Efficiently Enforcing Path Consistency on Qualitative Constraint Networks by Use of Abstraction

Michael Sioutis and **Jean François Condotta**
Université d'Artois, CRIL-CNRS UMR 8188, Lens, France
{sioutis,condotta}@cril.fr

Abstract

Partial closure under weak composition, or partial $\diamond$-consistency for short, is essential for tackling fundamental reasoning problems associated with qualitative constraint networks, such as the satisfiability checking problem, and therefore it is crucial to be able to enforce it as fast as possible. To this end, we propose a new algorithm, called PWC^{α}, for efficiently enforcing partial $\diamond$-consistency on qualitative constraint networks, that exploits the notion of *abstraction* for qualitative constraint networks, utilizes certain properties of partial $\diamond$-consistency, and adapts the functionalities of some state-of-the-art algorithms to its design. It is worth noting that, as opposed to a related approach in the recent literature, algorithm PWC^{α} is complete for arbitrary qualitative constraint networks. The evaluation that we conducted with qualitative constraint networks of the Region Connection Calculus against a competing state-of-the-art generic algorithm for enforcing partial $\diamond$-consistency, demonstrates the usefulness and efficiency of algorithm PWC^{α}.

1 Introduction

Qualitative Spatial and Temporal Reasoning (QSTR) is a major field of study in Artificial Intelligence, and Knowledge Representation & Reasoning in particular. This field has received a lot of attention over the past decades, as it extends to a plethora of areas and domains that include ambient intelligence, dynamic GIS, cognitive robotics, and spatiotemporal design [Bhatt *et al.*, 2011]. QSTR abstracts from numerical quantities of space and time by using qualitative descriptions instead (e.g., *precedes*, *contains*, *is left of*), thus providing a concise framework that allows for rather inexpensive reasoning about entities located in space and time.

The problem of representing and reasoning about qualitative information can be modeled as a qualitative constraint network (QCN), i.e., a network of constraints corresponding to qualitative spatial or temporal relations between spatial or temporal variables respectively, using a qualitative constraint language. A qualitative constraint language involves constraints defined over a finite set of binary relations, called *base relations* (or *atoms*) [Ligozat and Renz, 2004].

The fundamental reasoning problems associated with a given QCN $\mathcal{N}$ are the problems of *satisfiability checking*, *minimal labeling* (or *deductive closure*), and *redundancy* (or *entailment*) [Renz and Nebel, 2007]. In particular, the satisfiability checking problem is the problem of deciding if there exists a spatial or temporal valuation of the variables of $\mathcal{N}$ that satisfies its constraints, such a valuation being called a *solution* of $\mathcal{N}$, the minimal labeling problem is the problem of finding the strongest implied constraints of $\mathcal{N}$, and the redundancy problem is the problem of determining if a given constraint is entailed by $\mathcal{N}$ (that constraint being called redundant, as its removal does not change the solution set of the QCN). In general, for most qualitative constraint languages the satisfiability checking problem is NP-complete. Further, the redundancy problem, the minimal labeling problem, and the satisfiability checking problem are equivalent under polynomial Turing reductions [Golumbic and Shamir, 1993].

The vast amount, if not all, of the published works that study the aforementioned reasoning problems, use *partial $\diamond$-consistency* as a means to define practical algorithms for efficiently tackling them [Amaneddine *et al.*, 2013; Sioutis *et al.*, 2015; 2016b; Li *et al.*, 2015; Renz and Nebel, 2001; Nebel, 1997]. Given a QCN $\mathcal{N}$ and a graph G, partial $\diamond$-consistency with respect to G, denoted by $\overset{\diamond}{}_{G}$-consistency, entails (weak) consistency for all triples of variables in $\mathcal{N}$ that correspond to three-vertex cycles (triangles) in G. We note that if G is complete, $\overset{\diamond}{}_{G}$-consistency becomes identical to *$\diamond$-consistency* [Renz and Ligozat, 2005]. Hence, $\diamond$-consistency is a special case of $\overset{\diamond}{}_{G}$-consistency. In fact, earlier works have relied solely on $\diamond$-consistency; it was not until the introduction of chordal (or triangulated) graphs in QSTR, due to some generalized theoretical results of [Huang, 2012], that researchers started restricting $\diamond$-consistency to a triangulation of the constraint graph of an input QCN and benefiting from better complexity properties.

Currently, and to the best of our knowledge, the fastest algorithm for enforcing partial $\diamond$-consistency, which is presented in [Long *et al.*, 2016], is complete only for certain subsets of the set of allowed relations that can occur in a QCN. This fact significantly limits its practicality for arbitrary QCNs. Further, since those subsets define subclasses of relations that are in general smaller than the classes of relations for which partial $\diamond$-consistency is able to decide satisfiability, the algorithm of [Long *et al.*, 2016] is able to tackle

fewer QCNs in comparison with a generic algorithm for enforcing partial $\diamond$-consistency. To ameliorate this issue, we exploit the notion of abstraction for QCNs, which is an idea adopted from concepts of abstract interpretation [Cousot and Cousot, 1992]. In particular, a QCN is typically abstracted by relaxing some of its constraints in order to satisfy some property (if possible); in our case, we aim to abstract a given QCN is such a way that all of its constraints involve relations for which the algorithm of [Long *et al.*, 2016] is complete. This should allow us to harvest as much of the performance gains of that algorithm as possible for an arbitrary QCN $\mathcal{N}$, as we can use the algorithm to enforce partial $\diamond$-consistency on a proper abstraction of $\mathcal{N}$ very fast, and then incrementally apply appropriate constraint propagations on its output QCN to achieve partial $\diamond$-consistency pertaining to the original QCN $\mathcal{N}$. Specifically, we make the following contributions: (i) we provide an efficient and generic algorithm for enforcing partial $\diamond$-consistency that exploits the notion of abstraction for QCNs and makes full use of the algorithm of [Long *et al.*, 2016], (ii) we prove the correctness of our approach by utilizing certain properties of partial $\diamond$-consistency, such as idempotence and monotonicity, and (iii) we experimentally validate the usefulness and efficiency of our method against the state-of-the-art generic algorithm for enforcing partial $\diamond$-consistency of [Chmeiss and Condotta, 2011].

2 Preliminaries

A (binary) qualitative spatial or temporal constraint language, is based on a finite set B of *jointly exhaustive and pairwise disjoint* relations defined over an infinite domain D, which is called the set of *base relations* [Ligozat and Renz, 2004]. The base relations of a particular qualitative constraint language can be used to represent the definite knowledge between any two of its entities with respect to the level of granularity provided by the domain D. The set 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. The set 2^{B} is equipped with the usual set-theoretic operations of union and intersection, the converse operation, and the *weak composition* operation denoted by the symbol $\diamond$ [Ligozat and Renz, 2004]. For all $r \in 2^{\mathsf{B}}$, we have that $r^{-1} = \bigcup\{b^{-1} \mid b \in r\}$. The weak composition ($\diamond$) of two base relations $b, b' \in \mathsf{B}$ is defined as the smallest (i.e., strongest) relation $r \in 2^{\mathsf{B}}$ that includes $b \circ b'$, or, formally, $b \diamond b' = \{b'' \in \mathsf{B} \mid b'' \cap (b \circ b') \neq \emptyset\}$, where $b \circ b' = \{(x, y) \in \mathsf{D} \times \mathsf{D} \mid \exists z \in \mathsf{D}$ such that $(x, z) \in b \wedge (z, y) \in b'\}$ is the (true) composition of b and b'. For all $r, r' \in 2^{\mathsf{B}}$, we have that $r \diamond r' = \bigcup\{b \diamond b' \mid b \in r, b' \in r'\}$.

As an example, the Region Connection Calculus (RCC) is a first-order theory for representing and reasoning about mereotopological information [Randell *et al.*, 1992]. The domain D of RCC comprises all possible non-empty regular subsets of some topological space. These subsets do not have to be internally connected and do not have a particular dimension, but are commonly required to be regular *closed* [Renz, 2002]. Simply put, a subset X of a topological space is regular closed in that space, if X equals the closure of its interior. The base relations of RCC are the following ones: *disconnected* (DC), *externally connected* (EC), *equal* (EQ), *partially overlapping* (PO), *tangential proper part* (TPP), *tangential proper part inverse* ($TPPi$), *non-tangential proper part* ($NTPP$), and *non-tangential proper part inverse* ($NTPPi$). These eight base relations form the RCC-8 constraint language. Relation EQ is the identity relation Id of RCC-8. Other notable and well known qualitative spatial and temporal constraint languages include Point Algebra [Vilain and Kautz, 1986], Cardinal Direction Calculus [Ligozat, 1998; Frank, 1991], Interval Algebra [Allen, 1983], and Block Algebra [Balbiani *et al.*, 2002].

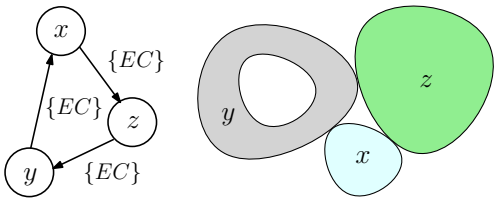

Figure 1: A QCN of RCC-8 along with a solution

The weak composition operation $\diamond$, the converse operation $^{-1}$, the union operation $\cup$, the complement operation $^{\mathsf{C}}$, and the total set of relations 2^{B} along with the identity relation Id of a qualitative constraint language, form an algebraic structure $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1}, ^{\mathsf{C}}, \cup)$ that can correspond to a *relation algebra* in the sense of Tarski [Tarski, 1941].

Proposition 1 ([Dylla *et al.*, 2013])**.** *The languages of Point Algebra, Cardinal Direction Calculus, Interval Algebra, Block Algebra, and* RCC*-8 are each a relation algebra with the algebraic structure* $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1}, ^{\mathsf{C}}, \cup)$.

In what follows, for a qualitative constraint language that is a relation algebra with the algebraic structure $(2^{\mathsf{B}}, \mathsf{Id}, \diamond, ^{-1}, ^{\mathsf{C}}, \cup)$, we will simply use the term *relation algebra*, as the algebraic structure will always be of the same format.

The problem of representing and reasoning about qualitative information can be modeled as a *qualitative constraint network* (QCN), defined in the following manner:

Definition 1. A *qualitative constraint network* (QCN) is a tuple (V, C) where: $V = \{v_1, \ldots, v_n\}$ is a non-empty finite set of variables, each representing an entity; and C is a mapping $C : V \times V \rightarrow 2^{\mathsf{B}}$ such that $C(v, v) = \{\mathsf{Id}\}$ for all $v \in V$ and $C(v, v') = (C(v', v))^{-1}$ for all $v, v' \in V$.

An example of a QCN of RCC-8 is shown in Figure 1. In particular, the QCN comprises the set of variables $\{x, y, z\}$ and the constraints $C(x, y) = C(y, z) = C(z, x) = \{EC\}$; for simplicity, converse relations as well as Id loops are not mentioned or shown in the figure.

Definition 2. Let $\mathcal{N} = (V, C)$ be a QCN, then: a *solution* of $\mathcal{N}$ is a mapping $\sigma : V \rightarrow \mathsf{D}$ such that $\forall (u, v) \in V \times V$, $\exists b \in C(u, v)$ such that $(\sigma(u), \sigma(v)) \in b$; $\mathcal{N}$ is *satisfiable* iff it admits a solution; a QCN is *equivalent* to $\mathcal{N}$ iff it admits the same set of solutions as $\mathcal{N}$; a *sub*-QCN $\mathcal{N}'$ of $\mathcal{N}$, denoted by $\mathcal{N}' \subseteq \mathcal{N}$, is a QCN (V, C') such that $C'(u, v) \subseteq C(u, v)$ $\forall u, v \in V$; the *constraint graph* of $\mathcal{N}$ is the graph (V, E) where $\{u, v\} \in E$ iff $C(u, v) \neq \mathsf{B}$ and $u \neq v$; and $\mathcal{N}$ is *trivially inconsistent* iff $\exists u, v \in V$ such that $C(u, v) = \emptyset$.

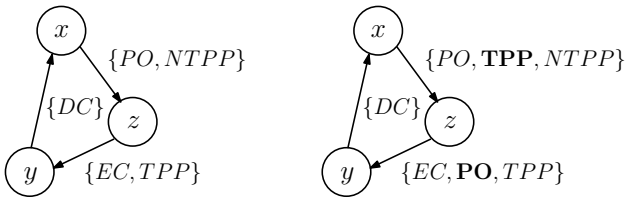

Figure 2: A QCN of RCC-8 along with an abstraction

We recall the following definition of $\diamond_G$-consistency, which, as noted in the introduction, is the basic local consistency used in the literature for solving fundamental reasoning problems of QCNs, such as the satisfiability checking problem.

Definition 3. Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, $\mathcal{N}$ is $\diamond_G$-*consistent* iff $\forall\{v_i, v_j\}, \{v_i, v_k\}, \{v_k, v_j\} \in E$ we have that $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$.

It is routine to formally prove the following properties of $\diamond_G$-consistency, which we will use in the sequel, as they naturally derive from respective properties of $\diamond$-consistency:

Property 1. Let $\mathcal{N} = (V, C)$ be a QCN and $G = (V, E)$ a graph. Then, the following properties hold:

- $\diamond_G(\mathcal{N}) \subseteq \mathcal{N}$ (viz., the $\diamond_G$-closure of $\mathcal{N}$) is the largest (w.r.t. $\subseteq$) $\diamond_G$-consistent sub-QCN of $\mathcal{N}$ (*Dominance*);
- $\diamond_G(\mathcal{N})$ is equivalent to $\mathcal{N}$ (*Equivalence*);
- $\diamond_G(\diamond_G(\mathcal{N})) = \diamond_G(\mathcal{N})$ (*Idempotence*);
- if $\mathcal{N}' \subseteq \mathcal{N}$ then $\diamond_G(\mathcal{N}') \subseteq \diamond_G(\mathcal{N})$ (*Monotonicity*).

We note again that if G is complete, $\diamond_G$-consistency becomes identical to $\diamond$-consistency [Renz and Ligozat, 2005], and, hence, $\diamond$-consistency is a special case of $\diamond_G$-consistency.

Definition 4. A *subclass* of relations is a subset $\mathcal{A} \subseteq 2^{\mathsf{B}}$ that contains the singleton relations of 2^{B} and is closed under converse, intersection, and weak composition.

Given a relation r of 2^{B} and a subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ containing B, $\mathcal{A}(r)$ denotes the smallest relation of $\mathcal{A}$ that includes r.

Definition 5. Given a QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and a subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ containing B, the *abstraction* of $\mathcal{N}$ w.r.t. to $\mathcal{A}$ and G, denoted by $\mathcal{A}_G(\mathcal{N})$, is the QCN (V, C') where $C'(v, v') = \mathcal{A}(C(v, v'))\ \forall\{v, v'\} \in E$.

As an example, Figure 2 illustrates the abstraction of a QCN of RCC-8 w.r.t. to the subclass that results from the closure of the base relations of RCC-8 under converse, intersection, and weak composition (which, in addition, contains B) and the complete graph on its set of variables.

Given three relations $r, r', r'' \in 2^{\mathsf{B}}$, we say that weak composition distributes over intersection if we have that $r \diamond (r' \cap r'') = (r \diamond r') \cap (r \diamond r'')$ and $(r' \cap r'') \diamond r = (r' \diamond r) \cap (r'' \diamond r)$.

Definition 6. A subclass $\mathcal{A}$ is *distributive* iff weak composition distributes over non-empty intersection $\forall r, r', r'' \in \mathcal{A}$.

Distributive subclasses of relations containing the universal relation B are defined for all of the qualitative constraint languages mentioned in Proposition 1 [Long and Li, 2015].

3 $\diamond_G$-Consistency by Use of Abstraction

The main idea behind our approach, is to benefit as much as possible from the performance characteristics of the algorithm for enforcing partial $\diamond$-consistency of [Long *et al.*, 2016], when given an arbitrary QCN $\mathcal{N}$. As noted in the introduction, that algorithm is complete only for certain subsets of 2^{B}. Those subsets essentially define subclasses of relations that are distributive (see Definition 6). Hence, to be able to make full use of the algorithm of [Long *et al.*, 2016], we utilize an abstraction of $\mathcal{N}$ with respect to some distributive subclass of relations, feed that abstraction to the algorithm, compare its output QCN $\mathcal{N}'$ with our initial QCN $\mathcal{N}$ and properly update $\mathcal{N}'$, and finally employ a generic algorithm for enforcing partial $\diamond$-consistency to incrementally apply appropriate constraint propagations on the updated QCN $\mathcal{N}'$ and achieve partial $\diamond$-consistency pertaining to the original QCN $\mathcal{N}$.

Algorithm 1: $\mathsf{PWC}^{\underline{\alpha}}(\mathcal{N}, G, \alpha, \mathcal{A})$

```
in  : A QCN N = (V, C), a graph G = (V, E), a
      bijection α : V → {0, 1, ..., |V| − 1}, and a
      subclass A of 2^B.
out : True or False, and a sub-QCN of N.
1  begin
2      N' = (V, C') ← A_G(N);
3      (decision, N') ← DPWC+(N', G, α);
4      if decision = False then
5          return (False, N);
6      e ← ∅;
7      foreach {v, v'} ∈ E do
8          r ← C'(v, v') ∩ C(v, v');
9          if r = ∅ then return (False, N);
10         if r ⊂ C'(v, v') then
11             e ← e ∪ {{v, v'}};
12             C'(v, v') ← r;
13             C'(v', v) ← r^{-1};
14     return PWC(N', G, e);
```

An algorithm that follows the aforementioned steps is presented in Algorithm 1, called $\mathsf{PWC}^{\underline{\alpha}}$ (which stands for partial closure under weak composition by use of abstraction), where subroutine DPWC+ in line 3 (which stands for directional partial closure under weak composition *plus*) corresponds to a generalized version of the algorithm of [Long *et al.*, 2016] and subroutine PWC in line 14 (which stands for partial closure under weak composition) is the state-of-the-art generic algorithm for enforcing partial $\diamond$-consistency of [Chmeiss and Condotta, 2011]. As opposed to the original algorithm of [Long *et al.*, 2016], its generalized version, viz., algorithm DPWC+, restricts consistency checks to constraints corresponding to edges of a given input graph G. Algorithm DPWC+, as well as its subroutine, algorithm DPWC, are presented in Algorithms 2 and 3 respectively. For completeness, algorithm PWC is presented in Algorithm 4. With respect to algorithm DPWC+, we can prove the following result:

Proposition 2 (cf. [Long *et al.*, 2016])**.** *Given a not trivially inconsistent* QCN $\mathcal{N} = (V, C)$ *defined over a distributive subclass of relations of a relation algebra, a chordal graph* $G = (V, E)$*, and a bijection* $\alpha : V \to \{0, 1, \ldots, |V| - 1\}$ *such that* $(\alpha^{-1}(|V|-1), \alpha^{-1}(|V|-2), \ldots, \alpha^{-1}(0))$ *is a perfect elimination ordering of* G*, algorithm* DPWC+ *returns* (True, $\diamond_G(\mathcal{N})$) *if* $\diamond_G(\mathcal{N})$ *is not trivially inconsistent, and* (False, $\mathcal{N}'$)

Algorithm 2: DPWC+$(\mathcal{N}, G, \alpha)$

in : A QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and a bijection $\alpha : V \to \{0, 1, \ldots, |V| - 1\}$.
out : True or False, and a sub-QCN of $\mathcal{N}$.

1 **begin**
2 $(\text{decision}, \mathcal{N}) \leftarrow \text{DPWC}(\mathcal{N}, G, \alpha)$;
3 **if** decision $=$ False **then**
4 **return** $(\text{False}, \mathcal{N})$;
5 **for** x **from** 1 **to** $|V| - 1$ **do**
6 $v \leftarrow \alpha^{-1}(x)$;
7 **foreach** $v' \in V \mid \{v', v\} \in E \wedge \alpha(v') < \alpha(v)$ **do**
8 $\text{adj} \leftarrow \{v'' \in V \mid \{v'', v\}, \{v', v''\} \in E \wedge$
9 $\alpha(v'') < \alpha(v)\}$;
10 $C(v, v') \leftarrow \bigcap_{v'' \in \text{adj}} C(v, v'') \diamond C(v'', v')$;
11 $C(v', v) \leftarrow (C(v, v'))^{-1}$;
12 **return** $(\text{True}, \mathcal{N})$;

Algorithm 3: DPWC$(\mathcal{N}, G, \alpha)$

in : A QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and a bijection $\alpha : V \to \{0, 1, \ldots, |V| - 1\}$.
out : True or False, and a sub-QCN of $\mathcal{N}$.

1 **begin**
2 **for** x **from** $|V| - 1$ **to** 1 **do**
3 $v \leftarrow \alpha^{-1}(x)$;
4 $\text{adj} \leftarrow \{v' \in V \mid \{v', v\} \in E \wedge \alpha(v') < \alpha(v)\}$;
5 **foreach** $v', v'' \in \text{adj} \mid \{v', v''\} \in E \wedge$
6 $\alpha(v') < \alpha(v'')$ **do**
7 $r \leftarrow C(v', v'') \cap (C(v', v) \diamond C(v, v''))$;
8 **if** $r = \emptyset$ **then return** $(\text{False}, \mathcal{N})$;
9 **if** $r \subset C(v', v'')$ **then**
10 $C(v', v'') \leftarrow r$;
11 $C(v'', v') \leftarrow r^{-1}$;
12 **return** $(\text{True}, \mathcal{N})$;

with $\mathcal{N}' \subseteq \mathcal{N}$ otherwise.

As shown in [Long *et al.*, 2016], algorithm DPWC+ (at least in its original and more particular version as it appears in that work) exhibits outstanding performance in comparison with algorithm PWC for enforcing partial $\diamond$-consistency on QCNs defined over a distributive subclass of relations. Due to space constraints, it is not possible to detail the functionality of that algorithm; however, it should suffice to say that it utilizes *partial* $\overleftarrow{\diamond}$*-consistency* as its core local consistency [Sioutis *et al.*, 2016a], which is enforced by algorithm DPWC, and exploits unique properties of distributive subclasses of relations to achieve partial $\diamond$-consistency in a given QCN defined over such a subclass. In summary, partial $\overleftarrow{\diamond}$-consistency is partial $\diamond$-consistency restricted (directionally) to a variable ordering α of the considered QCN. On the other hand, algorithm PWC is both efficient and complete for enforcing partial $\diamond$-consistency on arbitrary QCNs. With respect to algorithm PWC, we recall the following result:

Proposition 3 ([Chmeiss and Condotta, 2011])**.** *Given a not trivially inconsistent* QCN $\mathcal{N} = (V, C)$ *of a relation algebra, and a graph $G = (V, E)$, algorithm* PWC *returns* $(\text{True}, \overset{\diamond}{}_G(\mathcal{N}))$ *if* $\overset{\diamond}{}_G(\mathcal{N})$ *is not trivially inconsistent, and* $(\text{False}, \mathcal{N}')$ *with $\mathcal{N}' \subseteq \mathcal{N}$ otherwise.*

Algorithm 4: PWC$(\mathcal{N}, G, e \leftarrow \emptyset)$

in : A QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and optionally a set e such that $e \subseteq E$.
out : True or False, and a sub-QCN of $\mathcal{N}$.

1 **begin**
2 $Q \leftarrow (e$ **if** $e \neq \emptyset$ **else** $E)$;
3 **while** $Q \neq \emptyset$ **do**
4 $\{v, v'\} \leftarrow Q.pop()$;
5 **foreach** $v'' \in V \mid \{v, v''\}, \{v', v''\} \in E$ **do**
6 $r \leftarrow C(v, v'') \cap (C(v, v') \diamond C(v', v''))$;
7 **if** $r = \emptyset$ **then return** $(\text{False}, \mathcal{N})$;
8 **if** $r \subset C(v, v'')$ **then**
9 $C(v, v'') \leftarrow r$;
10 $C(v'', v) \leftarrow r^{-1}$;
11 $Q \leftarrow Q \cup \{\{v, v''\}\}$;
12 $r \leftarrow C(v'', v') \cap (C(v'', v) \diamond C(v, v'))$;
13 **if** $r = \emptyset$ **then return** $(\text{False}, \mathcal{N})$;
14 **if** $r \subset C(v'', v')$ **then**
15 $C(v'', v') \leftarrow r$;
16 $C(v', v'') \leftarrow r^{-1}$;
17 $Q \leftarrow Q \cup \{\{v'', v'\}\}$;
18 **return** $(\text{True}, \mathcal{N})$;

Having presented the structure and the core components of algorithm PWC$^{\alpha}$, we obtain the following lemma to be used for establishing the correctness of our algorithm in the sequel:

Lemma 1. *Let $\mathcal{N}' = (V, C')$ and $\mathcal{N} = (V, C)$ be two* QCN*s such that $\mathcal{N} \subseteq \mathcal{N}'$. Then, we have that $\overset{\diamond}{}_G(\mathcal{N}) \subseteq \overset{\diamond}{}_G(\mathcal{N}') \cap \mathcal{N}$.*

Proof. As $\mathcal{N} \subseteq \mathcal{N}'$, by monotonicity of $\overset{\diamond}{}_G$-consistency we have that $\overset{\diamond}{}_G(\mathcal{N}) \subseteq \overset{\diamond}{}_G(\mathcal{N}')$. Moreover, since it also holds that $\overset{\diamond}{}_G(\mathcal{N}) \subseteq \mathcal{N}$, it follows that $\overset{\diamond}{}_G(\mathcal{N}) \subseteq \overset{\diamond}{}_G(\mathcal{N}') \cap \mathcal{N}$. □

The following result states that algorithm PWC$^{\alpha}$ is complete for enforcing partial $\diamond$-consistency on arbitrary QCNs:

Theorem 1. *Given a not trivially inconsistent* QCN $\mathcal{N} = (V, C)$ *of a relation algebra, a distributive subclass of relations $\mathcal{A}$ of that relation algebra, a chordal graph $G = (V, E)$, and a bijection $\alpha : V \to \{0, 1, \ldots, |V| - 1\}$ such that $(\alpha^{-1}(|V| - 1), \alpha^{-1}(|V| - 2), \ldots, \alpha^{-1}(0))$ is a perfect elimination ordering of G, algorithm* PWC$^{\alpha}$ *returns* $(\text{True}, \overset{\diamond}{}_G(\mathcal{N}))$ *if* $\overset{\diamond}{}_G(\mathcal{N})$ *is not trivially inconsistent, and* $(\text{False}, \mathcal{N}')$ *with $\mathcal{N}' \subseteq \mathcal{N}$ otherwise.*

Proof. Let $\mathcal{N}' = \mathcal{A}_G(\mathcal{N})$. Clearly, $\mathcal{N} \subseteq \mathcal{N}'$. As $\mathcal{N}'$ is defined over a distributed subclass of relations, by Proposition 2 we have that, in line 3 of algorithm PWC$^{\alpha}$, algorithm DPWC+ returns $(\text{False}, \mathcal{N}'')$ with $\mathcal{N}'' \subseteq \mathcal{N}'$ if $\overset{\diamond}{}_G(\mathcal{N}')$ is trivially inconsistent, and $(\text{True}, \overset{\diamond}{}_G(\mathcal{N}'))$ otherwise. By monotonicity of $\overset{\diamond}{}_G$-consistency we have that $\overset{\diamond}{}_G(\mathcal{N}) \subseteq \overset{\diamond}{}_G(\mathcal{N}')$. Thus, algorithm PWC$^{\alpha}$ in line 5 returns $(\text{False}, \mathcal{N})$ only if $\overset{\diamond}{}_G(\mathcal{N})$ is trivially inconsistent. Assuming that $\overset{\diamond}{}_G(\mathcal{N}')$ is not

trivially inconsistent and the algorithm continues its execution, the operation $\diamond_G(\mathcal{N}') \cap \mathcal{N}$ is performed in lines 7–13 of algorithm PWC$^{\alpha}$. By Lemma 1 we have that $\diamond_G(\mathcal{N}) \subseteq \diamond_G(\mathcal{N}') \cap \mathcal{N}$. Hence, if $\diamond_G(\mathcal{N}') \cap \mathcal{N}$ defines a trivially inconsistent QCN, then $\diamond_G(\mathcal{N})$ is trivially inconsistent. Therefore, algorithm PWC$^{\alpha}$ in line 9 returns (False, $\mathcal{N}$) only if $\diamond_G(\mathcal{N})$ is trivially inconsistent. Let $\mathcal{M} = \diamond_G(\mathcal{N}') \cap \mathcal{N}$, and further assume that $\mathcal{M}$ is not trivially inconsistent and the algorithm continues its execution. $\mathcal{M}$ is a not trivially inconsistent QCN that results by tightening some of the constraints of $\diamond_G(\mathcal{N}')$. Note also that $\mathcal{M} \subseteq \mathcal{N}$; hence, by monotonicity of $\diamond_G$-consistency we have that $\diamond_G(\mathcal{M}) \subseteq \diamond_G(\mathcal{N})$. In addition, as we have already established that $\diamond_G(\mathcal{N}) \subseteq \mathcal{M}$, by monotonicity of $\diamond_G$-consistency we have that $\diamond_G(\diamond_G(\mathcal{N})) \subseteq \diamond_G(\mathcal{M})$, and by idempotence of $\diamond_G$-consistency it follows that $\diamond_G(\mathcal{N}) \subseteq \diamond_G(\mathcal{M})$. Consequently, we have that $\diamond_G(\mathcal{N}) = \diamond_G(\mathcal{M})$. Now, by utilizing the incremental functionality of algorithm PWC, we need only feed the algorithm with the QCN $\mathcal{M}$, the graph G, and the set of edges that correspond to the tightened constraints of $\diamond_G(\mathcal{N}')$ in order to obtain $\diamond_G(\mathcal{M})$ (see [Gerevini, 2005, Section 3]). Thus, by Proposition 3 we have that, in line 14 of algorithm PWC$^{\alpha}$, algorithm PWC returns (True, $\diamond_G(\mathcal{M})$) if $\diamond_G(\mathcal{M})$ is not trivially inconsistent, and (False, $\mathcal{M}'$) with $\mathcal{M}' \subseteq \mathcal{M}$ otherwise. Since $\diamond_G(\mathcal{N}) = \diamond_G(\mathcal{M})$, we deduce that algorithm PWC$^{\alpha}$ returns (True, $\diamond_G(\mathcal{N})$) if $\diamond_G(\mathcal{N})$ is not trivially inconsistent, and (False, $\mathcal{N}''$) with $\mathcal{N}'' \subseteq \mathcal{N}$ otherwise. □

Regarding time complexity, given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$ with a maximum vertex degree Δ, algorithm PWC$^{\alpha}$ runs in $O(\max\{\Delta^2|V|, \Delta|E||B|\})$ time, as the run-times of DPWC+ and PWC are $O(\Delta^2|V|)$ [Long *et al.*, 2016] and $O(\Delta|E||B|)$ [Chmeiss and Condotta, 2011] respectively. In terms of the number of triangles t in G, and assuming that $|V| + |E| \in O(t)$, algorithm PWC$^{\alpha}$ runs in $O(t|B|)$ time. However, depending on the percentage of distributive relations that are in the given QCN $\mathcal{N}$ and the quality of the abstraction that is obtained with respect to a given distributive subclass of relations, algorithm DPWC+ (employed in line 3 of algorithm PWC$^{\alpha}$) may diminish that run-time in practice, as it runs in $O(t)$ time [Long *et al.*, 2016], which is independent of the size of the set of base relations B. We will explore the aforementioned implication by means of a thorough experimental evaluation in the following section.

4 Experimental Evaluation

We evaluated the performance of an implementation of algorithm PWC$^{\alpha}$, against an implementation of the state-of-the-art generic algorithm for enforcing partial $\diamond$-consistency PWC, with a varied dataset of arbitrary QCNs of RCC-8.

Technical Specifications. The evaluation was carried out on a computer with an Intel Core i7-2820QM processor (which has a frequency of 2.30 GHz per CPU core), 8 GB of RAM, and the Trusty Tahr x86_64 OS (Ubuntu Linux). All algorithms were coded in Python and run using the standard CPython 2.7 interpreter. Only one CPU core was used.

Datasets and Measures. We considered random scale-free RCC-8 networks generated by the BA(n, m) model [Barabasi and Albert, 1999], the use of which in qualitative constraint-based reasoning is well motivated in [Sioutis *et al.*, 2016b], and random (unstructured) RCC-8 networks generated by the A$(\mathsf{n}, \mathsf{d}, \mathsf{l})$ model [Renz and Nebel, 2001], which has been traditionally employed over the past decades in the literature. Each of the aforementioned models was enriched with a parameter $\mathsf{r_d}$ that allowed us to specify the percentage of distributive relations we would like to have in our RCC-8 networks. In particular, we used the enriched BA$(\mathsf{n}, \mathsf{m}, \mathsf{r_d})$ model to create random scale-free RCC-8 constraints graphs of order n, with a preferential attachment value m, and with $\mathsf{r_d}\%$ of distributive relations, and the enriched A$(\mathsf{n}, \mathsf{d}, \mathsf{l}, \mathsf{r_d})$ model to create random RCC-8 constraint graphs of order n, with an average vertex degree d, with an average size of relation per constraint l (which defaults to $|\mathsf{B}|/2$ for model BA$(\mathsf{n}, \mathsf{m}, \mathsf{r_d})$), and with $\mathsf{r_d}\%$ of distributive relations. We generated a total of 400 RCC-8 networks; more specifically, we considered 10 satisfiable and 10 unsatisfiable RCC-8 networks for each of the models BA$(\mathsf{n} = 10\,000, \mathsf{m} = 2, \mathsf{r_d})$ and A$(\mathsf{n} = 1\,000, \mathsf{d} = 10.0, \mathsf{l} = 4.0, \mathsf{r_d})$, and for all values of $\mathsf{r_d}$ ranging from 0% to 90% with a step of 10%. Regarding distributive relations in particular, we considered the distributive subclass $\mathcal{D}_8^{64}$ of RCC-8 [Li *et al.*, 2015] as our selection pool and as an input to algorithm PWC$^{\alpha}$. Satisfiability or otherwise of our networks was guaranteed by the generic qualitative constraint-based reasoner GQR [Gantner *et al.*, 2008; Westphal *et al.*, 2009]. Finally, the *maximum cardinality search* algorithm [Tarjan and Yannakakis, 1984] was used to obtain a variable elimination ordering α of a given QCN $\mathcal{N}$ for PWC$^{\alpha}$, and a triangulation G of the constraint graph of $\mathcal{N}$ based on α for both PWC$^{\alpha}$ and PWC. We note that, with respect to our evaluation, any perfect elimination orderings and, hence, any respective chordal graphs would have been adequate, as they would have affected all involved algorithms proportionally and would not have qualitatively distorted the obtained results. An overview of the use of chordal graphs in the QSTR literature is presented in [Sioutis *et al.*, 2016c].

Our evaluation involved two measures, which we describe as follows. The first measure considers the number of *constraint checks* performed by an algorithm for enforcing partial $\diamond$-consistency. Given a QCN $\mathcal{N} = (V, C)$ and three variables $v_i, v_k, v_j \in V$, a constraint check occurs when we compute the relation $r = C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$ and check if $r \subset C(v_i, v_j)$, so that we can propagate its constrainedness if that condition is satisfied. The second measure concerns the CPU time and is strongly correlated with the first one, as the run-time of any proper implementation of an algorithm for enforcing partial $\diamond$-consistency should rely mainly on the number of constraint checks performed.

Results. The experimental results for random RCC-8 networks of models BA$(\mathsf{n}, \mathsf{m}, \mathsf{r_d})$ and A$(\mathsf{n}, \mathsf{d}, \mathsf{l}, \mathsf{r_d})$ are presented in Tables 1a and 1b respectively, where a fraction $\frac{x}{y}$ denotes that an approach required x seconds of CPU time and performed y constraint checks on average per dataset of networks during its operation. Regarding satisfiable networks, despite a rather sluggish performance of PWC$^{\alpha}$ when 0% of distributive relations were considered, in terms of being around 20% slower on average than PWC, it caught up fast, at around 20% of distributive relations, and constantly outperformed PWC from that point on, being nearly 80% faster on average than PWC when 90% of distributive relations were

Table 1: Evaluation of the performance of algorithms PWC and PWC$^{\alpha}$

(a) Evaluation with random scale-free RCC-8 networks of model BA($\mathsf{n} = 10\,000, \mathsf{m} = 2, \mathsf{r_d}$)

	min				average μ				max				standard deviation σ			
	SAT		UNSAT		SAT		UNSAT		SAT		UNSAT		SAT		UNSAT	
$\mathsf{r_d}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$
0%	$\frac{9.6s}{15k}$	$\frac{12.0s}{33k}$	$\frac{0.4s}{996}$	$\frac{2.7s}{19k}$	$\frac{12.0s}{22k}$	$\frac{14.5s}{41k}$	$\frac{1.0s}{6k}$	$\frac{3.6s}{24k}$	$\frac{16.4s}{42k}$	$\frac{18.2s}{61k}$	$\frac{2.6s}{13k}$	$\frac{5.8s}{32k}$	$\frac{1.8s}{7k}$	$\frac{1.8s}{7k}$	$\frac{0.7s}{4k}$	$\frac{0.9s}{4k}$
10%	$\frac{10.7s}{17k}$	$\frac{12.4s}{34k}$	$\frac{0.3s}{27}$	$\frac{0.7s}{164}$	$\frac{13.8s}{26k}$	$\frac{15.5s}{43k}$	$\frac{1.1s}{5k}$	$\frac{3.0s}{21k}$	$\frac{16.6s}{35k}$	$\frac{18.3s}{50k}$	$\frac{5.0s}{22k}$	$\frac{7.2s}{38k}$	$\frac{2.0s}{5k}$	$\frac{2.0s}{5k}$	$\frac{1.4s}{7k}$	$\frac{1.6s}{9k}$
20%	$\frac{13.4s}{28k}$	$\frac{13.4s}{42k}$	$\frac{0.3s}{26}$	$\frac{0.0s}{229}$	$\frac{16.2s}{48k}$	$\frac{16.2s}{59k}$	$\frac{2.1s}{10k}$	$\frac{3.7s}{23k}$	$\frac{19.1s}{85k}$	$\frac{18.8s}{92k}$	$\frac{9.8s}{37k}$	$\frac{14.5s}{55k}$	$\frac{1.6s}{17k}$	$\frac{1.7s}{16k}$	$\frac{2.8s}{11k}$	$\frac{3.7s}{13k}$
30%	$\frac{17.6s}{38k}$	$\frac{15.7s}{48k}$	$\frac{0.3s}{145}$	$\frac{0.1s}{404}$	$\frac{22.6s}{66k}$	$\frac{20.3s}{70k}$	$\frac{1.2s}{9k}$	$\frac{2.6s}{20k}$	$\frac{34.6s}{145k}$	$\frac{31.4s}{136k}$	$\frac{6.4s}{35k}$	$\frac{7.5s}{47k}$	$\frac{4.7s}{30k}$	$\frac{4.2s}{24k}$	$\frac{1.8s}{13k}$	$\frac{2.1s}{14k}$
40%	$\frac{18.4s}{52k}$	$\frac{15.0s}{55k}$	$\frac{0.3s}{56}$	$\frac{0.0s}{323}$	$\frac{22.7s}{76k}$	$\frac{18.5s}{73k}$	$\frac{2.1s}{12k}$	$\frac{2.1s}{12k}$	$\frac{30.0s}{129k}$	$\frac{24.9s}{121k}$	$\frac{15.1s}{61k}$	$\frac{12.4s}{55k}$	$\frac{4.1s}{25k}$	$\frac{3.6s}{21k}$	$\frac{4.4s}{17k}$	$\frac{3.6s}{18k}$
50%	$\frac{15.9s}{54k}$	$\frac{12.3s}{53k}$	$\frac{0.3s}{333}$	$\frac{0.0s}{114}$	$\frac{21.1s}{87k}$	$\frac{15.7s}{77k}$	$\frac{0.7s}{7k}$	$\frac{1.4s}{13k}$	$\frac{27.9s}{171k}$	$\frac{20.5s}{136k}$	$\frac{1.4s}{15k}$	$\frac{3.4s}{36k}$	$\frac{3.7s}{39k}$	$\frac{2.7s}{28k}$	$\frac{0.3s}{4k}$	$\frac{1.4s}{15k}$
60%	$\frac{22.4s}{70k}$	$\frac{14.4s}{61k}$	$\frac{0.3s}{569}$	$\frac{0.0s}{242}$	$\frac{27.5s}{141k}$	$\frac{18.5s}{109k}$	$\frac{1.6s}{15k}$	$\frac{1.1s}{11k}$	$\frac{40.2s}{372k}$	$\frac{29.3s}{274k}$	$\frac{4.6s}{39k}$	$\frac{3.0s}{37k}$	$\frac{5.1s}{81k}$	$\frac{4.2s}{58k}$	$\frac{1.5s}{12k}$	$\frac{1.2s}{16k}$
70%	$\frac{25.6s}{85k}$	$\frac{13.6s}{64k}$	$\frac{0.4s}{2}$	$\frac{0.0s}{22}$	$\frac{35.1s}{171k}$	$\frac{17.9s}{103k}$	$\frac{0.7s}{7k}$	$\frac{0.9s}{9k}$	$\frac{55.4s}{309k}$	$\frac{30.6s}{173k}$	$\frac{1.3s}{17k}$	$\frac{3.3s}{42k}$	$\frac{8.7s}{74k}$	$\frac{4.8s}{34k}$	$\frac{0.3s}{5k}$	$\frac{1.0s}{15k}$
80%	$\frac{26.5s}{109k}$	$\frac{10.6s}{67k}$	$\frac{0.3s}{54}$	$\frac{0.0s}{69}$	$\frac{32.7s}{161k}$	$\frac{13.1s}{87k}$	$\frac{0.5s}{5k}$	$\frac{0.2s}{1k}$	$\frac{36.7s}{221k}$	$\frac{15.2s}{128k}$	$\frac{1.3s}{25k}$	$\frac{0.5s}{2k}$	$\frac{3.0s}{28k}$	$\frac{1.5s}{16k}$	$\frac{0.3s}{7k}$	$\frac{0.2s}{694}$
90%	$\frac{21.1s}{93k}$	$\frac{5.9s}{49k}$	$\frac{0.3s}{131}$	$\frac{0.0s}{221}$	$\frac{35.2s}{227k}$	$\frac{8.9s}{85k}$	$\frac{0.5s}{3k}$	$\frac{0.2s}{1k}$	$\frac{42.1s}{293k}$	$\frac{12.9s}{117k}$	$\frac{0.7s}{10k}$	$\frac{0.4s}{2k}$	$\frac{5.7s}{60k}$	$\frac{2.0s}{20k}$	$\frac{0.1s}{3k}$	$\frac{0.2s}{700}$

(b) Evaluation with random (unstructured) RCC-8 networks of model A($\mathsf{n} = 1\,000, \mathsf{d} = 10.0, \mathsf{l} = 4.0, \mathsf{r_d}$)

	min				average μ				max				standard deviation σ			
	SAT		UNSAT		SAT		UNSAT		SAT		UNSAT		SAT		UNSAT	
$\mathsf{r_d}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$	PWC	PWC$^{\alpha}$
0%	$\frac{3.2s}{11k}$	$\frac{4.0s}{15k}$	$\frac{0.1s}{373}$	$\frac{0.8s}{5k}$	$\frac{3.4s}{12k}$	$\frac{4.2s}{16k}$	$\frac{0.6s}{4k}$	$\frac{1.3s}{8k}$	$\frac{3.7s}{13k}$	$\frac{4.5s}{17k}$	$\frac{1.3s}{8k}$	$\frac{2.1s}{12k}$	$\frac{0.1s}{554}$	$\frac{0.1s}{546}$	$\frac{0.4s}{3k}$	$\frac{0.5s}{3k}$
10%	$\frac{3.8s}{15k}$	$\frac{4.0s}{18k}$	$\frac{0.1s}{301}$	$\frac{0.0s}{115}$	$\frac{3.9s}{16k}$	$\frac{4.2s}{20k}$	$\frac{0.5s}{4k}$	$\frac{0.9s}{6k}$	$\frac{4.2s}{19k}$	$\frac{4.4s}{22k}$	$\frac{1.3s}{9k}$	$\frac{2.0s}{13k}$	$\frac{0.1s}{1k}$	$\frac{0.1s}{1k}$	$\frac{0.4s}{3k}$	$\frac{0.6s}{4k}$
20%	$\frac{3.9s}{18k}$	$\frac{4.1s}{20k}$	$\frac{0.1s}{715}$	$\frac{0.0s}{52}$	$\frac{4.3s}{22k}$	$\frac{4.4s}{23k}$	$\frac{0.6s}{5k}$	$\frac{0.9s}{7k}$	$\frac{4.6s}{25k}$	$\frac{4.5s}{26k}$	$\frac{1.4s}{11k}$	$\frac{2.1s}{16k}$	$\frac{0.2s}{2k}$	$\frac{0.1s}{2k}$	$\frac{0.4s}{3k}$	$\frac{0.6s}{4k}$
30%	$\frac{4.7s}{24k}$	$\frac{4.3s}{23k}$	$\frac{0.1s}{529}$	$\frac{0.0s}{107}$	$\frac{5.0s}{29k}$	$\frac{4.7s}{27k}$	$\frac{0.4s}{3k}$	$\frac{0.5s}{5k}$	$\frac{5.4s}{34k}$	$\frac{4.9s}{31k}$	$\frac{0.6s}{6k}$	$\frac{1.7s}{16k}$	$\frac{0.2s}{3k}$	$\frac{0.2s}{2k}$	$\frac{0.2s}{2k}$	$\frac{0.5s}{5k}$
40%	$\frac{4.9s}{30k}$	$\frac{4.2s}{26k}$	$\frac{0.2s}{898}$	$\frac{0.0s}{89}$	$\frac{5.6s}{37k}$	$\frac{4.6s}{32k}$	$\frac{0.7s}{6k}$	$\frac{0.4s}{6k}$	$\frac{6.1s}{47k}$	$\frac{5.0s}{38k}$	$\frac{1.6s}{17k}$	$\frac{0.9s}{11k}$	$\frac{0.3s}{5k}$	$\frac{0.3s}{3k}$	$\frac{0.4s}{5k}$	$\frac{0.4s}{5k}$
50%	$\frac{5.9s}{41k}$	$\frac{4.3s}{32k}$	$\frac{0.1s}{338}$	$\frac{0.0s}{46}$	$\frac{6.5s}{51k}$	$\frac{4.7s}{39k}$	$\frac{0.5s}{6k}$	$\frac{0.4s}{6k}$	$\frac{7.4s}{67k}$	$\frac{5.6s}{51k}$	$\frac{1.2s}{18k}$	$\frac{1.0s}{14k}$	$\frac{0.5s}{9k}$	$\frac{0.3s}{6k}$	$\frac{0.4s}{6k}$	$\frac{0.5s}{6k}$
60%	$\frac{6.7s}{54k}$	$\frac{4.2s}{37k}$	$\frac{0.2s}{2k}$	$\frac{0.0s}{110}$	$\frac{7.3s}{63k}$	$\frac{4.5s}{42k}$	$\frac{0.4s}{4k}$	$\frac{0.3s}{4k}$	$\frac{7.9s}{75k}$	$\frac{5.3s}{51k}$	$\frac{0.7s}{10k}$	$\frac{1.1s}{18k}$	$\frac{0.4s}{7k}$	$\frac{0.3s}{4k}$	$\frac{0.2s}{3k}$	$\frac{0.4s}{6k}$
70%	$\frac{7.6s}{66k}$	$\frac{3.8s}{39k}$	$\frac{0.1s}{331}$	$\frac{0.0s}{16}$	$\frac{8.8s}{94k}$	$\frac{4.5s}{54k}$	$\frac{0.4s}{5k}$	$\frac{0.2s}{5k}$	$\frac{10.5s}{143k}$	$\frac{5.4s}{81k}$	$\frac{1.5s}{22k}$	$\frac{1.2s}{22k}$	$\frac{1.0s}{26k}$	$\frac{0.6s}{15k}$	$\frac{0.4s}{6k}$	$\frac{0.4s}{8k}$
80%	$\frac{8.0s}{78k}$	$\frac{3.4s}{42k}$	$\frac{0.1s}{289}$	$\frac{0.0s}{496}$	$\frac{10.7s}{128k}$	$\frac{4.2s}{63k}$	$\frac{0.3s}{3k}$	$\frac{0.0s}{2k}$	$\frac{12.2s}{162k}$	$\frac{5.2s}{80k}$	$\frac{1.1s}{15k}$	$\frac{0.1s}{4k}$	$\frac{1.2s}{24k}$	$\frac{0.5s}{11k}$	$\frac{0.3s}{4k}$	$\frac{0.0s}{1k}$
90%	$\frac{8.3s}{79k}$	$\frac{2.3s}{37k}$	$\frac{0.1s}{178}$	$\frac{0.0s}{141}$	$\frac{11.0s}{144k}$	$\frac{3.0s}{58k}$	$\frac{0.4s}{6k}$	$\frac{0.0s}{1k}$	$\frac{14.8s}{250k}$	$\frac{5.0s}{110k}$	$\frac{0.9s}{14k}$	$\frac{0.1s}{2k}$	$\frac{1.9s}{55k}$	$\frac{0.7s}{20k}$	$\frac{0.3s}{5k}$	$\frac{0.0s}{711}$

considered. The same trend held true, more or less, for unsatisfiable networks, with the addition that the minimum time for refuting unsatisfiable networks for PWC$^{\alpha}$ was often significantly less than that of PWC, which is explained by the fact that PWC$^{\alpha}$ may generally detect certain inconsistencies at an earlier stage (see lines 5 and 9 in Algorithm 1).

5 Conclusion and Future Work

Partial $\diamond$-consistency is an essential local consistency for tackling fundamental reasoning problems associated with qualitative constraint networks (QCNs), such as the problems of satisfiability checking, minimal labeling, and redundancy. We proposed a new algorithm for efficiently applying partial $\diamond$-consistency on arbitrary QCNs, that exploits the notion of abstraction for QCNs, utilizes certain properties of partial $\diamond$-consistency, and adapts the functionalities of some state-of-the-art algorithms to its design. The evaluation that we conducted with QCNs of the Region Connection Calculus showed the usefulness and efficiency of our method. For future work, we would like to obtain a variation of our algorithm for efficiently enforcing singleton partial $\diamond$-consistency (i.e., partial $\blacklozenge$-consistency [Amaneddine *et al.*, 2013]) on arbitrary QCNs, which is an important local consistency for solving the minimal labeling problem, in particular, of QCNs.

References

[Allen, 1983] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26:832–843, 1983.

[Amaneddine *et al.*, 2013] Nouhad Amaneddine, Jean-François Condotta, and Michael Sioutis. Efficient Approach to Solve the Minimal Labeling Problem of Temporal and Spatial Qualitative Constraints. In *IJCAI*, 2013.

[Balbiani *et al.*, 2002] Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. Tractability Results in the Block Algebra. *J. Log. Comput.*, 12:885–909, 2002.

[Barabasi and Albert, 1999] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

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

[Chmeiss and Condotta, 2011] Assef Chmeiss and Jean-Francois Condotta. Consistency of Triangulated Temporal Qualitative Constraint Networks. In *ICTAI*, 2011.

[Cousot and Cousot, 1992] Patrick Cousot and Radhia Cousot. Abstract Interpretation Frameworks. *J. Log. Comput.*, 2:511–547, 1992.

[Dylla *et al.*, 2013] Frank Dylla, Till Mossakowski, Thomas Schneider, and Diedrich Wolter. Algebraic Properties of Qualitative Spatio-Temporal Calculi. In *COSIT*, 2013.

[Frank, 1991] Andrew U. Frank. Qualitative Spatial Reasoning with Cardinal Directions. In *ÖGAI*, 1991.

[Gantner *et al.*, 2008] Zeno Gantner, Matthias Westphal, and Stefan Wölfl. GQR-A Fast Reasoner for Binary Qualitative Constraint Calculi. In *AAAI Workshop on Spatial and Temporal Reasoning*, 2008.

[Gerevini, 2005] Alfonso Gerevini. Incremental qualitative temporal reasoning: Algorithms for the point algebra and the ord-horn class. *Artif. Intell.*, 166(1-2):37–80, 2005.

[Golumbic and Shamir, 1993] Martin Charles Golumbic and Ron Shamir. Complexity and Algorithms for Reasoning about Time: A Graph-Theoretic Approach. *J. ACM*, 40:1108–1133, 1993.

[Huang, 2012] Jinbo Huang. Compactness and its implications for qualitative spatial and temporal reasoning. In *KR*, 2012.

[Li *et al.*, 2015] Sanjiang Li, Zhiguo Long, Weiming Liu, Matt Duckham, and Alan Both. On redundant topological constraints. *Artif. Intell.*, 225:51–76, 2015.

[Ligozat and Renz, 2004] Gérard Ligozat and Jochen Renz. What Is a Qualitative Calculus? A General Framework. In *PRICAI*, 2004.

[Ligozat, 1998] Gerard Ligozat. Reasoning about cardinal directions. *J. Vis. Lang. Comput.*, 9:23–44, 1998.

[Long and Li, 2015] Zhiguo Long and Sanjiang Li. On Distributive Subalgebras of Qualitative Spatial and Temporal Calculi. In *COSIT*, 2015.

[Long *et al.*, 2016] Zhiguo Long, Michael Sioutis, and Sanjiang Li. Efficient Path Consistency Algorithm for Large Qualitative Constraint Networks. In *IJCAI*, 2016.

[Nebel, 1997] Bernhard Nebel. Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class. *Constraints*, 1:175–190, 1997.

[Randell *et al.*, 1992] David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions & Connection. In *KR*, 1992.

[Renz and Ligozat, 2005] Jochen Renz and Gérard Ligozat. Weak Composition for Qualitative Spatial and Temporal Reasoning. In *CP*, 2005.

[Renz and Nebel, 2001] Jochen Renz and Bernhard Nebel. Efficient Methods for Qualitative Spatial Reasoning. *J. Artif. Intell. Res.*, 15:289–318, 2001.

[Renz and Nebel, 2007] Jochen Renz and Bernhard Nebel. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. 2007.

[Renz, 2002] Jochen Renz. A Canonical Model of the Region Connection Calculus. *J. Appl. Non-Classical Logics*, 12:469–494, 2002.

[Sioutis *et al.*, 2015] Michael Sioutis, Sanjiang Li, and Jean-François Condotta. Efficiently Characterizing Non-Redundant Constraints in Large Real World Qualitative Spatial Networks. In *IJCAI*, 2015.

[Sioutis *et al.*, 2016a] M. Sioutis, Z. Long, and S. Li. Efficiently Reasoning about Qualitative Constraints through Variable Elimination. In *SETN*, 2016.

[Sioutis *et al.*, 2016b] Michael Sioutis, Jean-François Condotta, and Manolis Koubarakis. An Efficient Approach for Tackling Large Real World Qualitative Spatial Networks. *Int. J. Artif. Intell. Tools*, 25:1–33, 2016.

[Sioutis *et al.*, 2016c] Michael Sioutis, Yakoub Salhi, and Jean-François Condotta. Studying the use and effect of graph decomposition in qualitative spatial and temporal reasoning. *Knowl. Eng. Rev.*, 32:e4, 2016.

[Tarjan and Yannakakis, 1984] R. E. Tarjan and M. Yannakakis. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.

[Tarski, 1941] Alfred Tarski. On the calculus of relations. *J. Symb. Log.*, 6:73–89, 1941.

[Vilain and Kautz, 1986] Marc B. Vilain and Henry A. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *AAAI*, 1986.

[Westphal *et al.*, 2009] Matthias Westphal, Stefan Wölfl, and Zeno Gantner. GQR: A Fast Solver for Binary Qualitative Constraint Networks. In *AAAI Spring Symposium on Benchmarking of QSTR Systems*, 2009.