite des rectangles qui permettent un trou au milieu, puis deux trous, etc.)

7 Remerciements

Les auteurs aimeraient remercier les relecteurs de *CP'18* pour leurs commentaires pertinents. Une partie de ce travail a été supportée par le Ministère de l'Enseignement Supérieur et la Recherche, par le projet ANR Investissement d'Avenir UCA$^{\text{JEDI}}$ (ANR-15-IDEX-01) et par le Conseil Régional des Haut-de-France au travers du "Contrat de Plan État Région (CPER) DATA".

Références

[1] Gilles Audemard, Jean-Marie Lagniez et Laurent Simon : Improving Glucose for Incremental SAT Solving with Assumptions : Application to MUS Extraction. *In Proc. of SAT'13*, 2013.

[2] Albert-László Barabási et Réka Albert : Emergence of Scaling in Random Networks. *Science*, 286(5439), 1999.

[3] Mehul Bhatt, Hans W. Guesgen, Stefan Wölfl et Shyamanta Moni Hazarika : Qualitative Spatial and Temporal Reasoning : Emerging Applications, Trends, and Directions. *Spatial Cognition & Computation*, 11(1):1–14, 2011.

[4] Bruno Bouzy : Les concepts spatiaux dans la programmation du go. *Revue d'Intelligence Artificielle*, 15(2):143–172, 2001.

[5] Robert Brummayer et Armin Biere : Effective Bit-Width and Under-Approximation. *In Proc. of EUROCAST'09*, volume 5717 de *LNCS*. Springer, 2009.

[6] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu et Helmut Veith : CounterExample-Guided Abstraction Refinement For Symbolic Model Checking. *Journal of ACM*, 50(5), 2003.

[7] Leonardo Mendonça de Moura, Harald Ruess et Maria Sorea : Lazy Theorem Proving for Bounded Model Checking over Infinite Domains. *In Proc. of CADE'02*, 2002.

[8] Rina Dechter, Itay Meiri et Judea Pearl : Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3), 1991.

[9] Niklas Eén et Niklas Sörensson : An Extensible SAT-solver. *In Proc. of SAT'03*, volume 2919 de *LNCS*. Springer, 2003.

[10] Gael Glorian, Jean-Marie Lagniez, Valentin Montmirail et Michael Sioutis : An Incremental SAT-Based Approach to Reason Efficiently on Qualitative Constraint Networks. *In Proc. of CP'18*, volume 11008, pages 160–178. Springer, 2018.

[11] Martin Charles Golumbic et Ron Shamir : Complexity and algorithms for reasoning about time : A graph-theoretic approach. *J. ACM*, 40(5):1108–1133, 1993.

[12] Fredrik Heintz et Daniel de Leng : Spatio-Temporal Stream Reasoning with Incomplete Spatial Information. *In Proc. of ECAI'14*, volume 263, pages 429–434. IOS Press, 2014.

[13] Jinbo Huang, Jason Jingshi Li et Jochen Renz : Decomposition and tractability in qualitative spatial and temporal reasoning. *Artificial Intelligence*, 195, 2013.

[14] Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima et Valentin Montmirail : An Assumption-Based Approach for Solving the Minimal S5-Satisfiability Problem. *In Proc. of IJCAR'18*, volume 10900, pages 1–18. Springer, 2018.

[15] Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima et Valentin Montmirail : A Recursive Shortcut for CEGAR : Application To The Modal Logic K Satisfiability Problem. *In Proc. of IJCAI'17*, 2017.

[16] Sanjiang Li : On Topological Consistency and Realization. *Constraints*, 11(1):31–51, 2006.

[17] Sanjiang Li et Mingsheng Ying : Region Connection Calculus : Its models and composition table. *Artif. Intell.*, 145(1-2):121–146, 2003.

[18] Zhiguo Long : *Qualitative Spatial And Temporal Representation And Reasoning : Efficiency in Time And Space*. Thèse de doctorat, University of Technology Sydney (UTS), January 2017.

[19] Zhiguo Long, Steven Schockaert et Sanjiang Li : Encoding Large RCC8 Scenarios Using Rectangular Pseudo-Solutions. *In Proc. of KR'16*, 2016.

[20] Zhiguo Long, Michael Sioutis et Sanjiang Li : Efficient Path Consistency Algorithm for Large Qualitative Constraint Networks. *In Proc. of IJCAI'16*, 2016.

[21] David A. Randell, Zhan Cui et Anthony G. Cohn : An interval logic for space based on "connection". *In ECAI*, pages 394–398, 1992.

[22] Jochen Renz : Qualitative spatial and temporal reasoning : Efficient algorithms for everyone. *In Proc. ocf IJCAI'07*, pages 526–531, 2007.

[23] Jochen Renz et Bernhard Nebel : On the Complexity of Qualitative Spatial Reasoning : A Maximal Tractable Fragment of the Region Connection Calculus. *Artificial Intelligence*, 108(1-2), 1999.

[24] Jendrik Seipp et Malte Helmert : Counterexample-Guided Cartesian Abstraction Refinement. *In Proc. of ICAPS'13*. AAAI, 2013.

[25] Michael Sioutis, Marjan Alirezaie, Jennifer Renoux et Amy Loutfi : Towards a synergy of qualitative spatio-temporal reasoning and smart environments for assisting the elderly at home. *In IJCAI Workshop on Qualitative Reasoning*, pages 901–907, 2017.

[26] Michael Sioutis, Jean-François Condotta et Manolis Koubarakis : An Efficient Approach for Tackling Large Real World Qualitative Spatial Networks. *IJAIT*, 25(2):1–33, 2016.

[27] Michael Sioutis, Zhiguo Long et Sanjiang Li : Leveraging Variable Elimination for Efficiently Reasoning about Qualitative Constraints. *IJAIT*, 27(4):1860001, 2018.