

Studying the use and effect of graph decomposition in qualitative spatial and temporal reasoning

MICHAEL SIOUTIS, YAKOUB SALHI and JEAN-FRANÇOIS CONDOTTA

Université Lille-Nord de France, Artois, CRIL-CNRS UMR 8188, Rue Jean Souvraz – SP 18, F 62307 Lens, France;
e-mail: fsioutis@cril.fr, salhi@cril.fr, condotta@cril.fr

Abstract

We survey the use and effect of decomposition-based techniques in qualitative spatial and temporal constraint-based reasoning, and clarify the notions of a tree decomposition, a chordal graph, and a partitioning graph, and their implication with a particular constraint property that has been extensively used in the literature, namely, patchwork. As a consequence, we prove that a recently proposed decomposition-based approach that was presented in the study by Nikolaou and Koubarakis for checking the satisfiability of qualitative spatial constraint networks lacks soundness. Therefore, the approach becomes quite controversial as it does not seem to offer any technical advance at all, while results of an experimental evaluation of it in a following work presented in the study by Sioutis become questionable. Finally, we present a particular tree decomposition that is based on the biconnected components of the constraint graph of a given large network, and show that it allows for cost-free utilization of parallelism for a qualitative constraint language that has patchwork for satisfiable atomic networks.

1 Introduction

Qualitative spatial and temporal reasoning (QSTR) is a major field of study in artificial intelligence and, particularly, in knowledge representation. This field studies representations of space and time that abstract from numerical quantities. The concise expressiveness of the representational languages used in the qualitative approach provides a promising framework that boosts research and applications in a plethora of areas and domains, such as ambient intelligence, dynamic Geographic Information System (GIS), cognitive robotics, and spatiotemporal design (Bhatt *et al.*, 2011; Hazarika, 2012).

The Interval Algebra (IA) (Allen, 1981) and a fragment of the Region Connection Calculus (RCC) (Randell *et al.*, 1992), namely RCC-8, are the dominant artificial intelligence approaches for representing and reasoning about qualitative temporal and topological relations, respectively. These qualitative calculi use constraints to encode knowledge about the spatial or temporal relationships between entities in an abstract manner. Thus, the problem of reasoning about qualitative information can be modeled as an infinite-domain variant of a constraint satisfaction problem (CSP) (Montanari, 1974), for which we use the term *qualitative constraint network* (QCN). For instance, there are infinitely many time points or temporal intervals in the timeline and infinitely many regions in a two- or three-dimensional space. One way of dealing with infinite domains is using constraints over a finite set of binary relations, called *base relations* (or *atoms*), by employing a *relation algebra* (Ladkin & Maddux, 1994), which is the approach we follow here.

Given a QCN over a set of variables corresponding to a set of spatial or temporal entities, we are particularly interested in its *satisfiability problem*, that is, deciding whether there exists an interpretation of all the variables of the QCN such that all of its constraints are satisfied by this interpretation; such an

interpretation being called a *solution*. The satisfiability problem is closely related to the *minimal labeling problem* (MLP) (Liu & Li, 2012; Amaneddine *et al.*, 2013) and the *redundancy problem* (Duckham *et al.*, 2014; Li *et al.*, 2015), in the sense that the latter two problems exhibit functions that build on the core algorithms used to obtain a solution of a QCN. In particular, the MLP is the problem of determining all the base relations for each of the constraints of a QCN that participate in at least one solution of the QCN, whilst the redundancy problem is the problem of obtaining all the constraints of a QCN that do not contain at least one base relation participating in a solution of the modified QCN that results by removing these constraints (these constraints being referred to as non-redundant constraints, since their removal changes the solution set of the QCN). As such, we will emphasize on the satisfiability problem throughout the paper, and make the link whenever there is a work related to the MLP or the redundancy problem whose results are affected by the notions and techniques that we will survey here. The satisfiability problem in IA and RCC-8 is \mathcal{NP} -complete in general (Nebel & Bürckert, 1995; Renz & Nebel, 1999). However, there exist large maximal tractable subclasses of relations of IA and RCC-8, which can be used to make reasoning much more efficient even in the general \mathcal{NP} -complete case (Nebel, 1997; Renz & Nebel, 2001). In recent years, many works surfaced that use graph decomposition to significantly improve the efficiency and scalability of practical reasoning (Li *et al.*, 2009; Chmeiss & Condotta, 2011; Condotta & D’Almeida, 2011; Sioutis & Koubarakis, 2012; Amaneddine *et al.*, 2013; Huang *et al.*, 2013; Westphal *et al.*, 2013; Nikolaou & Koubarakis, 2014; Sioutis & Condotta, 2014; Westphal & Hué, 2014; Sioutis *et al.*, 2016). All these works make use of a particular constraint property, namely, patchwork (Lutz & Milicic, 2007; Huang, 2012). Intuitively, patchwork ensures that the combination of two satisfiable QCNs that completely agree on the constraints between their common variables continues to be satisfiable.

The contribution of this paper is threefold:

1. We recall the notions of a tree decomposition, a chordal graph, and a partitioning graph that have been used in the literature, and clarify the relationship between one another, and also their implication with patchwork.
2. Consequently, we show that the approach proposed in Nikolaou and Koubarakis (2014) for efficiently checking the satisfiability of qualitative spatial constraint networks violates patchwork in two ways, namely, both in the complete agreement between two satisfiable QCNs and in the graph decomposition that is obtained, and, therefore, lacks soundness.
3. Finally, we present a particular tree decomposition that is based on the biconnected components of the constraint graph of a given large QCN, and show that it allows for sound and cost-free utilization of parallelism for a qualitative constraint language that has patchwork for satisfiable atomic QCNs and that it can significantly decongest search when using it for solving intractable QCNs¹.

As such, our paper can be viewed as a survey on the use and effect of graph decomposition in QSTR, as a response paper to Nikolaou and Koubarakis (2014), and partially to Sioutis (2014), and also as a report of an original decomposition approach that paves the way for efficient utilization of parallelism.

The paper is organized as follows. In Section 2, we recall the definition of a QCN, along with the definition of the property of patchwork. Section 3 introduces the notions of a tree decomposition and a chordal graph, and the way they are interrelated and used in the literature. In Section 4, we present the definition of a partitioning graph (Nikolaou & Koubarakis, 2014), and prove that it yields non-soundness when used solely with patchwork. In Section 5, we introduce a particular tree decomposition that allows for cost-free utilization of parallelism for a qualitative constraint language that has patchwork for satisfiable atomic QCNs. In Section 6, we briefly describe QCNs in the context of CSPs, and review some related work in the constraint programming community that could inspire future research in the QSTR community. Finally, in Section 7 we make a discussion and conclude our work.

¹ In what follows, a tractable (resp. intractable) QCN will be a QCN whose satisfiability problem is tractable (resp. intractable).

2 Preliminaries

A (binary) qualitative temporal or spatial constraint language is based on a finite set B of *jointly exhaustive and pairwise-disjoint* relations defined on a domain D (Ladkin & Maddux, 1994), called the set of base relations. The base relations of the set B of a particular qualitative constraint language can be used to represent the definite knowledge between any two entities with respect to the given level of granularity. B contains the identity relation Id , and is closed under the converse operation ($^{-1}$). Indefinite knowledge can be specified by a union of possible base relations, and is represented by the set containing them. Hence, 2^B represents the total set of relations. 2^B is equipped with the usual set-theoretic operations (union and intersection), the converse operation, and the weak composition operation denoted by \diamond (Renz & Ligozat, 2005). The converse of a relation is the union of the converses of its base relations, in particular, for every $r \in 2^B$, we have that $r^{-1} = \{b^{-1} \mid b \in r\}$. The weak composition (\diamond) of two base relations $b, b' \in B$ is defined as the strongest relation $r \in 2^B$ that contains $b \circ b'$, or, formally, $b \diamond b' = \{b'' \in B \mid b'' \cap (b \circ b') \neq \emptyset\}$, where $b \circ b' = \{(x, y) \in D \times D \mid \exists z \in D \text{ such that } (x, z) \in b \wedge (z, y) \in b'\}$ is the relational composition of b and b' . For every $r, r' \in 2^B$, we have that $r \diamond r' = \{b \diamond b' \mid b \in r, b' \in r'\}$.

The set of base relations of RCC-8 (Randell *et al.*, 1992) is the set $\{dc, ec, po, tpp, ntpp, tppi, ntppi, eq\}$. These eight relations represent the binary topological relations between *regions* that are non-empty regular closed subsets of some topological space, that is, for any spatial region X we have that $X = c(i(X))$, where $i(\cdot)$ specifies the topological interior of a spatial region and $c(\cdot)$ the topological closure (Renz, 2002a). The eight base relations of RCC-8 are depicted in Figure 1 (for the two-dimensional case). The set of base relations of IA (Allen 1981) is the set $\{eq, p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$. These 13 relations represent the possible relations between *time intervals*, as depicted in Figure 2.

The weak composition operation \diamond along with the converse operation $^{-1}$, and the total set of relations 2^B along with the identity relation Id of a qualitative constraint language, form an algebraic structure $(2^B, \text{Id}, \diamond, ^{-1})$ that can correspond to a *relation algebra* in the sense of Tarski (1941). This topic has been extensively discussed in Dylla *et al.* (2013). In fact, Dylla *et al.* (2013) summarizes findings on the relationship between relation algebras and some well-known qualitative constraint languages into the following result:

PROPERTY 1 (Dylla *et al.*, 2013) *Each of the qualitative constraint languages of Point Algebra (Vilain *et al.*, 1990), Cardinal Direction Calculus (Frank, 1991; Ligozat, 1998), IA (Allen, 1981), Block Algebra (Balbiani *et al.*, 2002), and RCC-8 (Randell *et al.*, 1992) is a relation algebra with the algebraic structure $(2^B, \text{Id}, \diamond, ^{-1})$.*

Networks of IA and RCC-8 can be modeled as particular instances of QCNs, with relation eq being the identity relation Id in both cases. A QCN is formally defined as follows:

DEFINITION 1 *A QCN is a tuple (V, C) where:*

- $V = \{v_1, \dots, v_n\}$ is a non-empty finite set of variables corresponding to a set of spatial or temporal entities.
- C is a mapping that associates a relation $r \in 2^B$ with each pair (v, v') of $V \times V$, that relation being denoted by $C(v, v')$. Further, mapping C is such that $C(v, v) = \{\text{Id}\}$ and $C(v, v') = (C(v', v))^{-1}$ for every $v, v' \in V$.

Note that we always regard a QCN as a complete network. In what follows, given a QCN $\mathcal{N} = (V, C)$ and $v, v' \in V$, the relation $C(v, v')$ will sometimes be denoted by $C_{vv'}$ for simplicity. The constraint graph of a QCN $\mathcal{N} = (V, C)$ is the graph (V, E) , denoted by $G(\mathcal{N})$, for which we have that $(v, v') \in E$ iff $C(v, v') \neq B$ and $v \neq v'$. (B corresponds to the universal relation, that is, the non-restrictive² relation that contains all base relations, thus, it does not really pose a constraint.) Given a QCN $\mathcal{N} = (V, C)$, \mathcal{N} is said to be *trivially inconsistent* iff $\exists v, v' \in V$ with $C(v, v') = \emptyset$. Further, $\mathcal{N} \downarrow v'$, with $V' \subseteq V$, is the QCN \mathcal{N} restricted to V' . A *solution* of \mathcal{N} is a mapping σ defined from V to the domain D , yielding a valid

² The result of the weak composition between any relation and the universal relation is the universal relation.

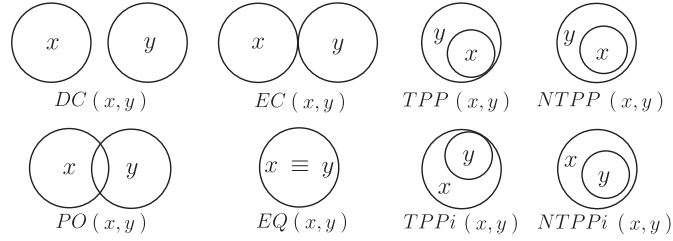


Figure 1 The base relations of the Region Connection Calculus (RCC)-8 constraint language

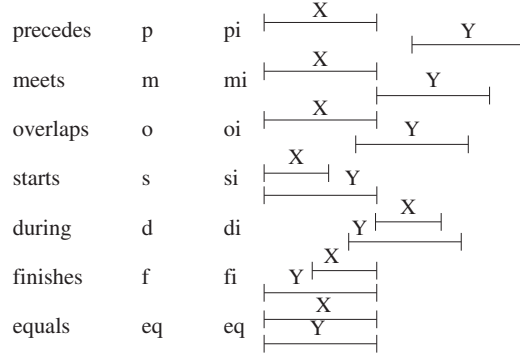


Figure 2 The base relations of the Interval Algebra constraint language

configuration, such that $\forall v, v' \in V$ we have that $(\sigma(v), \sigma(v'))$ can be described by $C(v, v')$, that is, there exists a base relation $b \in C(v, v')$ such that $(\sigma(v), \sigma(v')) \in b$, viz. $(\sigma(v), \sigma(v'))$ satisfies base relation b .

DEFINITION 2 A QCN \mathcal{N} is satisfiable iff it admits a solution. The satisfiability problem is the problem of determining if \mathcal{N} is satisfiable.

A sub-QCN³ \mathcal{N}' of $\mathcal{N}=(V, C)$, denoted by $\mathcal{N}' \subseteq \mathcal{N}$, is a QCN (V, C') such that $C'(v, v') \subseteq C(v, v')$ $\forall v, v' \in V$. If b is a base relation, $\{b\}$ is a singleton relation. An *atomic* QCN is a QCN where each constraint is a singleton relation. A *scenario* \mathcal{S} of \mathcal{N} is a satisfiable atomic sub-QCN of \mathcal{N} . Given a QCN $\mathcal{N}=(V, C)$, a base relation $b \in C(v, v')$, with $v, v' \in V$, is *feasible* (resp. *infeasible*) iff there exists (resp. there does not exist) a scenario $\mathcal{S}=(V, C')$ of \mathcal{N} such that $C'(v, v') = \{b\}$.

DEFINITION 3 A QCN $\mathcal{N}=(V, C)$ is minimal iff $\forall v, v' \in V$ and $\forall b \in C(v, v')$, b is a feasible base relation of \mathcal{N} . The MLP is the problem of determining all the feasible base relations for each of the constraints of \mathcal{N} .

Given a QCN $\mathcal{N}=(V, C)$, we say that a relation $C(v, v')$, with $v, v' \in V$, is non-redundant in \mathcal{N} , if there exists a base relation $b \notin C(v, v')$ and a solution σ of the QCN $\mathcal{N}'=(V, C')$ defined by $C'(v, v') = B \setminus C(v, v')$, $C'(v', v) = B \setminus (C(v, v'))^{-1}$, and $C'(u, w) = C(u, w) \forall (u, w) \in (V \times V) \setminus \{(v, v'), (v', v)\}$ such that $(\sigma(v), \sigma(v')) \in b$. The relation is called non-redundant, because if we were to remove it and effectively replace it with relation B , the solution set of \mathcal{N} would be changed. Note that by definition every universal relation B in a QCN is redundant.

DEFINITION 4 A QCN $\mathcal{N}=(V, C)$ is reducible iff it contains a redundant relation other than relation B , and irreducible otherwise. The redundancy problem is the problem of obtaining all the non-redundant relations in \mathcal{N} and, hence, determining if \mathcal{N} is reducible.

Checking the satisfiability of a QCN is \mathcal{NP} -complete in the general case for the most well known and interesting calculi, such as RCC-8 (Renz & Nebel, 1999) and IA (Nebel & Bürckert, 1995). As a direct

³ This term is also found by the name ‘refined QCN’ throughout the literature.

consequence, checking if a base relation of a QCN is feasible, or if a constraint of a QCN is non-redundant, is also \mathcal{NP} -complete in the general case. However, there exist *maximal tractable subclasses* $\mathcal{A} \subseteq 2^{\mathcal{B}}$ of the considered calculi, for which the satisfiability problem becomes tractable through the use of a *path-consistency*⁴ algorithm.

With respect to subclasses of relations we have the following definition:

DEFINITION 5 *A subclass of relations is a subset $\mathcal{A} \subseteq 2^{\mathcal{B}}$ that contains the singleton relations of $2^{\mathcal{B}}$ and is closed under converse, intersection, and weak composition. A subclass $\mathcal{A} \subseteq 2^{\mathcal{B}}$ is tractable iff a QCN defined over \mathcal{A} is tractable (i.e. its satisfiability problem is tractable). A tractable subclass $\mathcal{A} \subseteq 2^{\mathcal{B}}$ is maximal iff there exists no tractable subclass $\mathcal{A}' \subseteq 2^{\mathcal{B}}$ with $\mathcal{A} \subset \mathcal{A}'$.*

Further, the notion of path consistency for QCNs is defined as follows:

DEFINITION 6 *A QCN $\mathcal{N}=(V, C)$ is path consistent iff $\forall v, v', v'' \in V$ we have that $C(v, v') \subseteq C(v, v'') \diamond C(v'', v')$.*

Given a QCN $\mathcal{N}=(V, C)$, path consistency can be applied on \mathcal{N} in $O(|V|^3)$ time (Vilain *et al.*, 1990). For the calculi in this paper, we have that not trivially inconsistent and path-consistent QCNs defined over a maximal tractable subclass $\mathcal{A} \subseteq 2^{\mathcal{B}}$ are satisfiable (Nebel & Bürckert, 1995; Renz & Nebel, 1999)⁵. The maximal tractable subclasses of relations of RCC-8 and IA are the classes $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and \mathcal{Q}_8 (Renz & Nebel, 2001) and \mathcal{H}_{IA} (Nebel, 1997), respectively. Classes $\hat{\mathcal{H}}_8$ and \mathcal{H}_{IA} contain exactly those relations that are transformed to propositional Horn formulas when using the propositional encodings of RCC-8 and IA, respectively. Further, and for RCC-8 in particular, if we denote by \mathcal{P}_8 the set of relations that belong to either one of the classes $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and \mathcal{Q}_8 , then all relations of \mathcal{P}_8 not contained in \mathcal{C}_8 contain EC and all relations of \mathcal{P}_8 not contained in \mathcal{Q}_8 contain EQ (Renz, 1999). The propositional encoding of either \mathcal{C}_8 or \mathcal{Q}_8 is neither a Horn formula nor a Krom formula, but classes \mathcal{C}_8 and \mathcal{Q}_8 themselves are directly related to class $\hat{\mathcal{H}}_8$ in the sense that any QCN defined over either \mathcal{C}_8 or \mathcal{Q}_8 can be polynomially refined to a QCN defined over $\hat{\mathcal{H}}_8$ (Renz, 1999).

Given two QCNs $\mathcal{N}=(V, C)$ and $\mathcal{N}'=(V', C')$, we have that $\mathcal{N} \cup \mathcal{N}'$ yields the QCN $\mathcal{N}''=(V'', C'')$, where $V'' = V \cup V'$, $C''(u, v) = C''(v, u) = \mathbf{B}$ for all $(u, v) \in (V \setminus V') \times (V' \setminus V)$, $C''(u, v) = C(u, v) \cap C'(u, v)$ for every $u, v \in V \cap V'$, $C''(u, v) = C(u, v)$ for all $(u, v) \in (V \times V) \setminus (V' \times V')$, and $C''(u, v) = C'(u, v)$ for all $(u, v) \in (V' \times V') \setminus (V \times V)$.

We now recall the definition of the *patchwork* property that was originally introduced in Lutz and Milicic (2007) and was shown to be satisfied by IA and RCC-8 for satisfiable atomic QCNs of their relations.

DEFINITION 7 *A constraint language has patchwork, iff for any finite satisfiable constraint networks $\mathcal{N}=(V, C)$ and $\mathcal{N}'=(V', C')$ defined on this language where $\forall u, v \in V \cap V'$ we have that $C(u, v) = C'(u, v)$, the constraint network $\mathcal{N} \cup \mathcal{N}'$ is satisfiable.*

Huang showed that IA and RCC-8 have patchwork for certain satisfiable non-atomic QCNs of their relations as well (Huang, 2012). In particular, we have the following proposition:

PROPOSITION 1 (Huang, 2012) *The qualitative constraint languages of IA and RCC-8 have patchwork for not trivially inconsistent and path-consistent QCNs defined over one of the maximal tractable subclasses \mathcal{H}_{IA} and $\hat{\mathcal{H}}_8, \mathcal{C}_8$, or \mathcal{Q}_8 , respectively.*

Other qualitative constraint languages known to have patchwork for some subclass of their relations are listed in Huang (2012). Notably, those languages include the languages listed in Property 1.

⁴ The literature suggests the term *algebraic closure* (Renz & Ligozat, 2005) instead, which is equivalent to a path-consistency algorithm where the weak composition operator \diamond is used instead of the relational composition operator \circ (Renz & Ligozat, 2005), so we will use this more traditional term throughout the paper.

⁵ Some of the cited works are based on encodings of QCNs into Boolean formulas. However, the Boolean formulas are constructed in such a way that each solution of a formula corresponds to a not trivially inconsistent and path-consistent QCN defined over some maximal tractable subclass of relations, and vice versa.

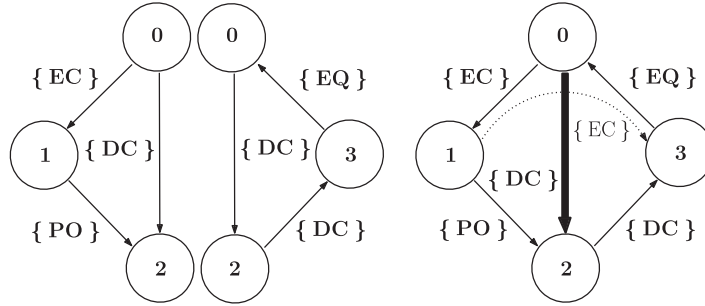


Figure 3 Patching two qualitative constraint networks of Region Connection Calculus (RCC)-8 that agree on their common part

Intuitively, patchwork ensures that the combination of two satisfiable constraint networks that agree on their common part, that is, on the constraints between their common variables, continues to be satisfiable. As an example, we can view the two QCNs of RCC-8 in Figure 3. (Self-loops corresponding to singleton relation $\{EQ\}$ and converses of constraints are not shown for simplicity.) The QCNs of RCC-8 are atomic, as they comprise singleton relations, and are also path consistent, therefore, by Proposition 1 and application of the patchwork property, their union is satisfiable, since they agree on the constraints between their common variables, namely, on $C_{02} = \{DC\}$. (The universal relation that exists by definition between variables 1 and 3 in the unified QCN would result to relation $C_{13} = \{EC\}$ if we were to calculate it, but it is not necessary to do so unless required by the specifics of the use case at hand.)

Patchwork is closely related to the *global consistency* property (Renz & Ligozat, 2005), which is defined as follows in Dechter (2003):

DEFINITION 8 A QCN $\mathcal{N}=(V, C)$ is *globally consistent* if and only if, for any $V' \subset V$, every partial solution on V' can be extended to a partial solution on $V' \cup \{v\} \subseteq V$, for any $v \in V \setminus V'$.

In particular, global consistency implies patchwork, but the opposite is not true. For example, even though RCC-8 has patchwork (Huang, 2012), it does not have global consistency (Renz & Ligozat, 2005). For instance, let us consider the spatial configuration shown in Figure 4(a). Region y is a doughnut, and region x is externally connected to it, by occupying its hole. Further, region z is externally connected to region y . With respect to RCC-8, we know that the constraint network defined by the set of constraints $\{EC(x, y), EC(y, z), EC(x, z)\}$ is satisfiable, as it is path consistent and atomic. However, the valuation of region variables x and y is such that it is impossible to extend it with a valuation of region variable z so that $EC(x, z)$ may hold. Patchwork allows us to disregard any partial valuations and focus on the satisfiability of the network. Then, we can consider a valuation that satisfies the constraint network. Such a valuation is, for example, the one presented in Figure 4(b) along with its corresponding scenario.

3 Tree decomposition and chordal graph

The utility of taking advantage of the structure of QCNs has already been addressed in the context of heuristics for the path-consistency algorithms in Beek and Manchak (1996) and Renz (2002b). In particular, the heuristics proposed in Renz (2002b) target the *denser* parts of the underlying constraint graph of a given QCN, that is, the qualitative relations that consist of few base relations, as an effort to propagate constraints more efficiently and also possibly resolve any local inconsistencies faster. Pruning infeasible base relations off a qualitative relation that already comprises very few base relations can almost immediately unveil an inconsistency. However, these heuristics always consider a complete underlying constraint graph of a given network. As such, they fail to completely isolate parts of the underlying constraint graph of a given QCN that are irrelevant to the process of satisfiability checking; such parts being universal relations that do not belong to the clusters of a tree decomposition corresponding to the constraint graph of the QCN at hand.

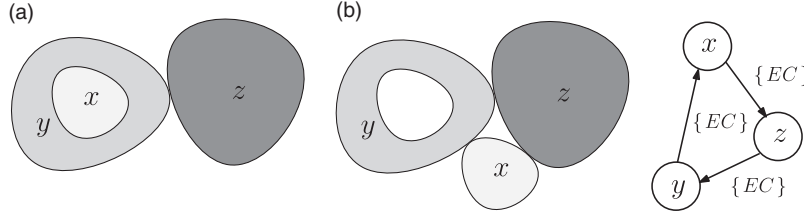


Figure 4 Region Connection Calculus (RCC)-8 configurations. (a) a configuration where $EC(x,z)$ does not hold; (b) a configuration where $EC(x,z)$ holds, along with its corresponding scenario

In this section, we recall the notions of a tree decomposition and a chordal graph, and review their use and effect in QSTR in combination with patchwork⁶. We will show that it is possible to omit satisfiability checks across clusters of a tree decomposition corresponding to the constraint graph of a QCN.

In what follows, we consult the book of Diestel on *Graph Theory* for related definitions and properties (Diestel, 2012). A tree decomposition is formally defined as follows:

DEFINITION 9 A tree decomposition of a graph $G = (V, E)$ is a tuple (T, X) where $T = (I, F)$ is a tree and $X = \{X_i \subseteq V \mid i \in I\}$ a collection of clusters (subsets of V) that satisfy the following properties:

- For every $v \in V$ there is at least one node $i \in I$ such that $v \in X_i$.
- For every $(u, v) \in E$ there exists a node $i \in I$ such that both $u, v \in X_i$.
- Let i_1, i_2, i_3 be three nodes in I such that i_2 lies on the path between i_1 and i_3 in T . Then, if $v \in V$ belongs to both X_{i_1} and X_{i_3} , v must also belong to X_{i_2} .

Let us view the example presented in Figure 5. In the upper part of the figure we can view a graph $G = (V, E)$, which can correspond to the structure of a constraint graph of a QCN. For the moment, we consider only the solid edges to be part of G and we disregard the dashed edges (3, 4) and (4, 5). A tree decomposition of G comprises a tree $T = (I, F)$ and a cluster X_i for every node $i \in I$ of that tree as shown in the lower part of the figure, for example, $X_a = \{0, 1, 2\}$.

Tree decompositions have been explicitly introduced in qualitative reasoning by Condotta and D’Almeida (2011), and implicitly by Li *et al.* (2009) and Huang *et al.* (2013). (The work presented in Huang *et al.* (2013) properly contains the work presented in Li *et al.* (2009), thus, we will stick to the former reference in what follows.)

In Condotta and D’Almeida (2011), the authors apply path consistency on the clusters of a tree decomposition of the constraint graph of a QCN. The graphs induced by the clusters of the tree decomposition are completed with the introduction of a new set of edges, called *fill edges*, that correspond to the universal relation for a QCN. These fill edges for the example graph of Figure 5 are edges (3, 4) and (4, 5). As such, the clusters of the tree decomposition are considered to be *cliques*, namely, sets of vertices such that every two vertices in a set are connected by an edge. This is done for two reasons: (i) by definition path consistency considers all triples of variables of a given constraint network and, hence, involves a complete graph, and (ii) the common vertices between any two complete graphs induce a complete graph, thus, the corresponding constraint networks will completely agree on the constraints between their common variables and the patchwork property can be used. The patchwork property is then considered to patch together the not trivially inconsistent and path-consistent *pre-convex* QCNs of IA (Ligozat, 2011) that correspond to the graphs induced by the clusters of the tree decomposition in a tree-like manner and construct a satisfiable network.

In Huang *et al.* (2013), the authors enlist a structure known as a *dtree* (decomposition tree), which, as the name suggests, is very close to a tree decomposition. Without going further into detail, a dtree is a full binary tree where the root represents a given graph and for each non-leaf node its two children represent a partitioning of the parent graph into two subgraphs. Thus, although a dtree is not a tree decomposition, it provides a way to construct a tree decomposition out of a given graph. A dtree and a tree decomposition are therefore equivalent

⁶ Some of the cited works use a property called *amalgamation*, which is equivalent to patchwork for satisfiable atomic networks when the satisfiability of atomic networks can be decided by path consistency.

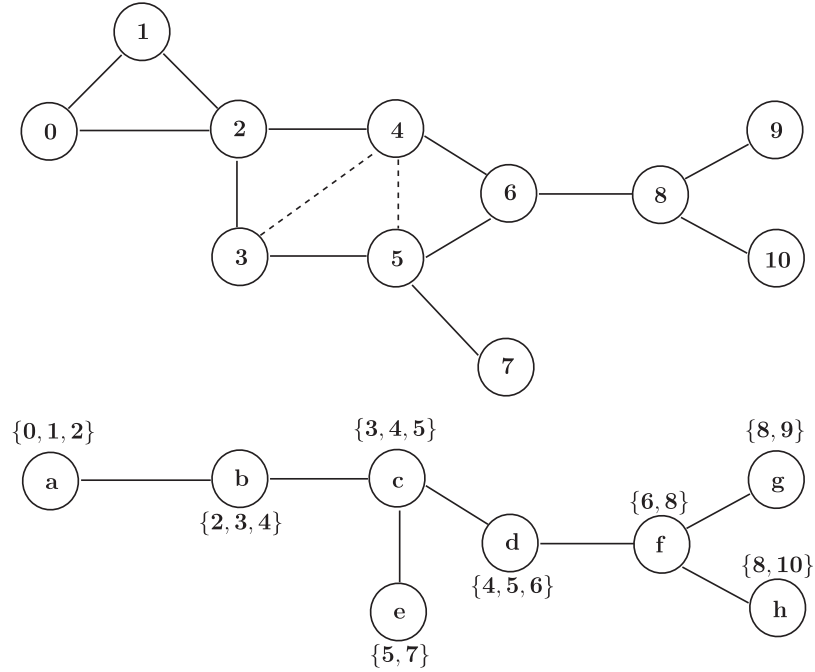


Figure 5 A graph (upper part) and its tree decomposition (lower part)

in the context of qualitative reasoning, since omitting path consistency checks across children of dtree nodes (as described in Huang *et al.* (2013)) corresponds to omitting those checks across clusters of the tree decomposition into which the dtree is converted, as has been specifically pointed out in Condotta and D’Almeida (2011). Similar to what is done in Condotta and D’Almeida (2011), children of dtree nodes are treated as cliques, and the patchwork property is considered to patch together the path-consistent atomic QCNs of either IA or RCC-8 in a tree-like recursive manner and construct a satisfiable network.

The observant reader will note that it would be convenient to operate directly on a tree decomposition (T, X) of a given graph G where X would be a collection of cliques. In this context, chordal graphs become relevant. Formally, a chordal graph is defined as follows:

DEFINITION 10 *A graph G is said to be chordal (or triangulated) iff every cycle of length > 3 has a chord, viz., an edge of G which is not in the edge set of the cycle and whose endpoints lie in the vertex set of the cycle.*

We then have the following proposition:

PROPOSITION 2 (Diestel, 2012) *A graph G is chordal if and only if it has a tree decomposition $(T, \{X_1, \dots, X_n\})$ where cluster X_i is a clique of G for every $i \in \{1, \dots, n\}$.*

For example, the graph presented in Figure 5, with the dashed edges included, is chordal. Chordality checking can be done in (linear) $O(|V|+|E|)$ time for a given graph $G = (V, E)$ with the *maximum cardinality search* algorithm, which also constructs an *elimination ordering* ω as a by-product (Tarjan & Yannakakis, 1984). If a graph is not chordal, it can be made so through the addition of fill edges. This process is usually called *triangulation* of a given graph $G = (V, E)$ and can run as fast as in $O(|V|+(|E \cup F(\omega)|))$ time, where $F(\omega)$ is the set of fill edges that results by following the elimination ordering ω , eliminating the nodes one by one, and connecting all nodes in the neighborhood of each eliminated node, thus, making it simplicial in the resulting subgraph. If the graph is already chordal, following the elimination ordering ω produced by the maximum cardinality search algorithm guarantees that no fill edges are added, that is, ω is actually a *perfect elimination ordering* (Diestel, 2012). For example, a perfect elimination ordering for the chordal graph shown in Figure 5 would be the ordering $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 8 \rightarrow 10$ of its set of nodes. In general, it is desirable to achieve chordality with as few fill edges as possible. However, triangulating a graph with the minimum number of fill edges is known to

be \mathcal{NP} -complete (Yannakakis, 1981). As noted earlier, fill edges correspond to the universal relation for a QCN. As such, the *chordal constraint graph* of a given QCN is exactly its constraint graph augmented with constraints corresponding to the universal relation to make it chordal.

In light of Propositions 1 and 2, research efforts focused on making the constraint graph of a given QCN chordal and restricting path consistency to that chordal graph, while fully utilizing maximal tractable subclasses of relations and not just base relations that are typically used to describe only atomic networks. Toward this direction, we have the works of Chmeiss and Condotta (2011) for IA and Sioutis and Koubarakis (2012) for RCC-8. These works were later combined in Amaneddine *et al.* (2013) to give the following result, which is the strongest yet concerning path consistency, patchwork, and subclasses of relations:

PROPOSITION 3 (Amaneddine *et al.*, 2013) *For a given QCN $\mathcal{N}=(V, C)$ defined over a subclass of relations of a QCN that has patchwork for not trivially inconsistent and path-consistent QCNs defined over that subclass of relations, and for $G = (V, E)$ a triangulation of its constraint graph, if $\forall (i, j), (i, k), (j, k) \in E$ we have that $\emptyset \neq C_{ij} \subseteq C_{ik} \diamond C_{kj}$, then \mathcal{N} is satisfiable.*

Consequently, by Propositions 1 and 3 we have the following result:

COROLLARY 1 *For a given QCN $\mathcal{N}=(V, C)$ of RCC-8 or IA, defined over one of the maximal tractable subclasses $\mathcal{H}_8, \mathcal{C}_8$, and \mathcal{Q}_8 , or \mathcal{H}_{IA} , respectively, and for $G = (V, E)$ a triangulation of its constraint graph, if $\forall (i, j), (i, k), (j, k) \in E$ we have that $\emptyset \neq C_{ij} \subseteq C_{ik} \diamond C_{kj}$, then \mathcal{N} is satisfiable.*

Proposition 3 generalizes the results of all the works that were discussed earlier in this section and make use of path consistency as the main tool for checking the satisfiability of a given QCN, and has a great effect in the efficiency and scalability of practical reasoning. In particular, regarding native search, an algorithm based on the work of Bliet and Sam-Haroud (1999) was devised, called *partial path consistency* (Chmeiss & Condotta, 2011), that enforces *partial path consistency* on a given QCN $\mathcal{N}=(V, C)$ with respect to a triangulation $G = (V, E)$ of its constraint graph in $O(\delta|E|)$ time, where δ is the maximum vertex degree of G .

DEFINITION 11 *Given a QCN $\mathcal{N}=(V, C)$ and a graph $G = (V, E)$, \mathcal{N} is partially path consistent (w.r.t. graph G) iff for $\forall (v, v'), (v, v''), (v'', v') \in E$ we have that $C(v, v') \subseteq C(v, v'') \diamond C(v'', v')$.*

Algorithm 1: PartialPathConsistency(\mathcal{N}, G)

```

in      : A QCN  $\mathcal{N} = (V, C)$ , and a graph  $G = (V, E)$ .
output : False if the constraint network  $\mathcal{N}$  results in a trivial inconsistency (contains the empty relation),
        True if the (possibly) refined constraint network  $\mathcal{N}$  is partially path-consistent.
1 begin
2   if  $\exists (i, j) \in E(G(\mathcal{N}))$  with  $C_{ij} = \emptyset$  then
3     return False;
4    $Q \leftarrow \{(i, j) \mid (i, j) \in E\}$ ;
5   while  $Q \neq \emptyset$  do
6      $(i, j) \leftarrow Q.pop()$ ;
7     foreach  $k$  such that  $(i, k), (k, j) \in E$  do
8        $t \leftarrow C_{ik} \cap (C_{ij} \diamond C_{jk})$ ;
9       if  $t \neq C_{ik}$  then
10        if  $t = \emptyset$  then
11          return False;
12         $C_{ik} \leftarrow t$ ;  $C_{ki} \leftarrow t^{-1}$ ;
13         $Q \leftarrow Q \cup \{(i, k)\}$ ;
14       $t \leftarrow C_{kj} \cap (C_{ki} \diamond C_{ij})$ ;
15      if  $t \neq C_{kj}$  then
16        if  $t = \emptyset$  then
17          return False;
18         $C_{kj} \leftarrow t$ ;  $C_{jk} \leftarrow t^{-1}$ ;
19         $Q \leftarrow Q \cup \{(k, j)\}$ ;
20  return True;
```

Algorithm 2: PartialConsistency($\mathcal{N}, G, \mathcal{A}$)

```

in      : A QCN  $\mathcal{N} = (V, C)$ , a graph  $G = (V, E)$ , and a subclass  $\mathcal{A}$ .
output : Null if the constraint network  $\mathcal{N}$  results in a trivial inconsistency, a (possibly) refined constraint
          network  $\mathcal{N}$  over  $\mathcal{A}$  otherwise.
1 begin
2   if not PartialPathConsistency( $\mathcal{N}, G$ ) then
3     return Null;
4   if  $\forall (i, j) \in E$  we have that  $C_{ij} \in \mathcal{A}$  then
5     return  $\mathcal{N}$ ;
6   choose constraint  $C_{ij}$  such that  $C_{ij} \notin \mathcal{A}$ ;
7   split  $C_{ij}$  into  $r_1, \dots, r_k \in \mathcal{A}$ :  $r_1 \cup \dots \cup r_k = C_{ij}$ ;
8   foreach  $r \in \{r_l \mid 1 \leq l \leq k\}$  do
9     replace  $C_{ij}$  with  $r$  in  $\mathcal{N}$ ;
10    result  $\leftarrow$  PartialConsistency( $\mathcal{N}, G, \mathcal{A}$ );
11    if result  $\neq$  Null then
12      return result;
13  return Null;

```

The partial path-consistency algorithm of Chmeiss and Condotta is presented in Algorithm 1. As it is suggested in Proposition 3, the partial path-consistency algorithm is able to decide the satisfiability of a QCN defined over some subclass of relations, when path consistency can yield patchwork with respect to that subclass of relations, and when a triangulation of its constraint graph is used as the input graph in the algorithm.

The search space for intractable G s was also reduced to $O(\alpha^{|E|})$ from $O(\alpha^{|V|^2})$ for a backtracking algorithm (Renz & Nebel, 2001), where α is the branching factor provided by some subclass of relations (e.g. $\alpha = 1.4375$ for class \mathcal{H}_8 of RCC-8 (Renz & Nebel, 2001)). Such a backtracking algorithm is presented in Algorithm 2. Note that if a QCN \mathcal{N} along with a triangulation G of its constraint graph and a subclass \mathcal{A} are given as input to the backtracking algorithm, then the algorithm is able to decide the satisfiability of the given network \mathcal{N} provided that path consistency can yield patchwork with respect to \mathcal{A} .

Regarding approaches based on encodings of QCNs into Boolean formulas, that is, Boolean Satisfiability Problem (SAT)-based approaches, the implication of Proposition 3 led to significant memory and speed improvements both for IA- (Westphal *et al.*, 2013) and for RCC-8 (Westphal & Hué, 2014) targeted implementations. Further, regarding works that consider the MLP and the redundancy problem of a QCN, partial path consistency has been used as the core local consistency condition to build algorithms both for the MLP as described in Amaneddine *et al.* (2013) and for the redundancy problem as described in Sioutis *et al.* (2015b).

Before closing this section with some strong theoretical results that concern tree decompositions and patchwork, let us introduce the *treewidth* of a graph. The *width* of a tree decomposition $(T, \{X_1, \dots, X_n\})$ is $\max_{1 \leq i \leq n} |X_i| - 1$. The *treewidth* of a graph G is the minimum width possible among all tree decompositions of G . We also recall the following result regarding the treewidth of a graph G that is augmented with a new edge:

THEOREM 1 (Elidan & Gould, 2008) *Let $G = (V, E)$ be a graph of treewidth k . Then, the treewidth of graph $G' = (V, E \cup \{e\})$, where e is a new edge, is at most $k + 1$.*

In the context of QCNs, the treewidth of a QCN \mathcal{N} is simply the treewidth of its constraint graph $G(\mathcal{N})$.

THEOREM 2 (Bodirsky & Wöfl, 2011; Huang *et al.*, 2013) *For any k , the satisfiability problem for QCNs of treewidth at most k that are defined on a language that has patchwork for path-consistent atomic QCNs can be solved in polynomial time.*

Consequently, by Proposition 1 and Theorem 2 we have the following result:

COROLLARY 2 *For any k , the satisfiability problem for QCNs of IA and RCC-8 of treewidth at most k can be solved in polynomial time.*

A detailed algorithm for Theorem 2 that offers an alternative to the proof sketch of Bodirsky and Wöflf (2011) is provided in Huang *et al.* (2013). (The proof sketch in Bodirsky & Wöflf (2011) is particular to RCC-8, but it can be generalized to IA and other languages satisfying certain common properties, such as patchwork.)

Further, regarding the MLP we can have the following result:

THEOREM 3 *For any k , the MLP for QNC of treewidth at most k that is defined on a language that has patchwork for path-consistent atomic QCNs can be solved in polynomial time.*

Proof. Let $\mathcal{N}=(V, C)$ be a QCN of treewidth at most k that is defined on a language that has patchwork for path-consistent atomic QCNs. To check if a base relation $b \in C(u, v)$, with $(u, v) \in G(\mathcal{N})$, participates in at least one solution of \mathcal{N} , we must check if the QCN $\mathcal{N}'=(V, C')$ defined by $C'(u, v) = \{b\}$, $C'(v, u) = \{b\}^{-1}$, and $C'(y, w) = C(y, w) \forall (y, w) \in (V \times V) \setminus \{(u, v), (v, u)\}$ is satisfiable, so that a scenario $S = (V, C')$ with $C'(u, v) = \{b\}$ can be constructed out of the admitted solution. This satisfiability check can be done in polynomial time by Theorem 2. Note that if $(u, v) \notin G(\mathcal{N})$, we must augment the constraint graph $G(\mathcal{N})$ with (u, v) to take into account the constraint $C(u, v)$. As such, the satisfiability check will be performed on a QCN of treewidth at most $k+1$ due to Theorem 1. (After the check, (u, v) can again be removed from $G(\mathcal{N})$.) As we can have at most $O(|B||V|^2)$ base relations in any given QCN, it follows that we can solve the MLP in polynomial time. \square

Consequently, by Proposition 1 and Theorem 3 we have the following result:

COROLLARY 3 *For any k , the MLP for QCNs of IA and RCC-8 of treewidth at most k can be solved in polynomial time.*

Regarding the redundancy problem, we can have the following result:

THEOREM 4 *For any k , the redundancy problem for QCNs of treewidth at most k that are defined on a language that has patchwork for path-consistent atomic QCNs can be solved in polynomial time.*

Proof. Let $\mathcal{N}=(V, C)$ be a QCN of treewidth at most k that is defined on a language that has patchwork for path-consistent atomic QCNs. To check if a constraint $C(u, v)$, with $(u, v) \in G(\mathcal{N})$, is non-redundant in \mathcal{N} , we must check if there exists a base relation $b \notin C(u, v)$ that participates in a solution of the modified \mathcal{N} that results by removing $C(u, v)$. This is equivalent to checking if the QCN $\mathcal{N}'=(V, C')$ defined by $C'(u, v) = B \setminus C(u, v)$, $C'(v, u) = B \setminus (C(u, v))^{-1}$, and $C'(y, w) = C(y, w) \forall (y, w) \in (V \times V) \setminus \{(u, v), (v, u)\}$ is satisfiable. This satisfiability check can be done in polynomial time by Theorem 2. (Note that if $(u, v) \notin G(\mathcal{N})$, $C(u, v)$ is by definition redundant.) Since we can have at most $O(|V|^2)$ constraints in any given QCN, it follows that we can solve the redundancy problem in polynomial time. \square

Consequently, by Proposition 1 and Theorem 4 we have the following result:

COROLLARY 4 *For any k , the redundancy problem for QCNs of IA and RCC-8 of treewidth at most k can be solved in polynomial time.*

4 Partitioning graph

In this section, we prove that the decomposition-based approach presented in Nikolaou and Koubarakis (2014) for checking the satisfiability of QCNs of RCC-8 lacks soundness, as the notion of a *partitioning graph* defined in that work is not coherent with the use of patchwork upon which it solely relies, in two ways, which we enumerate and analyze in the form of issues.

Let $G = (V, E)$ be a graph and k a positive integer. If $U \subseteq V$, then $G(U)$ will denote the subgraph of G that is induced by the set of vertices U . A set $\{V_i \subseteq V \mid 1 \leq i \leq k\}$ with k pairwise-disjoint elements such that $\bigcup_{i=1}^k V_i = V$, is called a k -way partitioning of G . Finally, let \emptyset denote the empty, edgeless, graph. We recall the following definition of a partitioning graph from Nikolaou and Koubarakis (2014):

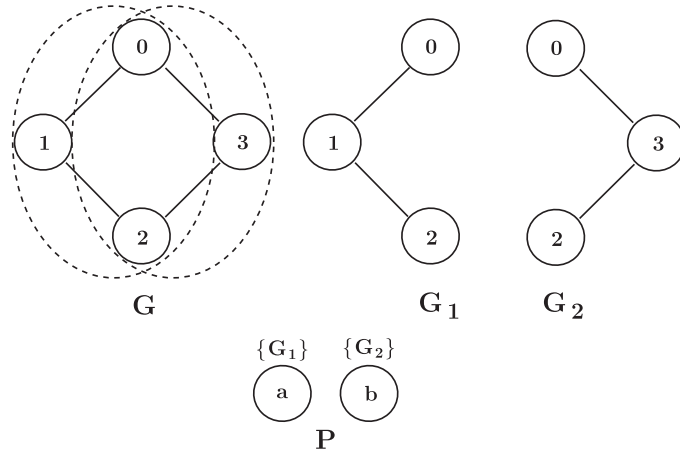


Figure 6 A graph and its partitioning graph with the parts comprising it (also contained in dashed circles in the initial graph)

DEFINITION 12 Let $G = (V, E)$ be a graph and $\{V_1, \dots, V_k\}$ a k -way partitioning of G for some positive integer k . A partitioning graph P of G is a graph $(V_P, E_P, \lambda_P, G_P)$, where $V_P = \{v_1, \dots, v_k\}$ is the set of its nodes, E_P the set of its edges, $\lambda_P: V_P \rightarrow 2^V$ a function that maps each node of P to a subset of the set of vertices V of G , and G_P a set of k subgraphs (parts) of G . The following conditions must be satisfied:

- If $G_i \in G_P$ then the set of vertices of G_i is a superset U of $\lambda_P(v_i)$ and the set of its edges is $E(G(U))$.
- Any edge in G should be present in at least one subgraph $G_i \in G_P$.
- An edge (v_i, v_j) belongs to E_P if and only if $G_i \cap G_j \neq \emptyset$ (i.e. if and only if the subgraphs G_i and G_j corresponding to nodes v_i and v_j , respectively, share a common edge).

Let G be a graph and $P = (V_P, E_P, \lambda_P, G_P)$ one of its partitioning graphs. Then, an edge e of G present in more than one subgraph $G_i \in G_P$ is called a *global edge*. An edge e of G present in exactly one subgraph $G_i \in G_P$ is called a *local edge*.

We will now enumerate the issues that lead to non-soundness and provide counter-examples for each case. The reader is kindly asked to refer to Nikolaou and Koubarakis (2014) and check that the flaws pointed out here are actually present in Nikolaou and Koubarakis (2014).

The issues that we will enumerate will allow us to infer the following fact:

PROPOSITION 4 *The approach presented in Nikolaou and Koubarakis (2014) for checking the satisfiability of a QCN of RCC-8 lacks soundness. In particular, Propositions 2 and 3 in Nikolaou and Koubarakis (2014) do not hold.*

We begin with the first issue.

Issue 1. The first issue has to do with the fact that a complete agreement on the constraints between the common variables of two networks is not achieved in order to allow the applicability of patchwork. Let us consider the example of Figure 6. Graph G is partitioned into two parts, namely, G_1 and G_2 . The partitioning graph is shown in the lower part of the figure, and it comprises the set of nodes $\{a, b\}$ and an empty set of edges. Node a corresponds to subgraph G_1 and node b to subgraph G_2 . Its set of edges E_P is empty as subgraphs G_1 and G_2 do not share a common edge (that would otherwise be the global edge $(0, 2)$), thus, the only possible edge (a, b) does not exist. In Nikolaou and Koubarakis (2014), the authors perform path consistency on the subgraphs of a graph separately, in a parallel fashion, and then rely on the set of edges E_P to identify the subgraphs among which a complete agreement has to be ensured (the reader is kindly asked to refer to line 7 in the function of Algorithm 2 in Nikolaou & Koubarakis (2014)). If, as in this example, such an edge does not exist, a complete agreement is never achieved. This can be the cause of failing to identify inconsistencies. Let us assume that graph G , as depicted in Figure 8, is the constraint graph of a given QCN comprising constraints $C_{01} = C_{12} = C_{23} = C_{30} = \{TPP\}$. This yields an

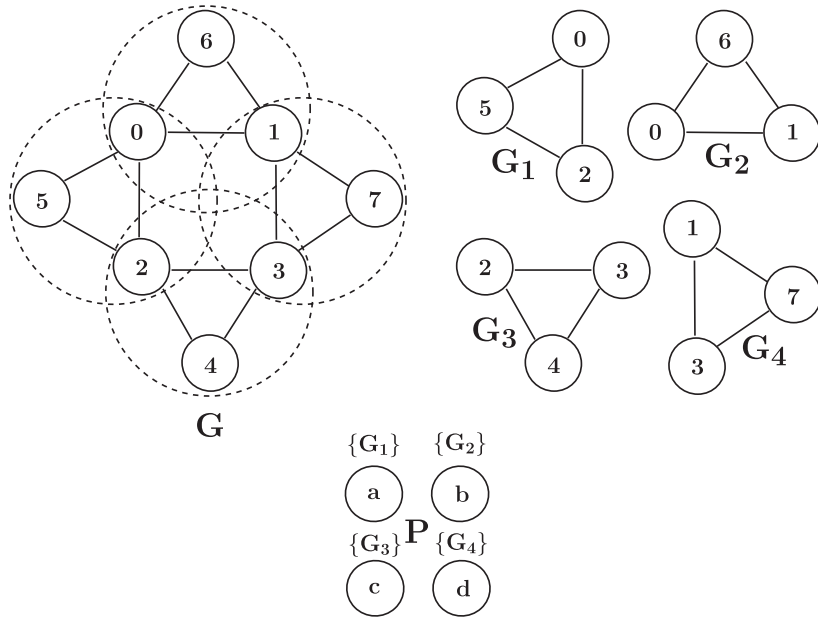


Figure 7 A graph and its partitioning graph with the parts comprising it (also contained in dashed circles in the initial graph)

unsatisfiable network, as it basically infers that region 0 is properly contained in region 2, and vice versa. Applying path consistency on that network would result in the empty relation assignment for constraint C_{02} (inconsistency). However, that constraint is never checked in our example. Although the authors implicitly complete subgraphs G_1 and G_2 in order to apply path consistency, they do not complete these subgraphs when computing their intersection, as clearly specified in the last bullet of Definition 12. Even if they did implicitly consider complete subgraphs for that part of the definition, and the edge (a, b) indeed existed, line 7 in the function of Algorithm 2 in Nikolaou and Koubarakis (2014) still requires that an agreement should only be achieved for every common edge of G_1 and G_2 (the initial non-complete subgraphs), which is none. If they implicitly considered complete subgraphs for that part of the algorithm too, then this particular issue for a two-way partitioning would be resolved. We have also verified this issue experimentally with the implementation used in Nikolaou and Koubarakis (2014)⁷.

Before proceeding to the next issue, let us assume that the first issue is fixed with everything that we propose, and a two-way partitioning is actually valid for applying patchwork. We mean to show that the concept of a partitioning graph is beyond repair, unless it is structured in a way that it defines a tree decomposition, which defeats the purpose of having to define a partitioning graph in the first place.

The second issue follows.

Issue 2. This issue has to do with the fact that even if the first issue is resolved, the partitioning graph can suffer from the existence of cycles that are created by subgraphs of a given graph. Let us consider the example of Figure 7. Graph G is partitioned into four parts, namely, G_1 , G_2 , G_3 , and G_4 . The partitioning graph is shown in the lower part of the figure, and the correspondence between its sets of nodes and edges with the different subgraphs should be clear up to this point. Note that all subgraphs are complete, thus, they completely overlap with each other on the common vertices. For example, graph G_1 completely overlaps with graph G_2 on the edges of the graph defined on the single common vertex 0, as their intersection yields the complete graph on single vertex 0. Although such an overlap is trivial, as a complete graph on a single vertex (singleton graph) does not have any edges, it is sufficient to ensure the applicability of patchwork for the corresponding constraint networks. (Our example

⁷ <https://www.dropbox.com/sh/h61edhshw5p8ne2/AAAu00WyYB5r8cLmoRBLV8x1a?dl=0>

can be easily extended to non-trivial overlaps.) However, due to the last bullet of Definition 12, the partitioning graph is unable to obtain any edges, as there can exist no global edges. In fact, even if some edges existed in E_P , in any possible combination and amount, the partitioning graph would still fail to detect the cycle that is constructed by the complete subgraphs G_1 , G_2 , G_3 , and G_4 , namely, the cycle defined by vertices 0, 1, 2, and 3. This cycle, as shown in the example of Figure 6, can harbor an inconsistency. Such a cycle exists also in Nikolaou and Koubarakis (2014, figure 1) between vertices 3, 4, 5, and 7 there. Patchwork alone is only valid for tree decompositions, as tree decompositions guarantee acyclicity of cliques and, thus, do not harbor cycles with potential inconsistencies that cannot be detected by the application of path consistency on the different cliques. This issue was again verified experimentally.

Essentially, the approach defines a partial algorithm; a given satisfiable QCN will be shown to be satisfiable, as the approach in Nikolaou and Koubarakis (2014) because disregarding constraints operates on a less restrictive constraint graph of the input network where constraint propagation and consistency checks are limited, whilst an unsatisfiable QCN may be shown to be satisfiable.

4.1 Impact on performance

The main contribution of Nikolaou and Koubarakis (2014) lies in the performance of its offered implementation, as it promises efficiency that goes well beyond the state-of-the-art. Computing a good k -way partitioning⁸ alone is among the graph partitioning problems that fall under the category of \mathcal{NP} -hard problems (Garey *et al.*, 1976), and solutions to these problems are generally derived using heuristics and approximation algorithms, such as the ones offered by the METIS⁹ software employed in Nikolaou and Koubarakis (2014). We leave aside any extra computational complexity that would result from needing to restrict a partitioning graph to being a tree decomposition (e.g. by identifying cycles or using some recursion as in Huang *et al.* (2013)) and focus on native search. As explained in Section 3, native search in QSTR is bound to the number of constraints of a given QCN, and not to its number of variables as in ‘traditional’ constraint programming. This is because, in a sense, the constraints of a given QCN are the *true* variables for which we have to assign some relation. Indeed, the search space defined in Nikolaou and Koubarakis (2014) relies mainly on the number of constraints of a given QCN. In particular, we can recall the following proposition from Nikolaou and Koubarakis (2014):

PROPOSITION 5 (Nikolaou & Koubarakis, 2014) *Let $G = (V, E)$ be the constraint graph of a QCN of RCC-8 and P a partitioning graph of G with k parts. The search space of algorithm *DConsistency* (Nikolaou & Koubarakis, 2014) is $O(|B|^g(|B|gkm^3 + \alpha^l m^3))$, where g is the number of global edges, l and m the maximum number of local edges and vertices, respectively, among all parts of P , and α the branching factor of the subclass of relations employed. If Π denotes the aforementioned search space, then given p processing units and assuming a balanced partitioning among the k parts (i.e. $m = |V|/k$), the elapsed running time of algorithm *DConsistency* is $O\left(\frac{\Pi}{p}\right)$.*

We showed earlier that some global edges can be disregarded, thus, parameter g as defined in Proposition 5 leads to a significantly reduced search space for the implementation of Nikolaou and Koubarakis (2014) with respect to the one that should normally be considered, as g has an exponential contribution. However, even in that case, a re-evaluation of the implementation used in Nikolaou and Koubarakis (2014) against state-of-the-art solvers, showed that it performs very poorly with respect to the state-of-the-art (Sioutis, 2014). The work in Sioutis (2014) does not deal with any of the issues that we dealt with in this paper as it assumes a partitioning graph to implicitly define a tree decomposition, thus, Sioutis (2014) presents mostly lower bounds on the performance of the implementation used in Nikolaou and Koubarakis (2014).

⁸ *Good* in terms of obtaining smaller components that meet specific properties, for example, a good partitioning can be defined as a partitioning in which the number of edges running between separated components is relatively small.

⁹ <http://glaros.dtc.umn.edu/gkhome/views/metis>

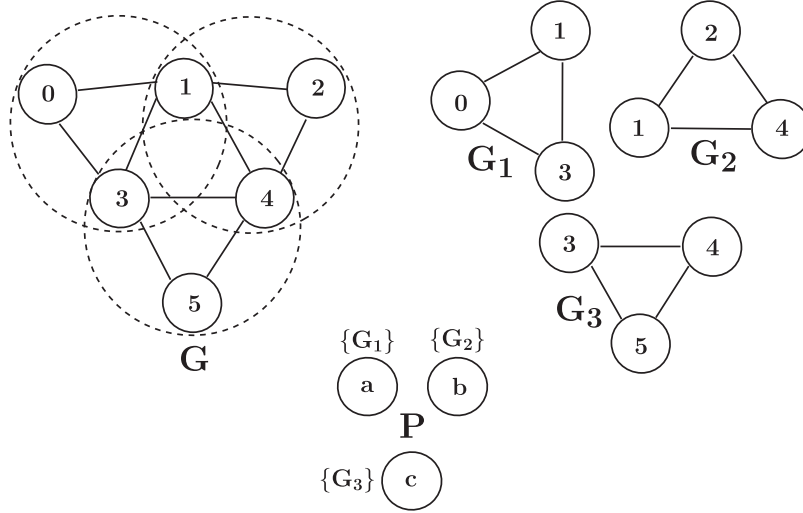


Figure 8 A chordal graph and its partitioning graph with the parts comprising it (also contained in dashed circles in the initial graph)

4.2 Fixing the issues

We noted earlier in this section that the concept of a partitioning graph is beyond repair, unless it is structured in a way that it defines a tree decomposition. It may seem tempting as a quick hack to triangulate the constraint graph of a given QCN of RCC-8 and, thus, obtain a chordal constraint graph of that QCN, and feed it directly to the partitioning algorithm described in Nikolaou and Koubarakis (2014). However, this may still yield non-soundness. We explain as follows.

Consider the example shown in Figure 8 where the chordal graph G is partitioned into three subgraphs. The partitioning graph P is such that it is unable to capture/break the cycle defined by vertices 1, 3, and 4. This cycle may harbor an inconsistency, which will not be detected by the application of path consistency on the different parts of the partitioning graph. One way to force a partitioning graph into defining a tree decomposition is using METIS in a recursive manner, as it is done in Huang *et al.* (2013). In particular, one has to initially partition a given graph G into two parts, and then recursively apply the same procedure on the obtained parts, until no further partitioning can occur. However, this can be a costly operation. A faster way is to rely on chordal graphs (tree decompositions into cliques), which can both be constructed and also yield a natural tree decomposition of their cliques in linear time (Diestel, 2012). The graphs induced by the cliques can then be collected at no extra cost and serve as the parts of the partitioning graph; consequently, the approach described in Nikolaou and Koubarakis (2014) can then be carried out with soundness and completeness.

5 Toward efficient utilization of parallelism

As noted in Nikolaou and Koubarakis (2014), the authors provided a parallel implementation for checking the satisfiability of a QCN of RCC-8, which however lacks soundness (Proposition 4). In this section, we present a simple decomposition scheme that exploits the sparse and loosely connected structure of the constraint graphs of very large real-world QCNs, which have been of notable interest in the recent literature (Nikolaou & Koubarakis, 2014; Sioutis, 2014; Sioutis & Condotta, 2014), and paves the way for efficient utilization of parallelism. Our approach is based on extracting the smaller QCNs that correspond to the biconnected components of the constraint graph of a given large QCN and reasoning with these smaller biconnected QCNs completely separately, in a parallel or serial fashion, which, as our experimentation suggests, significantly decongests search when solving intractable QCNs.

First, we recall a definition from Dechter (2003) regarding biconnected graphs and components.

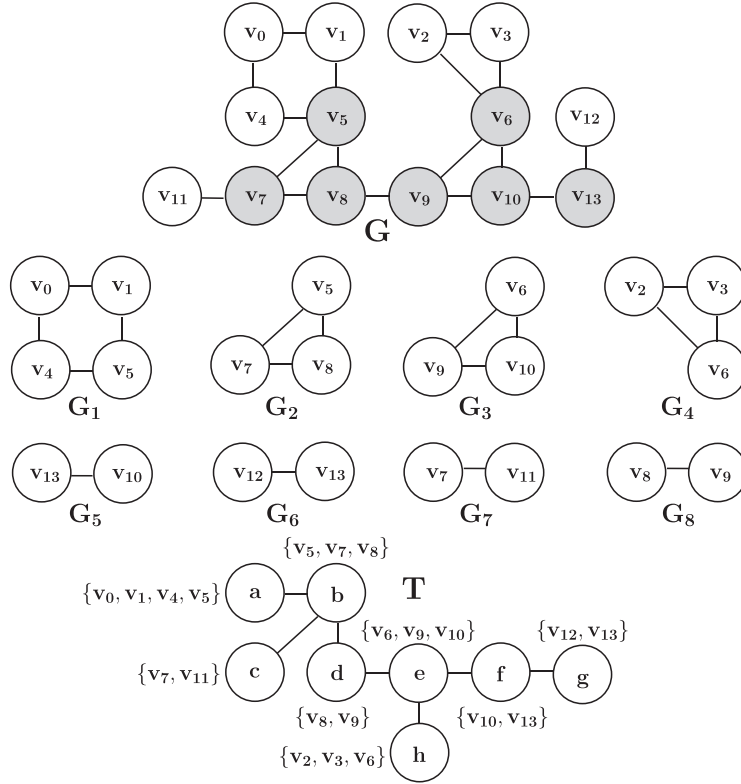


Figure 9 A graph G (top) with its biconnected components (middle) and its tree decomposition (bottom)

DEFINITION 13 A connected graph G is said to have an articulation vertex u iff there exist vertices v and v' such that all paths connecting v and v' pass through u . A graph that has an articulation vertex is called separable, and one that has none is called biconnected. A maximal subgraph with no articulation vertices is called a biconnected component.

Intuitively, an articulation vertex is any vertex whose removal increases the number of connected components in a given graph. From Dechter (2003), we also have the following property:

PROPERTY 2 (Dechter, 2003) Let G be a graph and $\{G_1, \dots, G_n\}$ its biconnected components. Then, there exists a tree decomposition $(T, \{X_1, \dots, X_n\})$ of G , where cluster $X_i \subseteq V(G)$ induces the biconnected component G_i of G , for every $i \in \{1, \dots, n\}$.

Let us now view the discussed notions in an example. Figure 9 depicts a graph G , along with its biconnected components, and its tree decomposition. Vertices in gray color are the articulation vertices of G . The tree decomposition comprises a tree $T = (I, F)$ and a cluster X_i for every node $i \in I$ of that tree, for example, $X_a = \{v_0, v_1, v_4, v_5\}$. We remind the reader that $\mathcal{N} \downarrow_V$ is the QCN \mathcal{N} restricted to a set of variables V . We can obtain the following proposition:

PROPOSITION 6 Let \mathcal{N} be a QCN defined on a language that has patchwork for satisfiable atomic QCNs, and let $\{G_1, \dots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, \mathcal{N} is satisfiable iff \mathcal{N}_i is satisfiable for every $i \in \{1, \dots, k\}$, where \mathcal{N}_i is $\mathcal{N} \downarrow_{V(G_i)}$.

Proof. By Property 2, the constraint graph $G(\mathcal{N})$ has a tree decomposition $(T, \{X_1, \dots, X_k\})$, where cluster X_i induces G_i , for every $i \in \{1, \dots, k\}$. We can also infer by Definition 13 that $\forall i, j \in \{1, \dots, k\}$ with $i \neq j$, $V(G_i) \cap V(G_j)$ contains at most one vertex u . If \mathcal{N}_i is satisfiable for every $i, j \in \{1, \dots, k\}$, we can obtain a satisfiable atomic sub-QCN of \mathcal{N}_i , that is, a scenario S_i of \mathcal{N}_i , for every $i \in \{1, \dots, k\}$. For any possible scenarios and any $i, j \in \{1, \dots, k\}$ with $i \neq j$, we will have that $S_i = (V(G_i), C_i)$ and $S_j = (V(G_j), C_j)$ will always agree on the single unary constraint that is defined by a single vertex $u \in V$

Algorithm 3: Decomposer(\mathcal{N})

```

in      : A QCN  $\mathcal{N} = (V, C)$ .
output : A collection of QCNs.
1 begin
2   if  $\exists (v, v') \in E(G(\mathcal{N}))$  with  $C(v, v') = \emptyset$  then
3     return  $\{(V, C)\}$ ;
4    $S \leftarrow \{g \mid g \in \text{BCSubgraphs}(G(\mathcal{N})); \text{ and } |V(g)| > 2\}$ ;
5    $\chi \leftarrow \emptyset$ ;
6   while  $S \neq \emptyset$  do
7      $g \leftarrow S.\text{pop}()$ ;
8      $V_g \leftarrow V(g)$ ;  $E_g \leftarrow E(g)$ ;
9      $C_g \leftarrow \text{map}(\{(v, v') : (\text{B if } v \neq v' \text{ else } \{\text{Id}\}) \mid v, v' \in V_g\})$ ;
10    foreach  $(v, v') \in E_g$  do
11       $C_g(v, v') \leftarrow C(v, v')$ ;  $C_g(v', v) \leftarrow C(v', v)$ ;
12     $\chi \leftarrow \chi \cup \{(V_g, C_g)\}$ ;
13  return  $\chi$ ;

```

Algorithm 4: Solver+(\mathcal{N})

```

in      : A QCN  $\mathcal{N} = (V, C)$ .
output : True, or False.
1 begin
2   foreach  $n \in \text{Decomposer}(\mathcal{N})$  do
3     if not  $\| \text{Solver}(n) \|$  then
4       return False;
5   return True;

```

$(G_i) \cap V(G_j)$ whenever we have that $V(G_i) \cap V(G_j) \neq \emptyset$, that is, $C_i(u, u) = C_j(u, u) = \{\text{Id}\}$, as by Definition 1 we have that for any QCN $\mathcal{M} = (V, C)$, $C(v, v) = \{\text{Id}\} \forall v \in V$. By Definition 7, we can apply patchwork to patch together all the satisfiable atomic QCNs \mathcal{S}_i with $i \in \{1, \dots, k\}$ in a tree-like manner and, thus, derive the satisfiability of \mathcal{N} . If \mathcal{N} is satisfiable, then, clearly, \mathcal{N}_i will be satisfiable for every $i \in \{1, \dots, k\}$. \square

Consequently, by Proposition 6 and the fact that IA and RCC-8 have patchwork for satisfiable atomic QCNs of their relations (Lutz & Milicic, 2007), we have the following result:

COROLLARY 5 *Let \mathcal{N} be a QCN of IA or RCC-8, and let $\{G_1, \dots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, \mathcal{N} is satisfiable iff \mathcal{N}_i is satisfiable for every $i \in \{1, \dots, k\}$, where \mathcal{N}_i is $\mathcal{N} \downarrow_{V(G_i)}$.*

It is important to note that the proof of Proposition 6 is based on tree decompositions whose nodes correspond to clusters where any two clusters share at most one vertex with each other. In case two clusters share more than one vertex with each other, the involved QCNs should be, for instance (and among other conditions), not trivially inconsistent, path consistent, and defined over a subclass of relations of a qualitative constraint language that has patchwork for not trivially inconsistent and path-consistent QCN defined over that subclass of relations, as it is specified in Proposition 3 and considered in Sioutis and Koubarakis (2012) and Chmeiss and Condotta (2011) for RCC-8 and IA, respectively. A simple algorithm for obtaining a collection of QCNs that correspond to the biconnected components of the constraint graph of a given QCN is presented in Algorithm 3. Note that in lines 2–3 we immediately return the input QCN if it is trivially inconsistent, as it would not make any sense to continue with the decomposition procedure. Function $\text{BCSubgraphs}(G)$ in line 4 returns the biconnected components of a graph $G = (V, E)$ and has a runtime of $O(|E|)$ (Dechter, 2003). Note that in line 4 we keep only the components of order > 2 , as any not trivially inconsistent QCN of < 3 variables is trivially satisfiable (by definition of a base relation). In what follows, we always consider components of order > 2 . Based on Decomposer, we can obtain an algorithm to increase the performance of any given state-of-the-art solver that is sound and complete for checking the satisfiability of a given QCN \mathcal{N} defined on a language that has patchwork for satisfiable atomic QCNs; that algorithm is presented in Algorithm 4. Let us denote any such given state-of-the-art solver by Solver. Then, Algorithm 4 will use Solver to decide the satisfiability of the QCNs that correspond to the

biconnected components of the constraint graph of \mathcal{N} . The enclosure with symbol \parallel for Solver denotes the fact that Solver can be used in a parallel or serial fashion.

Regarding the MLP, we can have the following result:

PROPOSITION 7 *Let $\mathcal{N}=(V, C)$ be a satisfiable QCN defined on a language that has patchwork for satisfiable atomic QCNs, and let $\{G_1, \dots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, a base relation $b \in C(u, v)$, with $u, v \in V(G_i)$, is feasible iff there exists a scenario $S_i=(V_i, C'_i)$ of $\mathcal{N}_i=(V_i, C_i)$ such that $C'_i(u, v)=\{b\}$, where \mathcal{N}_i is $\mathcal{N} \downarrow_{V(G_i)}$, for some $i \in \{1, \dots, k\}$.*

Proof. Let $\mathcal{N}'=(V, C')$ be the QCN defined by $C'(u, v) = \{b\}$, $C'(v, u) = \{b\}^{-1}$, and $C'(y, w) = C(y, w) \forall (y, w) \in (V \times V) \setminus \{(u, v), (v, u)\}$. Further, let $\mathcal{N}'_i=(V_i, C'_i)$ be the restriction of \mathcal{N}' to $V(G_i)$, viz., $\mathcal{N}' \downarrow_{V(G_i)}$. Then, by Proposition 6 and as G_i is a biconnected component of $G(\mathcal{N})$, we know that \mathcal{N}' is satisfiable iff \mathcal{N}'_i is satisfiable; in addition, any scenario $S_i=(V_i, C'_i)$ of \mathcal{N}'_i is the restriction of some scenario $S=(V, C'')$ of \mathcal{N}' to V_i , and any scenario $S=(V, C'')$ of \mathcal{N}' is the extension of some scenario $S_i=(V_i, C'_i)$ of \mathcal{N}'_i to V . As such, the feasibility of b can be characterized by considering \mathcal{N}'_i instead of \mathcal{N}' . \square

Given a satisfiable QCN $\mathcal{N}=(V, C)$, Proposition 7 allows one to quickly characterize the feasibility of a base relation $b \in C(u, v)$, with $u, v \in V(G')$, where G' is a biconnected component of the constraint graph $G(\mathcal{N})$. If $u, v \notin V(G')$ for any biconnected component G' of $G(\mathcal{N})$, then b belongs to a constraint that is labeled with the universal relation B and its feasibility can still be efficiently characterized under certain conditions by a function similar to `extractFeasible` as described in Amaneddine *et al.* (2013).

Consequently, by Proposition 7 and the fact that IA and RCC-8 have patchwork for satisfiable atomic QCNs of their relations (Lutz & Milicic, 2007), we have the following result:

COROLLARY 6 *Let $\mathcal{N}=(V, C)$ be a satisfiable QCN of IA or RCC-8, and let $\{G_1, \dots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, a base relation $b \in C(u, v)$, with $u, v \in V(G_i)$, is feasible iff there exists a scenario $S_i=(V_i, C'_i)$ of $\mathcal{N}_i=(V_i, C_i)$ such that $C'_i(u, v)=\{b\}$, where \mathcal{N}_i is $\mathcal{N} \downarrow_{V(G_i)}$, for some $i \in \{1, \dots, k\}$.*

Regarding the redundancy problem, we can have the following result:

PROPOSITION 8 *Let $\mathcal{N}=(V, C)$ be a satisfiable QCN defined on a language that has patchwork for satisfiable atomic QCNs, and let $\{G_1, \dots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, a relation $C(u, v)$, with $u, v \in V$, is non-redundant in \mathcal{N} iff $(u, v) \in E(G_i)$ and $C(u, v)$ is non-redundant in $\mathcal{N}_i=(V_i, C_i)$, where \mathcal{N}_i is $\mathcal{N} \downarrow_{V(G_i)}$, for some $i \in \{1, \dots, k\}$.*

Proof. Clearly, a relation $C(u, v)$ is redundant in \mathcal{N} if $(u, v) \notin E(G_i)$ for any $i \in \{1, \dots, k\}$, as it will correspond to the universal relation B. Let us consider a relation $C(u, v)$ where $(u, v) \in E(G_i)$ for some $i \in \{1, \dots, k\}$. Let $\mathcal{N}'=(V, C')$ be the QCN defined by $C'(u, v) = B \setminus C(u, v)$, $C'(v, u) = B \setminus (C(u, v))^{-1}$, and $C'(y, w) = C(y, w) \forall (y, w) \in (V \times V) \setminus \{(u, v), (v, u)\}$. Further, let $\mathcal{N}'_i=(V_i, C'_i)$ be the restriction of \mathcal{N}' to $V(G_i)$, viz., $\mathcal{N}' \downarrow_{V(G_i)}$. Then, by Proposition 6 and as G_i is a biconnected component of $G(\mathcal{N})$, we know that \mathcal{N}' is satisfiable iff \mathcal{N}'_i is satisfiable; in addition, any scenario $S_i=(V_i, C'_i)$ of \mathcal{N}'_i is the restriction of some scenario $S=(V, C'')$ of \mathcal{N}' to V_i , and any scenario $S=(V, C'')$ of \mathcal{N}' is the extension of some scenario $S_i=(V_i, C'_i)$ of \mathcal{N}'_i to V . Finally, since for any scenario there exists a solution that satisfies all of its base relations, the redundancy of $C(u, v)$ can be characterized by considering \mathcal{N}'_i instead of \mathcal{N}' . \square

Consequently, by Proposition 8 and the fact that IA and RCC-8 have patchwork for satisfiable atomic QCNs of their relations (Lutz & Milicic, 2007), we have the following result:

COROLLARY 7 *Let $\mathcal{N}=(V, C)$ be a satisfiable QCN of IA or RCC-8, and let $\{G_1, \dots, G_k\}$ be the biconnected components of its constraint graph $G(\mathcal{N})$. Then, a relation $C(u, v)$, with $u, v \in V$, is non-redundant in \mathcal{N} iff $(u, v) \in E(G_i)$ and $C(u, v)$ is non-redundant in $\mathcal{N}_i=(V_i, C_i)$, where \mathcal{N}_i is $\mathcal{N} \downarrow_{V(G_i)}$, for some $i \in \{1, \dots, k\}$.*

5.1 Data set

We review the data set of real RCC-8 network instances that was originally introduced in Nikolaou and Koubarakis (2014), and which we describe here as follows:

- `nuts`: a nomenclature of territorial units¹⁰.
- `adm1`: a network that describes the administrative geography of Great Britain (Goodwin et al., 2008).
- `gadm1`: a network that describes the German administrative units (see footnote 10).
- `gadm2`: a network that describes the world’s (global) administrative areas¹¹.
- `adm2`: a network that describes the Greek administrative geography (see footnote 10).

The aforementioned network instances are tractable and contain at most two base RCC-8 relations per edge. The characteristics of the constraint graphs of these networks are presented in Table 1.

As it can be seen, the constraint graphs of the networks vary in order, but they are all relatively sparse. This comes as no surprise, as real-world graphs often present a scale-free structure (Barabasi & Bonabeau, 2003), which results in them being sparse (Del Genio *et al.*, 2011). Thus, we expect these constraint graphs to be loosely connected and yield a high number of biconnected components. We can view information regarding the biconnected components of the constraint graphs of our networks in Table 2 (whereby *max order*, *median order*, and *min order* we refer to the maximum, median, and minimum number of vertices, respectively, met among the biconnected components).

The findings are quite impressive, in the sense that the maximum order among the biconnected components of a constraint graph is significantly smaller than the order of that graph. For example, the constraint graph of the biggest real RCC-8 network, namely, `adm2`, has an order of value 1 733 000, but the maximum order among its biconnected components is only of value 22 808. Note also that, as the median metric suggests, most of the biconnected components of a graph have an order much closer to the minimum order than the maximum order among the biconnected components of that graph.

Instances for evaluating the satisfiability checking performance of the reasoners for intractable QCNs, which are of our interest in this paper, were constructed in Nikolaou and Koubarakis (2014) with the introduction of \mathcal{NP}_8 relations (Renz & Nebel, 2001) in the networks’ edges. These instances will be denoted by `hard-nuts`, `hard-adm1`, and `hard-gadm1` in the evaluation to follow, and are structurally identical to networks `nuts`, `adm1`, and `gadm1`, respectively, that is, their constraint graphs have the same characteristics as those presented in Tables 1 and 2.

As Nikolaou and Koubarakis (2014) suggests, some state-of-the-art reasoners, such as GQR (Gantner *et al.*, 2008), use a matrix to represent a QCN $\mathcal{N}=(V, C)$, which has a $O(|V|^2)$ memory requirement. It would be impossible to store a graph of the order of `adm2` in a matrix as we would need ~ 3 TB of memory. Even if memory was not the issue, the time complexity alone of a path-consistency algorithm would explode, while the backtracking algorithm that is typically used for tackling intractable QCNs and makes use of path consistency as a forward checking step, could suffer from an increased search space. Heuristics for the backtracking algorithm could also have a hard time distinguishing between biconnected components. Consider, for example, a situation where the backtracking algorithm backtracks from an instantiation of a constraint in a biconnected component to an instantiation of a constraint in a different biconnected component. Since the constraints belong to different biconnected components, we have already shown that they are completely unrelated to each other (i.e. satisfying one constraint does not affect the other in any way); nevertheless, they might still define a huge branch in the search tree that is spawned by the backtracking algorithm. Such a situation is depicted in Figure 10, which presents two QCNs $\mathcal{N}_i=(V_i, C_i)$ and $\mathcal{N}_j=(V_j, C_j)$ such that $V_i \cap V_j = \{v\}$. Let us assume that their constraint graphs are biconnected. Then, the constraint graph of $\mathcal{N}_i \cup \mathcal{N}_j$ has $G(\mathcal{N}_i)$ and $G(\mathcal{N}_j)$ as its biconnected components. It is clear that the valuation of constraint $C_i(u_i, v)$ with any of the values r_1 or r_2 does not affect the satisfiability or unsatisfiability of the valuation of constraint $C_j(v, u_j)$ with any of the values l_1, l_2 , or l_3 , and vice versa. However, if we choose not to treat the biconnected components separately, a huge branch

¹⁰ Retrieved from <http://www.linkedopendata.gr/>

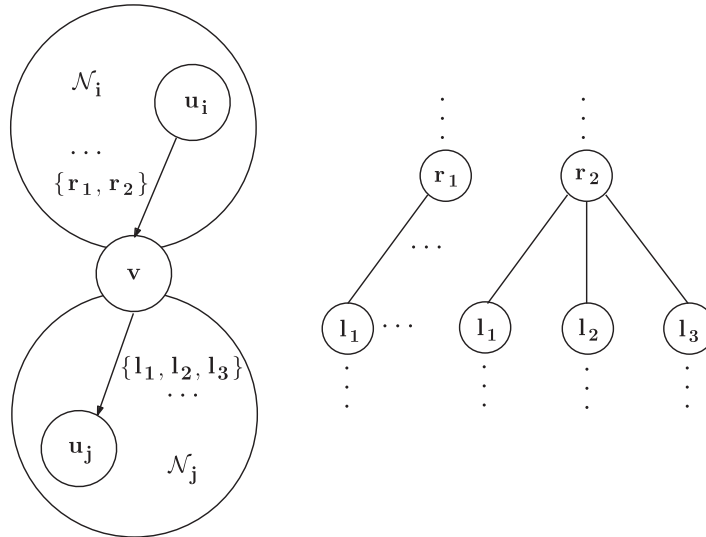
¹¹ <http://gadm.geovocab.org/>

Table 1 Characteristics of real Region Connection Calculus (RCC)-8 networks

Network	Number of nodes	Number of edges	Average degree
nuts	2236	3176	2.84
adm1	11 762	44 833	7.62
gadm1	42 750	159 600	7.47
gadm2	276 728	590 443	4.27
adm2	1 733 000	5 236 270	6.04

Table 2 Biconnected components of real Region Connection Calculus (RCC)-8 networks

Network	Number of components	Maximum order	Median order	Minimum order
nuts	64	52	8	3
adm1	5	11 666	30	3
gadm1	166	19 864	6	3
gadm2	2285	2371	18	3
adm2	2889	22 808	579	4

**Figure 10** A separable constraint graph with an articulation vertex v

might be defined, as viewed in Figure 10, that could otherwise be entirely avoided. Proposition 6 allows us to treat the QCNs that correspond to biconnected components completely separately, in a parallel or serial fashion, and avoid the aforementioned bothersome issues.

5.2 Evaluation

We consider the *hard* network instances *hard-nuts*, *hard-adm1*, and *hard-gadm1* from Nikolaou and Koubarakis (2014) that comprise \mathcal{NP}_8 relations (Renz & Nebel, 2001) to utilize the whole reasoning engine of a reasoner. If Solver is the name of a reasoner, Solver+ denotes the use of Algorithm 4 with that reasoner. The experiments were carried out on a computer with an Intel Core 2 Quad Q9400 processor with a CPU frequency of 2.66 GHz per core, 8 GB RAM, and the Precise Pangolin x86_64 OS. GQR (under version 1500) was compiled with gcc/g++ 4.6.3 and Sarissa, Phalanx, and

Table 3 Performance comparison based on elapsed time

Solver	GQR (seconds)	GQR+ (seconds)	Pha. (seconds)	Pha.+ (seconds)	Sar. (seconds)	Sar.+ (seconds)	Pha.∇ (seconds)	Pha.∇+ (seconds)
hard-nuts	2.0	0.1	4.0	0.6	0.8	0.6	0.9	0.6
hard-adm1	4.7E3	5.2E3	3.4E3	3.7E3	161.5	137.7	98.3	97.4
hard-gadm1	1.4E4	1.2	1.0E5	3.5	2.0E3	3.4	1.1E3	3.0

Pha. = Phalanx; Sar. = Sarissa.

Phalanx∇ (Sioutis & Condotta, 2014) (all under version 0.2) were run with PyPy 2.4.0¹², which fully implements Python 2.7.8. For all reasoners, the best performing heuristics were enabled. (Obviously, we did not consider the implementation of Nikolaou & Koubarakis (2014) in our evaluation as it is not sound.) We chose to reason in a serial fashion, from smaller to bigger QCN, so as to stress how much more path consistency and the backtracking algorithm that utilizes it along with the heuristics in each reasoner benefit from reasoning with the smaller biconnected QCNs than reasoning with the initial large and loosely connected QCN, when both approaches are offered the same computational power. Thus, only one CPU core was used in our experiments.

The results are shown in Table 3 and make clear that our simple decomposition scheme aids the performance of each reasoner substantially, with the more apparent case being that of `hard-gadm1`, which is unsatisfiable. Networks `hard-nuts` and `hard-adm1` are satisfiable. In particular, GQR decides `gadm1` in ~4 hours, while GQR+ in 1.2 seconds, and similar results are obtained for the other reasoners too. When an inconsistency is detected in a QCN n that corresponds to some biconnected component of the constraint graph of an input QCN \mathcal{N} , each reasoner backtracks only within the search space defined by n , and considers a very small search tree to either verify or dispute that inconsistency with respect to the search tree that would have been obtained by the input QCN \mathcal{N} . Obviously, the time obtained for reasoner Solver+ is the time that it took it to serially reason with every QCN n , until it reached an unsatisfiable QCN (thus, assuring that the input QCN \mathcal{N} is also unsatisfiable by Corollary 5).

It is worth commenting on the performance of the reasoners with respect to network `hard-adm1`. Reasoners `Sarissa+` and `Phalanx∇+` present a performance that is slightly better than that of reasoners `Sarissa` and `Phalanx∇`, respectively. On the other hand, reasoners `GQR+` and `Phalanx+` present a performance that is slightly worse than that of reasoners `GQR` and `Phalanx`, respectively. This is due to the fact that the maximum order among the biconnected components of the constraint graph of `adm1` is very close to the order of the entire graph itself (see Table 2). Thus, in such cases, the use of Algorithm 4 may not lead to drastically improved performance, while sometimes due to the randomness of the heuristics in a reasoner, even slightly worse performance may be observed, as in this particular case.

Finally, we note that the results presented in Table 3 do not take into account the time needed for decomposing the networks with Algorithm 4, but only the time needed for performing satisfiability checks on the networks. However, the time needed for decomposing `hard-nuts`, `hard-adm1`, and `hard-gadm1` was negligible, and does not change the results qualitatively. In particular, a simple Python script that makes use of the `networkx`¹³ library was able to decompose `hard-nuts`, `hard-adm1`, and `hard-gadm1` in 0.2, 1.4, and 7.6 seconds, respectively.

6 Decomposition techniques in the constraint satisfaction problem framework

As noted in our introduction, a QCN is most efficiently modeled as an infinite-domain variant of a CSP through the use of a *relation algebra* (Ladkin & Maddux, 1994), which is also the approach we followed in our work. However, a QCN can also be encoded as a finite CSP instance (Renz & Nebel, 2001; Brand, 2004; Condotta *et al.*, 2006). In particular, given a QCN (V, C) , where $|V| = n$ we can obtain a CSP

¹² <http://pypy.org>

¹³ <https://networkx.github.io/>

instance as follows. Let X denote the set of variables containing a variable x_{ij} for each pair of variables $v_i, v_j \in V$ with $1 \leq i < j \leq n$. Then, our instance has the form $(X, B, DCon \cup TCon)$, where $DCon$ is the set of domain constraints $\{(x_{ij}, C_{ij}) \mid 1 \leq i < j \leq n\}$ and $TCon$ the set of ternary constraints $\{(x_{ij}, x_{ik}, x_{kj}), R_\diamond \mid 1 \leq i < j < k \leq n\}$ with $R_\diamond = \{(b, b', b'') \in B^3 \mid b \in b' \diamond b''\}$. Namely, $DCon$ restricts the values of a variable x_{ij} to the base relations of the corresponding qualitative constraint C_{ij} and $TCon$ encodes all the consistent paths of length 2 in the network. The resulting finite network has $\frac{n(n-1)}{2}$ variables and $\binom{n}{3}$ ternary constraints. A solution of this finite instance corresponds to a path-consistent atomic refinement of a given QCN, and vice versa (Condotta *et al.*, 2006). The main disadvantage of this approach is that we are not able to make use of tractable subclasses of relations. This can seriously impact the performance of satisfiability checking for calculi that heavily rely upon those subclasses, such as RCC-8 and IA. However, for large-sized qualitative calculi (viz., comprising hundreds of base relations) for which no tractable subclasses are known, a finite CSP encoding can provide a considerable performance gain (Westphal & Wöfl, 2009).

In light of the strong relation that exists between qualitative and ‘traditional’ constraint programming, it is worth mentioning some works in the latter paradigm that exploit the structure of constraint graphs in a similar manner to what we presented in our paper. The interested reader may review the cited works and obtain a deeper understanding on the analogy that exists between structural characteristics of QCNs and finite CSP instances. What is more important, the cited works may drive future research by enabling the reader to identify theoretical properties in the context of QSTR that can be used to adopt certain techniques for exploiting the structure of constraint graphs that exist in constraint programming.

Walsh (2001) measures the impact that the structure of a constraint graph can have on the performance of solving the *graph coloring problem*, which is the problem of coloring the vertices of a graph in such a way that no two adjacent vertices share the same color.

Baget and Tognetti (2001) propose a backtracking algorithm for solving CSP instances that exploits the biconnected components of a given constraint graph to reduce search space, permanently removing values and compiling partial solutions during exploitation.

Dechter and Pearl (1989) propose a constraint graph restructuring technique, based on tree decompositions, that guarantees that a large variety of queries could be answered swiftly either by sequential backtrack-free procedures, or by distributed constraint propagation methods.

Based on the work of Dechter and Pearl (1989), Jégou and Terrioux (2003) propose a framework for solving CSP instances that relies both on backtracking techniques and on the notion of tree decomposition of the constraint graphs. Notably, this mixed approach has been implemented and used successfully for practical CSP solving (Jégou & Terrioux, 2003).

Jégou *et al.* (2005) study several methods for computing a rough optimal tree decomposition and assess their relevance for solving CSP instances; the same authors also proposed dynamic heuristics for efficient backtrack search on tree decompositions of constraint graphs in Jégou *et al.* (2006, 2007).

Recently, Jégou and Terrioux (2014a, 2014b) introduced and exploited a new graph parameter, called *bag-connected treewidth*, which considers tree decompositions for which each cluster induces a connected graph. It is experimentally shown in Jégou and Terrioux (2014b) that such bag-connected tree decompositions significantly improve the solving of CSP instances by decomposition methods.

Finally, a presentation of the major structural constraint network decomposition methods discussed here is given in Gottlob *et al.* (2000).

7 Conclusion

To conclude, we surveyed the use and effect of decomposition-based techniques in qualitative constraint-based reasoning and showed that the decomposition-based approach presented in Nikolaou and Koubarakis (2014) for checking the satisfiability of QCNs of RCC-8 lacks soundness, as the notion of a *partitioning graph* defined in that work is not coherent with the use of patchwork upon which it solely relies. Further, we showed how that notion is beyond repair, unless it is reformulated to define a tree decomposition, implicitly or explicitly, and discussed the impact of these observations on the performance of the offered implementation in Nikolaou and Koubarakis (2014), which was already found to be poor in Sioutis (2014).

We think that future efforts regarding decomposition-based approaches utilizing parallelism, such as the approach attempted in Nikolaou and Koubarakis (2014), should rely on chordal graphs (tree decompositions into cliques), which can both be constructed and also yield a natural tree decomposition of their cliques in linear time (Diestel, 2012). The cliques can then be collected at no extra cost and parallelism *might* be efficiently utilized. It is an issue that looks promising and calls for further research. Toward that direction, we offered an approach that relies on a particular tree decomposition that is based on the biconnected components of the constraint graph of a given large QCN, and showed that it allows for cost-free utilization of parallelism for a qualitative constraint language that has patchwork for satisfiable atomic QCNs.

Acknowledgments

This work was funded by a PhD grant from Université d'Artois and region Nord-Pas-de-Calais. The material presented in this paper draws from the work of Sioutis *et al.* (2015c, 2015d).

References

- Allen, J. F. 1981. An interval-based representation of temporal knowledge. In *IJCAI*.
- Amaneddine, N., Condotta, J.-F. & Sioutis, M. 2013. Efficient approach to solve the minimal labeling problem of temporal and spatial qualitative constraints. In *IJCAI*.
- Baget, J.-F. & Tognetti, Y. S. 2001. Backtracking through biconnected components of a constraint graph. In *IJCAI*.
- Balbani, P., Condotta, J.-F. & Cerro, L. F. d. 2002. Tractability results in the block algebra. *Journal of Logic and Computation* **12**, 885–909.
- Barabasi, A.-L. & Bonabeau, E. 2003. Scale-free networks. *Scientific American* **288**, 50–59.
- Beek, P. v. & Manchak, D. W. 1996. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research* **4**, 1–18.
- Bhatt, M., Guesgen, H., Wöfl, S. & Hazarika, S. 2011. Qualitative spatial and temporal reasoning: emerging applications, trends, and directions. *Spatial Cognition & Computation* **11**, 1–14.
- Bliek, C. & Sam-Haroud, D. 1999. Path consistency on triangulated constraint graphs. In *IJCAI*.
- Bodirsky, M. & Wöfl, S. 2011. RCC8 is polynomial on networks of bounded treewidth. In *IJCAI*.
- Brand, S. 2004. Relation variables in qualitative spatial reasoning. In *KI*.
- Chmeiss, A. & Condotta, J.-F. 2011. Consistency of triangulated temporal qualitative constraint networks. In *ICTAI*.
- Condotta, J.-F. & D'Almeida, D. 2011. Consistency of qualitative constraint networks from tree decompositions. In *TIME*.
- Condotta, J.-F., Dalmeida, D., Lecoutre, C. & Sais, L. 2006. From qualitative to discrete constraint networks. In *KI Workshop on Qualitative Constraint Calculi*.
- Dechter, R. 2003. *Constraint Processing*. Elsevier Morgan Kaufmann.
- Dechter, R. & Pearl, J. 1989. Tree clustering for constraint networks. *Artificial Intelligence* **38**, 353–366.
- Del Genio, C. I., Gross, T. & Bassler, K. E. 2011. All scale-free networks are sparse. *Physical Review Letters* **107**, 178701.
- Diestel, R. 2012. *Graph Theory, 4th Edition*, **173**. Springer.
- Duckham, M., Li, S., Liu, W. & Long, Z. 2014. On redundant topological constraints. In *KR*.
- Dylla, F., Mossakowski, T., Schneider, T. & Wolter, D. 2013. Algebraic properties of qualitative spatio-temporal calculi. In *COSIT*.
- Elidan, G. & Gould, S. 2008. Learning bounded treewidth Bayesian networks. In *NIPS*.
- Frank, A. U. 1991. Qualitative spatial reasoning with cardinal directions. In *OGAI*.
- Gantner, Z., Westphal, M. & Wöfl, S. 2008. GQR—a fast reasoner for binary qualitative constraint calculi. In *AAAI Workshop on Spatial and Temporal Reasoning*.
- Garey, M. R., Johnson, D. S. & Stockmeyer, L. J. 1976. Some simplified NP-complete graph problems. *Theoretical Computer Science* **1**, 237–267.
- Goodwin, J., Dolbear, C. & Hart, G. 2008. Geographical linked data: the administrative geography of Great Britain on the semantic web. *Transactions in GIS* **12**, 19–30.
- Gottlob, G., Leone, N. & Scarcello, F. 2000. A comparison of structural CSP decomposition methods. *Artificial Intelligence* **124**, 243–282.
- Hazarika, S. M. 2012. *Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions*. IGI Global.
- Huang, J. 2012. Compactness and its implications for qualitative spatial and temporal reasoning. In *KR*.
- Huang, J., Li, J. J. & Renz, J. 2013. Decomposition and tractability in qualitative spatial and temporal reasoning. *Artificial Intelligence* **195**, 140–164.

- Jégou, P., Ndiaye, S. & Terrioux, C. 2005. Computing and exploiting tree-decompositions for solving constraint networks. In *CP*.
- Jégou, P., Ndiaye, S. & Terrioux, C. 2006. An extension of complexity bounds and dynamic heuristics for tree-decompositions of CSP. In *CP*.
- Jégou, P., Ndiaye, S. & Terrioux, C. 2007. Dynamic heuristics for backtrack search on tree-decomposition of CSPs. In *IJCAI*.
- Jégou, P. & Terrioux, C. 2003. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence* **146**, 43–75.
- Jégou, P. & Terrioux, C. 2014a. Bag-connected tree-width: a new parameter for graph decomposition. In *ISAAC*.
- Jégou, P. & Terrioux, C. 2014b. Tree-decompositions with connected clusters for solving constraint networks. In *CP*.
- Ladkin, P. B. & Maddux, R. D. 1994. On binary constraint problems. *Journal of the ACM* **41**, 435–469.
- Li, J. J., Huang, J. & Renz, J. 2009. A divide-and-conquer approach for solving interval algebra networks. In *IJCAI*.
- Li, S., Long, Z., Liu, W., Duckham, M. & Both, A. 2015. On redundant topological constraints. *Artificial Intelligence* **225**, 51–76.
- Ligozat, G. 1998. Reasoning about cardinal directions. *Journal of Visual Languages and Computing* **9**, 23–44.
- Ligozat, G. 2011. *Qualitative Spatial and Temporal Reasoning*, Iste Series. Wiley.
- Liu, W. & Li, S. 2012. Solving minimal constraint networks in qualitative spatial and temporal reasoning. In *CP*.
- Lutz, C. & Milicic, M. 2007. A tableau algorithm for DLs with concrete domains and GCIs. *Journal of Automated Reasoning* **38**, 227–259.
- Montanari, U. 1974. Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences* **7**, 95–132.
- Nebel, B. 1997. Solving hard qualitative temporal reasoning problems: evaluating the efficiency of using the ORD-Horn class. *Constraints* **1**, 175–190.
- Nebel, B. & Bürckert, H.-J. 1995. Reasoning about temporal relations: a maximal tractable subclass of Allen’s interval algebra. *Journal of the ACM* **42**, 43–66.
- Nikolaou, C. & Koubarakis, M. 2014. Fast consistency checking of very large real-world RCC-8 constraint networks using graph partitioning. In *AAAI*.
- Randell, D. A., Cui, Z. & Cohn, A. 1992. A spatial logic based on regions and connection. In *KR*.
- Renz, J. 1999. Maximal tractable fragments of the region connection calculus: a complete analysis. In *IJCAI*.
- Renz, J. 2002a. A canonical model of the region connection calculus. *Journal of Applied Non-Classical Logics* **12**, 469–494.
- Renz, J. 2002b. *Qualitative Spatial Reasoning with Topological Information*. Springer-Verlag.
- Renz, J. & Ligozat, G. 2005. Weak composition for qualitative spatial and temporal reasoning. In *CP*.
- Renz, J. & Nebel, B. 1999. On the complexity of qualitative spatial reasoning: a maximal tractable fragment of the region connection calculus. *Artificial Intelligence* **108**, 69–123.
- Renz, J. & Nebel, B. 2001. Efficient methods for qualitative spatial reasoning. *Journal of Artificial Intelligence Research* **15**, 289–318.
- Sioutis, M. 2014. Triangulation versus graph partitioning for tackling large real world qualitative spatial networks. In *ICTAI*.
- Sioutis, M. & Condotta, J.-F. 2014. Tackling large qualitative spatial networks of scale-free-like structure. In *SETN*.
- Sioutis, M., Condotta, J.-F. & Koubarakis, M. 2016. An efficient approach for tackling large real world qualitative spatial networks. *International Journal of Artificial Intelligence Tools* **25**, 1–33.
- Sioutis, M. & Koubarakis, M. 2012. Consistency of chordal RCC-8 networks. In *ICTAI*.
- Sioutis, M., Li, S. & Condotta, J.-F. 2015b. Efficiently characterizing non-redundant constraints in large real world qualitative spatial networks. In *IJCAI*.
- Sioutis, M., Salhi, Y. & Condotta, J.-F. 2015c. A simple decomposition scheme for large real world qualitative constraint networks. In *FLAIRS*.
- Sioutis, M., Salhi, Y. & Condotta, J.-F. 2015d. On the use and effect of graph decomposition in qualitative spatial and temporal reasoning. In *SAC*.
- Tarjan, R. E. & Yannakakis, M. 1984. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing* **13**, 566–579.
- Tarski, A. 1941. On the calculus of relations. *Journal of Symbolic Logic* **6**, 73–89.
- Vilain, M., Kautz, H. & Beek, P. v. 1990. Readings in qualitative reasoning about physical systems. In *Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report*, Weld, D. S. & de Kleer, J. (eds). Morgan Kaufmann Publishers Inc., 373–381.
- Walsh, T. 2001. Search on high degree graphs. In *IJCAI*.
- Westphal, M. & Hué, J. 2014. A concise Horn theory for RCC8. In *ECAI*.
- Westphal, M., Hué, J. & Wöflf, S. 2013. On the propagation strength of SAT encodings for qualitative temporal reasoning. In *ICTAI*.
- Westphal, M. & Wöflf, S. 2009. Qualitative CSP, finite CSP, and SAT: comparing methods for qualitative constraint-based reasoning. In *IJCAI*.
- Yannakakis, M. 1981. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods* **2**, 77–79.