An Incremental Algorithm for Handling Qualitative Spatio-Temporal Information

Zhiguo Long
School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China

Qiyuan Hu
School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China

Hua Meng[1]
School of Mathematics, Southwest Jiaotong University, Chengdu, China

Michael Sioutis
Faculty of Information Systems and Applied Computer Sciences, Universität Bamberg, Germany

Abstract

In this paper, we present an online (incremental) algorithm for checking the satisfiability of qualitative spatio-temporal data, with direct implications to other fundamental knowledge representation and reasoning problems for such data, like the problems of deductive closure and redundancy removal. In particular, qualitative data come in the form of human-like, symbolic, descriptions such as "region x *contains* or *overlaps* region y", which are abundant in the Web of Data. Our approach is also able to maintain, to some extent, any sparse graph structure that may be inherent in the data, i.e., it acts parsimoniously and only tries to infer new information when needed for soundness and completeness. To this end, we complement our practical algorithm with certain theoretical results to assert its correctness and efficiency. A subsequent evaluation with publicly available large-scale real-world and random datasets against the state of the art, shows the interest and promise of our method.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Computing methodologies → Temporal reasoning; Computing methodologies → Spatial and physical reasoning

Keywords and phrases Online algorithm, qualitative data, spatio-temporal reasoning, satisfiability checking, knowledge representation and reasoning

Digital Object Identifier 10.4230/LIPIcs.COSIT.2022.5

Supplementary Material *Software (Source Code)*: `https://github.com/ZhiguoLong/DPCI_code`

Funding This work was supported by the National Natural Science Foundation of China (61806170), the Humanities and Social Sciences Fund of Ministry of Education (18XJC72040001), and the National Key Research and Development Program of China (2019YFB1706104).

1 Introduction

Real-world spatially or temporally annotated Big Data, largely due to their huge size, naturally pose significant technical challenges in terms of implementing efficient tools for associated practical tasks such as verification, repair, and visualization.[2][3] An Artificial Intelligence area that aims to handle such challenges relating to spatio-temporal data is Qualitative Spatio-Temporal Reasoning (QSTR) [19]. QSTR abstracts away the exact metric information that may be associated with such data, e.g., geometries or time points, and instead uses some natural language, qualitative, descriptions, e.g., *is right of*, *before*, or *inside*, to represent and reason about the data [19, 27]. This qualitative approach has applications

[1] Corresponding author
[2] `https://catalog.ldc.upenn.edu/LDC2006T08`
[3] `http://gadm.geovocab.org/`


© Zhiguo Long, Qiyuan Hu, Hua Meng, and Michael Sioutis;
licensed under Creative Commons License CC-BY 4.0

in a plethora of areas and domains that include cognitive robotics [9], spatio-temporal design [28], qualitative model generation from video [6], ambient intelligence [3], visual sensemaking [29], and qualitative case-based reasoning and learning [14], to name a few. A survey of representation languages, called qualitative constraint languages (or calculi) in QSTR, for various aspects of space and time, e.g., orientation or intervals, appears in [7].

Motivation

We are particularly interested in offering a parsimonious and incremental (online) algorithm for verifying, or, more formally, *checking the admissibility/satisfiability* of big qualitative spatio-temporal data of dynamic nature, which is an NP-hard task in the general case for most spatio-temporal calculi [7]. As satisfiability checking lies at the heart of most (if not all) reasoning tasks for such data, this algorithm also has direct implications to other fundamental knowledge representation and reasoning problems for the data, such as the problems of *deductive closure (minimal labeling)* and *redundancy removal* [25]; these are in fact polynomial-time Turing reducible to the satisfiability checking problem [12]. In sum, deductive closure concerns identifying all the information that either is or can be inferred to be true/valid and subsequently building a *minimal* knowledge base (KB), and redundancy removal concerns removing any information that can be inferred from the rest of the data and and subsequently building a non-redundant or *prime* KB. Clearly, these problems also relate to one another [18]. As noted in [13], a minimal KB is a quite useful knowledge compilation, since it can allow one to answer some queries in polynomial time that would otherwise be (at least) NP-hard. Indeed, in the context of QSTR, for instance, one could exploit minimality of a qualitative spatio-temporal KB to immediately deduce whether some task a should be scheduled before another task b. A non-redundant KB is particularly important in cases where we want to have our knowledge to be as concise as possible, yet without sacrificing essential information, like in the cases of pattern discovery or search in data mining [15, 16]; for instance, in [15] qualitative temporal data signatures are used for sepsis prediction, and explanation thereof, in intensive care medicine, and the authors identify the need to *optimize* such data with redundancy removal.

Related Work

We build upon the work of [26], where a technique for checking the satisfiability of a particular, so called *distributive* [21], class of qualitative spatio-temporal data is proposed, that performs a single pass over the spatial or temporal entities in the data and hence results in a dramatic performance boost with respect to the state of the art. That technique is based on a notion of weak local consistency, called $\overleftarrow{\diamond}_G$-consistency (directional partial path consistency with *weak composition* [20], to be detailed in Section 2), and was subsequently used as the backbone of a stronger consistency, called $\diamond_G$-consistency, in a later work [22]. However, the aforementioned technique is static in nature, i.e., it solely operates on fixed input data, which dramatically limits its applicability in real-world, dynamic, and time-critical situations; clearly, this limitation extends to other methods that rely on it, such as the one in [22] mentioned earlier. Here, we address this limitation and propose an online variant of the technique in [26] that is able to perform on-demand satisfiability checking of dynamic, incrementally-available, qualitative spatio-temporal data. Our approach mirrors to some extent that of [4], where an online algorithm is proposed for satisfiability checking of *quantitative* temporal data. Our own method is firstly distinguished by the fact that the involved consistency notions and operations are tailored to handle infinite domains and qualitative relations. Further, in our

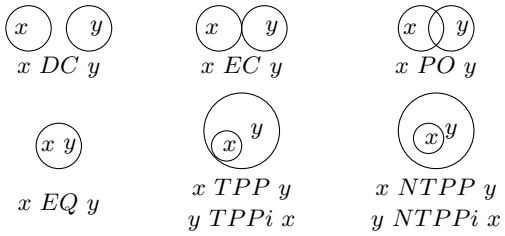


Figure 1 A $2D$ representation of the 8 base relations of RCC8, each one relating two potential regions x and y as in $x\ b\ y$; here, bi denotes the converse of b (formally b^{-1}).

work we handle more generic updates to existing (past) information, i.e., not just refinements of such information as in [4], but information that may be completely different to (or even contradicting) what exists in the KB. In our discussion, we focus on the spatial calculus of RCC8 [23] (to be detailed in Section 2) as far as examples, datasets, and evaluations are concerned, but it must be noted that the approach presented here is generic and applies to most widely adopted qualitative constraint languages, as they are listed in [7] (we explicitly state the required properties for a calculus in Section 2).

Contributions

In this paper, we make the following contributions: (i) we present an online and parsimonious algorithm for checking the satisfiability of qualitative spatio-temporal data, and recall certain implications to other fundamental knowledge representation and reasoning problems for such data, viz., deductive closure and redundancy removal, (ii) we establish the correctness of our algorithm with some theoretical results that relate to both its decision (output) and the triangulation technique that it uses to maintain any sparsity that may be inherent in the data, and (iii) we implement the algorithm and compare it against the state of the art with real-world and random datasets that scale up to thousands of variables, and subsequently demonstrate the efficiency of our approach.

The rest of the paper is organized as follows. In Section 2 we introduce some necessary terminology and notations that are followed thoughout the paper. In Section 3 we present our algorithm along with the results that establish its soundness and completeness. In Section 4 we evaluate an implementation of our algorithm against the state of the art. Finally, in Section 5 we conclude with a discussion and some directions for future work.

2 Preliminaries

A *binary* qualitative constraint language is based on a finite set B of *jointly exhaustive and pairwise disjoint* relations, called the set of *base relations* (*atoms*), that is defined over an infinite domain D (e.g., some topological space) [20]. These base relations represent definite knowledge between two entities of D; indefinite knowledge can be specified by a union of possible base relations, and is represented by the set containing them. The set B contains the identity relation Id, and is closed under the *converse* operation $(^{-1})$. The entire set of relations 2^{B} is equipped with the set-theoretic operations of union and intersection, the converse operation, and the *weak composition* operation denoted by $\diamond$ [20]. The weak composition $(\diamond)$ of two base relations $b, b' \in \mathsf{B}$ is the smallest relation $r \in 2^{\mathsf{B}}$ that includes $b \circ b'$; formally, $b \diamond b' = \{b'' \in \mathsf{B} : b'' \cap (b \circ b') \neq \emptyset\}$, where $b \circ b' = \{(x, y) \in \mathsf{D} \times \mathsf{D} : \exists z \in \mathsf{D}$ such that $(x, z) \in b \wedge (z, y) \in b'\}$. Finally, for all $r \in 2^{\mathsf{B}}$, $r^{-1} = \bigcup\{b^{-1} : b \in r\}$, and for all $r, r' \in 2^{\mathsf{B}}$, $r \diamond r' = \bigcup\{b \diamond b' : b \in r, b' \in r'\}$.

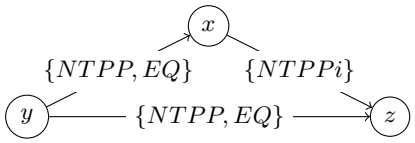


Figure 2 Illustration of a QCN $\mathcal{N}$ and $\overleftarrow{\diamond}_G$-consistency: $\mathcal{N}$ is $\overleftarrow{\diamond}_G$-consistent w.r.t. the ordering (z, y, x), i.e., x is "eliminated" first and z last, but not w.r.t. the ordering (x, y, z) (the base relation EQ between variables x and y would have to be removed).

As an illustration, consider the Region Connection Calculus (RCC), which is a first-order theory for representing and reasoning about mereotopological information [23]. The domain D of RCC comprises all possible non-empty regular closed subsets of some topological space [24]; these subsets serve as regions in RCC. A fragment of the Region Connection Calculus (RCC), denoted by RCC8, is the dominant qualitative spatial constraint language for representing and reasoning about qualitative spatial information [23]. In particular, RCC8 makes use of the mereotopological relations *disconnected* (DC), *externally connected* (EC), *equal* (EQ), *partially overlapping* (PO), *tangential proper part* (TPP) and its inverse ($TPPi$), and *non-tangential proper part* ($NTPP$) and its inverse ($NTPPi$) to encode knowledge about the spatial relations between two potential regions, as depicted in Figure 1. Relation EQ is the identity relation Id of RCC8.

The problem of representing and reasoning about qualitative spatial (or temporal) information may be tackled via the use of a *Qualitative Constraint Network*, defined in the following manner:

▶ **Definition 1** (QCN). *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 of an infinite domain* D*;*
- *and* C *is a mapping* $C : V \times V \to 2^{\mathsf{B}}$ *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 is shown in Figure 2; for clarity, neither converse relations nor Id loops are shown in the figure, but they are part of any QCN.

▶ **Definition 2.** *Let* $\mathcal{N} = (V, C)$ *be a* QCN*, then:*

- *a* solution *of* $\mathcal{N}$ *is a mapping* $\sigma : V \to \mathsf{D}$ *such that* $\forall v, v' \in V$*,* $\exists b \in C(v, v')$ *such that* $(\sigma(v), \sigma(v')) \in b$*, and* $\mathcal{N}$ *is* satisfiable *iff it admits a solution;*
- $\mathcal{N}$ *is* trivially inconsistent*, denoted by* $\emptyset \in \mathcal{N}$*, iff* $\exists v, v' \in V$ *such that* $C(v, v') = \emptyset$*;*
- *a* sub-QCN $\mathcal{N}'$ *of* $\mathcal{N}$*, denoted by* $\mathcal{N}' \subseteq \mathcal{N}$*, is a* QCN (V, C') *such that* $C'(v, v') \subseteq C(v, v')$ $\forall v, v' \in V$*;*
- $\mathcal{N}$ *is* atomic *iff* $\forall v, v' \in V$*,* $C(v, v') = \{b\}$ *with* $b \in \mathsf{B}$*;*
- *the* constraint graph *of* $\mathcal{N}$*, denoted by* $\mathsf{G}(\mathcal{N})$*, is the graph* (V, E) *where* $\{v, v'\} \in E$ *iff* $C(v, v') \neq \mathsf{B}$ *and* $v \neq v'$*;*

Given a QCN $\mathcal{N} = (V, C)$, a relation $C(u, v) = R$ of $\mathcal{N}$ can be denoted by $\mathcal{N}[u, v]$ and $(u\ R\ v)$ too, if it facilitates presentation.

We recall the definition of $\overleftarrow{\diamond}_G$-consistency [26], which is a fundamental local consistency for reasoning with QCNs. For simplicity, in what follows, a set of variables $V = \{v_1, v_2, \ldots, v_n\}$ and an ordering $(v_1, v_2, \ldots, v_n)$ is implied whenever a bijection $\alpha : V \to \{1, 2, \ldots, n\}$ is defined.

▶ **Definition 3** ($\overleftarrow{\diamond}_G$-consistency). *A* QCN $\mathcal{N} = (V, C)$ *is* $\overleftarrow{\diamond}_G$-consistent *with respect to a graph* $G = (V, E)$ *and an ordering* $(\alpha^{-1}(1),\alpha^{-1}(2),\ldots,\alpha^{-1}(n))$ *defined by a bijection* $\alpha : V \to \{1, 2, \ldots, n\}$ *iff for all* $v_i, v_j, v_k \in V$ *such that* $\{v_k, v_i\}, \{v_k, v_j\}, \{v_i, v_j\} \in E$, $\alpha(v_i) < \alpha(v_k)$, *and* $\alpha(v_j) < \alpha(v_k)$ *we have that* $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$.

In sum, $\overleftarrow{\diamond}_G$-consistency entails consistency for all *ordered* triples of variables of a QCN that correspond to triangles of a given graph G. This ordering can be specified by a bijection between the set of the variables of a QCN and a set of integers, and can be chosen randomly, or via an algorithm or some heuristic [26]. Figure 2 shows an example of how $\overleftarrow{\diamond}_G$-consistency may relate to a QCN.

In what follows, we assume that the following two properties for a given qualitative constraint language $\mathcal{L}$ hold; these hold for most well-known calculi [7, 8]:

$\mathcal{L}$ is a relation algebra. (1)

Every atomic $\diamond$-consistent QCN of $\mathcal{L}$ is satisfiable. (2)

These properties are only needed to simplify our algorithms and establish certain syntactic satisfiability conditions that allow us to be consistent with our claims. However, they can be further relaxed to accommodate some more "exotic" calculi, subject to complicating the algorithms and related operations and conditions of course.

3 Approach

In Algorithm 1 we present our online method for checking the satisfiability of a spatial or temporal QCN by means of $\overleftarrow{\diamond}_G$-consistency; we call this method DPCI. First, we briefly describe how DPCI works, and later we establish the assumptions under which it is sound and complete.

The basic idea of DPCI is that, when a constraint $C(v_i, v_k)$ in a $\overleftarrow{\diamond}_G$-consistent QCN is updated, we often do not need to run a full pass of the static algorithm for enforcing $\overleftarrow{\diamond}_G$-consistency as in [26]. Instead, an updated constraint $C(v_i, v_k)$ ($\alpha(v_i) < \alpha(v_k)$) may affect the constraints $C(v_i, v_j)$ where $\{v_k, v_i\}, \{v_k, v_j\}, \{v_i, v_j\} \in E$ and $\alpha(v_i) < \alpha(v_j) < \alpha(v_k)$, and any further updates might propagate in this way. In addition, if a constraint $C(v_i, v_j)$ in a $\overleftarrow{\diamond}_G$-consistent QCN is updated, then it might invalidate the previously established relation $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$ for some k s.t. $\{v_k, v_i\}, \{v_k, v_j\}, \{v_i, v_j\} \in E$ and $\alpha(v_i) < \alpha(v_j) < \alpha(v_k)$; this must be checked before propagating the updates.

As DPCI is an online algorithm, it assumes that some input has already been processed piece-by-piece in a serial fashion, and specifically that a $\overleftarrow{\diamond}_G$-consistent QCN has been formed. Clearly, in the base case, the initial QCN would not contain any constraints (or, equivalently, every constraint would be defined by a universal relation), which would make the QCN by default $\overleftarrow{\diamond}_G$-consistent. This base case of constructing an $\overleftarrow{\diamond}_G$-consistent QCN from scratch is presented in Algorithm 2. Specifically, a QCN $\mathcal{N}$ with no constraints is initialized in line 1 of the algorithm, which is then continuously updated with new constraints via calls to DPCI (lines 3–6); it is assumed that the new constraints are taken from some QCN $\mathcal{M}$ that is defined on the same set of variables as $\mathcal{N}$. The number m of m unprocessed constraints in line 4 may vary from 1 to the total number of (non-universal) constraints in $\mathcal{N}$. When $m = 1$, the simulation acts in a fully online manner, since it adds constraints 1 by 1, and when $m =$ *total number of constraints*, the simulation acts in a fully static manner, since it degenerates into a single application of DPCI. Next, we continue to describe how a call to DPCI behaves.

DPCI receives as input a $\overleftarrow{\diamond}_G$-consistent QCN $\mathcal{N} = (V, C)$ w.r.t. a graph $G = (V, E)$ and an ordering $(\alpha^{-1}(1),\alpha^{-1}(2),\ldots,\alpha^{-1}(n))$ defined by a bijection $\alpha : V \to \{1, 2, \ldots, n\}$, and a set of additional constraints C'. In the first two lines of the algorithm we introduce a dictionary

Algorithm 1 DPCI($\mathcal{N}, G, \alpha, C'$): Incremental Directional Path Consistency Algorithm.

Input: A $\overleftarrow{\diamond}_G$-consistent QCN $\mathcal{N} = (V, C)$ w.r.t. a graph $G = (V, E)$ and an ordering $(\alpha^{-1}(1),\alpha^{-1}(2),\ldots,\alpha^{-1}(n))$ defined by a bijection $\alpha : V \to \{1, 2, \ldots, n\}$; and a set of constraints C' of the form $(v_{i_0}\ R'_{i_0,j_0}\ v_{j_0})$, where $\{v_{i_0}, v_{j_0}\} \in E' \subseteq V \times V$.
Output: True or False, the updated graph G, and the updated QCN $\mathcal{N}$.

1 **foreach** $v_i \in V$ **do**
2 $P[v_i] \leftarrow \emptyset$;
3 $Q \leftarrow \emptyset$;
4 **foreach** $(v_{i_0}\ R'_{i_0,j_0}\ v_{j_0}) \in C'$ **do**
5 **foreach** v_k *s.t.* $\{v_{i_0}, v_k\}, \{v_k, v_{j_0}\} \in E$, $\alpha(v_k) > \alpha(v_{j_0}) > \alpha(v_{i_0})$ **do**
6 $R'_{i_0,j_0} \leftarrow R'_{i_0,j_0} \cap (\mathcal{N}[v_{i_0}, v_k] \circ \mathcal{N}[v_k, v_{j_0}])$;
7 **if** $R'_{i_0,j_0} = \emptyset$ **then**
8 **return** (False, $\emptyset, \emptyset$)
9 **if** $R'_{i_0,j_0} \neq \mathcal{N}[v_{i_0}, v_{j_0}]$ **then**
10 $\mathcal{N}[v_{i_0}, v_{j_0}] = R'_{i_0,j_0}$;
11 $\mathcal{N}[v_{j_0}, v_{i_0}] = R'_{j_0,i_0}$;
12 $P[v_{j_0}] \leftarrow P[v_{j_0}] \cup \{v_{i_0}\}$;
13 $Q \leftarrow Q \cup \{\alpha(v_{j_0})\}$;
14 **while** $Q \neq \emptyset$ **do**
15 $p \leftarrow \max(Q)$;
16 $v_k \leftarrow \alpha^{-1}(p)$;
17 $Q \leftarrow Q \setminus \{p\}$;
18 **foreach** $\{v_i, v_j\}$ *s.t.* $\alpha(v_i) < \alpha(v_j) < \alpha(v_k) \wedge \{v_i, v_k\}, \{v_j, v_k\} \in E \wedge (v_i \in P[v_k]$ *or* $v_j \in P[v_k])$ **do**
19 $T_{ij} \leftarrow \mathcal{N}[v_i, v_j] \cap (\mathcal{N}[v_i, v_k] \diamond \mathcal{N}[v_k, v_j])$;
20 $E \leftarrow E \cup \{\{v_i, v_j\}\}$;
21 **if** $T_{ij} = \emptyset$ **then**
22 **return** (False, $\emptyset, \emptyset$);
23 **if** $T_{ij} \neq \mathcal{N}[v_i, v_j]$ **then**
24 $\mathcal{N}[v_i, v_j] \leftarrow T_{ij}$;
25 $\mathcal{N}[v_j, v_i] \leftarrow T_{ij}^{-1}$;
26 $Q \leftarrow Q \cup \{\alpha(v_j)\}$;
27 $P[v_j] \leftarrow P[v_j] \cup \{v_i\}$;
28 **return** (True, $G, \mathcal{N}$).

of lists P to keep track of affected edges (corresponding to constraints) that need to be processed. Every key in the dictionary corresponds to a vertex $v \in V$ and maps to a list, and the edges that are incident on v *and* need to be processed are stored in the form of adjacent vertices to v in the respective list. In lines 4–13 the new constraints are added to the QCN. Specifically, in lines 5–8 a simple preprocessing takes place to ensure that no inconsistency is introduced, and in lines 9–13, in lack of such inconsistency, we move on and perform the constraint updates on the QCN and store all affected edges to be processed later on. Here, we also keep track of where the update occured with respect to our ordering, by pushing the respective index information into a queue Q (line 13). The main functionality of the algorithm occurs in lines 14–27. We start from the highest index in our ordering where an earlier corresponding constraint update occurred (line 15), and propagate the constrainedness of that update to constraints that are associated with variables that are indexed earlier in

Algorithm 2 FullSimulation$(\mathcal{M}, G, \alpha)$: Simulating the run of Algorithm 1 from scratch.

Input: A QCN $\mathcal{M} = (V, C)$, a graph $G = (V, E)$, and an ordering $(\alpha^{-1}(1), \alpha^{-1}(2), \ldots, \alpha^{-1}(n))$ defined by a bijection $\alpha : V \to \{1, 2, \ldots, n\}$.
Output: True or False, the updated graph G, and a new QCN $\mathcal{N}$.

1 $\mathcal{N} \leftarrow$ a QCN (V, C^*) s.t. $C^*(v, v') = \mathsf{B}$ for all $v, v' \in V$ with $v \neq v'$;
2 decision $\leftarrow$ True;
3 **while** $\exists$ *unprocessed constraints* $(v_i\ R_{i,j}\ v_j) \in C \wedge$ *decision* $\neq$ False **do**
4 Choose a subset C' of m unprocessed constraints $(v_i\ R_{i,j}\ v_j) \in C$;
5 Mark the chosen constraints as processed;
6 (decision, G, $\mathcal{N}$) $\leftarrow$ DPCI$(\mathcal{N}, G, \alpha, C')$;
7 **return** $(decision, G, \mathcal{N})$.

the ordering; as a reminder, this is the definition of $\overleftarrow{\diamond}_G$-consistency (see again Definition 3). It is important to note that when an edge $\{v_i, v_k\}$ or $\{v_j, v_k\}$ of G has been affected and we have that $\alpha(v_i) < \alpha(v_j) < \alpha(v_k)$, i.e., v_i is indexed earlier than v_j and v_j is indexed earlier than v_k in the ordering, we *must* also consider the edge $\{v_i, v_j\}$ in order for the constraints to propagate soundly with respect to enforcing $\overleftarrow{\diamond}_G$-consistency (line 20). Clearly, whenever further constraint updates occur, and no inconsistency is reported (lines 21–22), the information about the affected edges and the respective indices are maintained in the way that we have explained so far. This process is repeated until no more indices that correspond to constraint updates exist (Q empties), and consequently no more affected edges exist either.

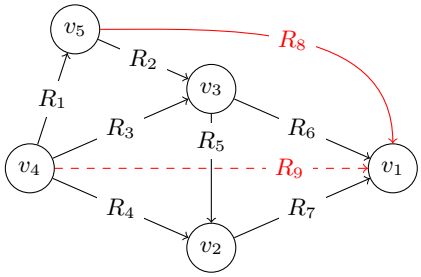


Figure 3 DPCI illustration.

▶ **Example 4.** Figure 3 illustrates how DPCI works. Initially, the QCN $\mathcal{N}$ contains the constraints shown as labelled black arrows in the figure, e.g., $(v_4\ R_1\ v_5)$, and it is $\overleftarrow{\diamond}_G$-consistent w.r.t. the ordering $(v_1, v_2, v_3, v_4, v_5)$. A new and different constraint $(v_5\ R_8\ v_1)$ is then added to the QCN, which is the same as updating the constraint between v_5 and v_1. Since there is no v_k satisfying the conditions in line 5, the loop in lines 5–8 will be skipped. The condition in line 9 is then satisfied by $R_8 \neq \mathcal{N}[v_{i_0}, v_{j_0}]$. Therefore, $\mathcal{N}[v_{i_0}, v_{j_0}]$ is updated to R_8, $P[v_5]$ is set to $\{v_1\}$, and $Q = \{5\}$. The process then moves to the loop in lines 14–27. In Q there is currently only one element, and $v_k = v_5$. The edges $\{v_i, v_j\}$ satisfying the conditions in line 18 are $\{v_1, v_3\}$ and $\{v_1, v_4\}$, because $P[v_5] = \{v_1\}$. Note that the edge $\{v_3, v_4\}$ will not be processed here because $v_3, v_4 \notin P[v_1]$, which is one of the benefits of our online algorithm. The contraints for the edges $\{v_1, v_3\}$ and $\{v_1, v_4\}$ may then get updated, and the edge $\{v_1, v_4\}$ is added to the graph (see the dashed edge). Suppose that only the constraint for $\{v_1, v_4\}$ is changed, then $\alpha(v_4) = 4$ is added to Q, and $P[v_4]$ is changed to $\{v_1\}$. In the next iteration of the loop, v_k would be v_4 and a similar process would be executed. On the other hand, if the initial QCN includes all the constraints shown

in Fig. 3, and the constraint $(v_4\ R_9\ v_1)$ is updated by another new constraint $(v_4\ R'_9\ v_1)$, then $R'_9 \leftarrow R'_9 \cap (R_1 \circ R_8)$ needs to be calculated. This is because we need to ensure that the updated $\mathsf{QCN}\ \mathcal{N}$ is $\overleftarrow{\diamond}_G$-consistent w.r.t. the ordering $(v_1, v_2, v_3, v_4, v_5)$, and thus for v_4, v_1, v_5 we should have $\mathcal{N}[v_4, v_1] \subseteq \mathcal{N}[v_4, v_5] \circ \mathcal{N}[v_5, v_1]$ (see lines 5–8).

Next, we recall and prove some results to establish the correctness of our algorithm. First, we introduce some necessary graph theoretic and other necessary concepts.

▶ **Definition 5** (PEO). *Given an undirected graph $G = (V, E)$ and an ordering $(\alpha^{-1}(1), \alpha^{-1}(2),\ldots,\alpha^{-1}(n))$ defined by a bijection $\alpha : V \to \{1, 2, \ldots, n\}$, let $F_k = \{v_j \in \mathsf{adj}(v_k) : \alpha(j) < \alpha(k)\}$; the ordering is a* perfect elimination ordering (PEO) *if and only if F_k induces a complete subgraph of G for every k.*

We can now recall the following result:

▶ **Lemma 6** ([11]). *A graph G is chordal iff G admits a PEO.*

A recall of *distributive subclasses* of relations is also required [21].

▶ **Definition 7.** *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 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 8** (distributive subclass). *A subclass $\mathcal{A} \subseteq 2^{\mathsf{B}}$ is* distributive *iff weak composition distributes over non-empty intersections for all relations r, r', $r'' \in \mathcal{A}$.*

We will use the following result to link $\overleftarrow{\diamond}_G$-consistency to chordal graphs, which are graphs that may allow for retaining the sparsity of a QCN [26]:

▶ **Proposition 9** ([26]). *Let $\mathcal{N} = (V, C)$ be a QCN defined over a distributive subclass of relations of a qualitative constraint language that satisfies properties 1 and 2, and $G = (V, E)$ a chordal graph s.t. $\mathsf{G}(\mathcal{N}) \subseteq G$ and $(\alpha^{-1}(1),\alpha^{-1}(2),\ldots, \alpha^{-1}(n))$ defined by a bijection $\alpha : V \to \{1, 2, \ldots, n\}$ is a PEO of G. If $\mathcal{N}$ is $\overleftarrow{\diamond}_G$-consistent w.r.t the PEO and $\emptyset \notin \mathcal{N}$, then $\mathcal{N}$ is satisfiable.*

We are ready to introduce our novel results.

▶ **Lemma 10.** *Let $G = (V, E)$ be a chordal graph, $(\alpha^{-1}(1),\alpha^{-1}(2),\ldots,\alpha^{-1}(n))$ defined by a bijection $\alpha : V \to \{1, 2, \ldots, n\}$ a PEO of G, and $|V| = n$. If E' is a set of new edges on V, and $G^{(n+1)} = (V, E^{(n+1)})$, where $E^{(n+1)} = E \cup E'$, and $E^{(k)} = E^{(k+1)} \cup E_k$, where $E_k = \{\{v_i, v_j\} : \{v_i, v_k\}, \{v_k, v_j\} \in E^{(k+1)}$ and $\alpha(v_i) < \alpha(v_j) < \alpha(v_k)\}$, then $G^{(1)} = (V, E^{(1)})$ is also a chordal graph with a same PEO.*

Proof. By Lemma 6, we only need to show that for any $v_k (k = n, \ldots, 1)$, the induced subgraph of $G^{(1)}$ on the set $F_k = \{v_i : \{v_i, v_k\} \in E^{(n)}$ and $\alpha(v_k) > \alpha(v_i)\}$ is a complete graph. In fact, by the definition of E_k, for any two vertices $v_i, v_j \in F_k$ s.t. $\alpha(v_i) < \alpha(v_j)$, there will be an edge $\{v_i, v_j\}$ in E_k. Since $E_k \subseteq E^{(1)}$, we know that the subgraph of $G^{(1)}$ on F_k is complete. Therefore, $G^{(1)} = (V, E^{(1)})$ is chordal and admits a same PEO. ◀

The previous lemma tells us that after adding new edges to a chordal graph G, we can easily construct a new chordal graph $G^{(1)}$ containing all the edges of G and maintain its PEO, by simply making sure that the induced subgraph on every F_k is complete. The following theorem shows the correctness of Algorithm 1.

▶ **Theorem 11.** *Let $\mathcal{N} = (V, C)$ be a* QCN *defined over a distributive subclass of relations of a qualitative constraint language that satisfies properties 1 and 2, $G = (V, E)$ a chordal graph s.t.* $\mathsf{G}(\mathcal{N}) \subseteq G$ *and* $(\alpha^{-1}(1), \alpha^{-1}(2), \ldots, \alpha^{-1}(n))$ *defined by a bijection $\alpha : V \to \{1, 2, \ldots, n\}$ is a PEO of G, and C' a set of new constraints of the form $(v_{i_0}\ R'_{i_0,j_0}\ v_{j_0})$, where $\{v_{i_0}, v_{j_0}\} \in E' \subseteq V \times V$ and $\alpha(v_{i_0}) < \alpha(v_{j_0})$. If $\mathcal{N}$ is $\overleftarrow{\diamond}_G$-consistent w.r.t the PEO and $\emptyset \notin \mathcal{N}$, then algorithm* DPCI *terminates and returns* $(\mathsf{True}, G', \mathcal{N}')$*, where $G' \supseteq G$ is still chordal and admits a same PEO, and $\mathcal{N}' \subseteq \mathcal{N}$ is $\overleftarrow{\diamond}_G$-consistent w.r.t. the PEO, if and only if $\mathcal{N} \setminus \{(v_{i_0} R_{i_0,j_0} v_{j_0}) : \{v_{i_0}, v_{j_0}\} \in E'\} \cup C'$ is satisfiable.*

Proof. The "only if" part is obvious by Proposition 9. For the "if" part we reason as follows. The algorithm returns False only when there is some v_i, v_j, v_k, where $\alpha(v_i) < \alpha(v_k)$, $\alpha(v_j) < \alpha(v_k)$, such that $\mathcal{N}[v_i, v_j] \cap (\mathcal{N}[v_i, v_k] \circ \mathcal{N}[v_k, v_j]) = \emptyset$. If $\mathcal{N} \setminus \{(v_{i_0} R_{i_0,j_0} v_{j_0}) : \{v_{i_0}, v_{j_0}\} \in E'\} \cup C'$ is satisfiable, then by Proposition 9 this cannot happen. Next, we assume that the algorithm returns True.

First, we show that the returned graph $G' \supseteq G$ is chordal. If we update G in the manner exactly as in Lemma 10 after adding E' and obtain G', then by Lemma 10 we know that G' is also chordal with a same PEO. In the sequel, we show that G' is exactly $G^{(1)}$ as defined in Lemma 10. Let $M^{(n+1)} = E \cup E'$ and $M^{(k)} = M^{(k+1)} \cup M_k$, where M_k is the set of edges added in line 20 in the iteration for v_k. We want to prove $M^{(k)} = E^{(k)}$ (with $E^{(k)}$ as in Lemma 10) for any k. Suppose by induction that $M^{(k)} = E^{(k)}$ holds for $u + 1 \leq k \leq n$; we will show $M^{(u)} = E^{(u)}$. To this end, we show that $\{v_i, v_u\} \in M^{(u)}$ for any $v_i, v_j \in V$ s.t. $\{v_i, v_u\}, \{v_u, v_j\} \in E^{(u+1)}$ and $\alpha(v_i) < \alpha(v_j) < \alpha(v_u)$. In fact, if the condition in line 18 of the algorithm is satisfied, i.e., $v_i \in P[v_u]$ or $v_j \in P[v_u]$, then $\{v_i, v_u\} \in M_u \subseteq M^{(u)}$. If $v_i \notin P[v_u]$ or $v_j \notin P[v_u]$, by the definition of $P[v_u]$ in the algorithm, we can see that $\{v_i, v_u\}$ and $\{v_j, v_u\}$ must have been in $M^{(u+1)} \subseteq M^{(u)}$ already. Therefore, in either case, $\{v_i, v_u\} \in M^{(u)}$ and thus $E^{(u)} \subseteq M^{(u)}$. Note that it is easy to see $M^{(u)} \subseteq E^{(u)}$, because the condition to add edges to M_u is stronger than to include an edge in E_u and thus $M_u \subseteq E_u$. In this way, we showed that the chordal graph $G^{(1)} = (V, E^{(1)})$ as defined in Lemma 10 is the same as the graph $G' = (V, M^{(1)})$ returned by the algorithm. Therefore, G' is also chordal and with a same PEO as G.

To show that the returned $\mathcal{N}'$ is $\overleftarrow{\diamond}_G$-consistent w.r.t. the PEO, suppose v_i, v_j, $v_k \in V$, $\{v_i, v_k\}, \{v_k, v_j\} \in E^{(n)}$, and $\alpha(v_i) < \alpha(v_j) < \alpha(v_k)$; we will show $\mathcal{N}'[v_i, v_j] \subseteq \mathcal{N}'[v_i, v_k] \circ \mathcal{N}'[v_k, v_j]$. If $v_i \in P[v_u]$ or $v_j \in P[v_u]$, then by the operations in lines 19, 24, and 25 of the algorithm, it is easy to see $\mathcal{N}'[v_i, v_j] \subseteq \mathcal{N}'[v_i, v_k] \circ \mathcal{N}'[v_k, v_j]$. If $v_i \notin P[v_u]$ or $v_j \notin P[v_u]$, then $\mathcal{N}[v_i, v_j] \subseteq \mathcal{N}'[v_i, v_k] \circ \mathcal{N}'[v_k, v_j]$ already holds from the beginning, because $\mathcal{N}'[v_i, v_k]$ and $\mathcal{N}'[v_i, v_k]$ are not changed by the algorithm, and $\mathcal{N}'[v_i, v_j]$ can only be refined (w.r.t. $\subset$) by the algorithm. Therefore, the returned $\mathcal{N}'$ is $\overleftarrow{\diamond}_G$-consistent w.r.t. to the PEO. ◀

We invite the reader to view Corollaries 3.2 and 3.3 in [26] for implications of the satisfiability checking result of Theorem 11 here to the fundamental knowledge representation and reasoning problems of *deductive closure (minimal labeling)* and *redundancy removal* respectively for spatial or temporal QCNs. The implications are direct, since we already mentioned in the introduction that these problems are polynomial-time Turing reducible to the satisfiability checking problem [12].

Complexity

Algorithm DPCI has a runtime of $O((|C'| + \Delta^2)|V|)$ for a given QCN $\mathcal{N} = (V, C)$ and a set of (new) constraints C', where Δ is the maximum vertex degree of the graph G returned through its output. In the worst case, the constraint updates may trigger a full-pass application of

$\overleftarrow{\diamond}_G$-consistency on the entire QCN, thus emulating the static algorithm of [26] with a runtime of $O(\Delta^2|V|)$. Specifically, if $\max(Q) = n$, i.e., the variable with order n is in Q, then a full-pass application of $\overleftarrow{\diamond}_G$-consistency will be triggered. Our algorithm also has some overhead costs for preprocessing the new constraints in C' (lines 4–13). In the worst case, for each of the new constraints, all vertices $v_k \in V$ satisfy the conditions in line 5, which in total will take $O(|C'||V|)$ time. Thus, the overall runtime of DPCI is $O((|C'| + \Delta^2)|V|)$.

4 Experimental Evaluation

We make use of the publicly available datasets from previous studies in QSTR, including real-world datasets from [18], i.e., `Footprint-`k (`F-`k, $k = 1, \ldots, 6$) and `StatArea-`k (`SA-`k, $k = 1, \ldots, 6$), and synthetic datasets from [26]. The datasets `F-1` to `F-6` have 108, 217, 434, 867, 1 736, and 3 470 variables, respectively, and `SA-1` to `SA-6` have 51, 100, 196, 376, 659, and 1 562 variables, respectively; all of them are satisfiable atomic QCNs of RCC8. The synthetic datasets range from 10^3 to 10^4 variables with a step size of 10^3. For each size n, we have 10 satisfiable QCNs of RCC8 over a distributive subclass of relations, generated using the model BA(n, m = 2) [1] for scale-free networks; networks of this type imitate real-world ones [1, 26]. All evaluations were performed on a PC with Ubuntu 18.04, CPU i7-8700 3.2GHz, RAM 64GB, and Python 3.8, and the time for each n is averaged over 10 tests.

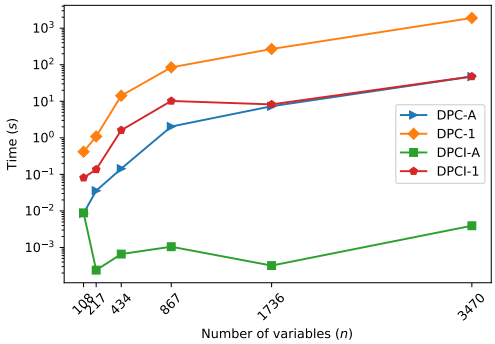
We compare the proposed online algorithm, viz., DPCI, against the state-of-the-art static algorithm of [26] for applying $\overleftarrow{\diamond}_G$-consistency on QCNs, viz., DPC. We note that we consider two ways of processing the additional constraints in C': (i) 1 *by* 1 and (ii) *altogether at once*. Regarding 1 *by* 1, each time we call DPCI we feed it a single new constraint from C', and we repeat this process until no constraints are left in C'; this method of using DPCI is denoted by DPCI-1. Regarding *altogether at once*, we feed all the additional constraints in C' at once to DPCI; this method is denoted by DPCI-A. Likewise, DPC-1 and DPC-A denote the 1 *by* 1 and *altogether at once* ways of processing constraints for the static algorithm, respectively. All of these approaches are analysed in our evaluations.

We want to measure the efficiency of enforcing $\overleftarrow{\diamond}_G$-consistency when new constraints arrive. Thus, we use the runtime as our performance metric. Specifically, we randomly generate 20 new constraints for each QCN, which can correspond to either existing or non-existing edges of each accompanying graph G. As a reminder, when edges do not exist in G, they have to be added to G *and* establish/maintain its chordality.

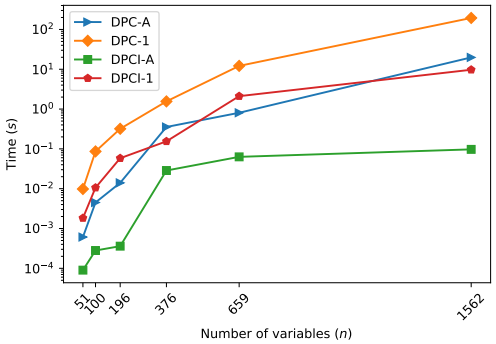
Results and Analysis

The results are shown in Figures 4 and 5, and it is clear that DPCI-A dominates all other approaches for all datasets. This is expected, as DPCI-A only updates part of a network, and processes the new constraints in bulk to avoid repeated updates triggered by individual new constraints. Notably, DPCI-1 is more efficient than DPC-1, by at least about an order of magnitude, which shows that our algorithm can indeed save a lot of time on calculations when single updates are considered. In fact, due to the time saved on calculations even when *repeated* single updates are considered, i.e., constraints are added in 1 *by* 1 fashion, DPCI-1 is also comparable to DPC-A on real-world datasets and more efficient on synthetic ones.

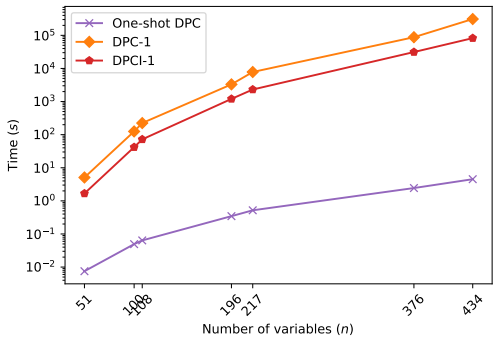
As the number of variables (and hence the number of constraints) increases, the runtime for all methods increases too, except for DPCI-A on the Footprint datasets due (to quick detection of inconsistency). This behaviour verifies that the runtimes of these methods are strongly related to the number of variables. Regarding the performance of DPCI-A on the Footprint datasets, it is due to the fact that these datasets are atomic networks, and are



(a) `Footprint`: Adding new constraints.

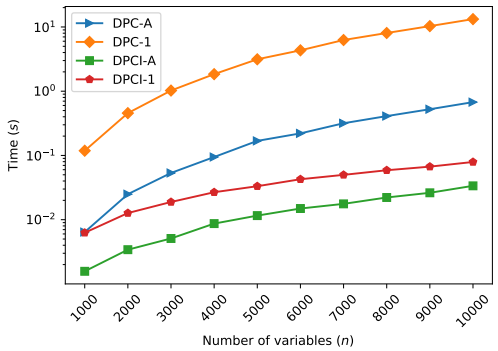


(b) `StatArea`: Adding new constraints.

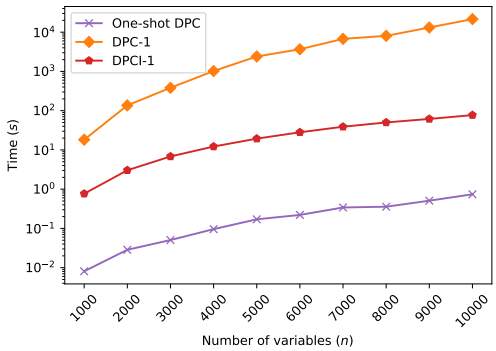


(c) `Footprint` (F-$\{1,\ldots,3\}$) + `StatArea` (SA-$\{1,\ldots,4\}$): Adding original constraints 1 *by* 1.

Figure 4 Results on real-world datasets, where 20 random new constraints are added in 4a and 4b and all original constraints are added in 1 *by* 1 fashion in 4c, i.e., QCNs are built from scratch (basically, Algorithm 2 with $m = 1$ in line 4). One-shot DPC applies $\overleftarrow{\mathcal{C}}_G$-consistency on the entire original QCN in a single step (our baseline, which serves as a practical lower bound).



(a) Adding new constraints.



(b) Adding original constraints 1 *by* 1.

Figure 5 Results on synthetic datasets, where 20 random new constraints are added in 5a, and all original constraints are added in 1 *by* 1 fashion in 5b, i.e., QCNs are built from scratch.

hence more likely to become inconsistent when some constraints are altered; DPCI-A can quickly detect this (due to its bulk constraint processing). Such inconsistencies appear very often in real-world data. For example, with the ever-increasing enrichment of the Semantic Web with geospatial data [10, 17], it is often the case that the geometries of geographical

objects are not captured correctly due to contradictory data of different sources. Thus, we can obtain inconsistent topological information when extracting topological relations from such geometries (e.g., two overlapping regions may be stated to be identical to a third region, which is impossible as they would also have to be identical to each other if that was the case); see also [5] in this respect.

Figure 5(b) shows the results of building $\overleftarrow{\diamond}_G$-consistent QCNs from scratch. For each QCN, we manually set all of its constraints to be the universal relation (viz., B), keep the graph G as is, and then add the original constraints back in 1 *by* 1 fashion. From this experiment, we can see how the proposed algorithm scales compared to the static algorithm for enforcing $\overleftarrow{\diamond}_G$-consistency, and subsequently how much better our online algorithm is compared to the static algorithm when new information arrives in a serial manner. It can be seen that, although the online algorithm takes more time to process new information serially (than in bulk, see baseline), it is much faster than using the static one in a serial manner.

In summary, we can conclude that the proposed online algorithm for enforcing $\overleftarrow{\diamond}_G$-consistency improves the static algorithm in processing dynamic information.

▶ Remark 12. To keep our evaluation concise, the number m of new constraints to add was fixed to 20 in Figures 4a, 4b, between 1 and n, the performance difference between DPCI-1 and DPC-1 largely remains qualitatively similar. On the other hand, the performance difference between DPCI-A and DPC-A might be affected by m, because the preprocessing step in lines 4–13 of Algorithm 1 might be more costly than the overhead of enforcing $\overleftarrow{\diamond}_G$-consistency in a static manner when m is close to n; in such cases, where almost the entire network is updated, DPC-A is by design the better choice.

5 Conclusion and Future Work

In this paper, we proposed an incremental (online) algorithm for checking the satisfiability of qualitative spatio-temporal data, which has important implications to other fundamental knowledge representation and reasoning problems for such data too, such as the problems of deductive closure and redundancy removal. Contrary to the state of the art, our approach acts parsimoniously and only infers new information when needed, subsequently maintaining soundness and completeness. An evaluation with publicly available large-scale real-world and random datasets against the state of the art, showed the interest and efficiency of our method. For future work, we would like to extend the current method with vertex-incremental capabilities [2], i.e., handle also the cases where new variables are incrementally made available, and not just new constraints among established variables.

References

1 Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

2 Anne Berry, Pinar Heggernes, and Yngve Villanger. A vertex incremental approach for maintaining chordality. *Discrete Mathematics*, 306(3):318–336, 2006.

3 Mehul Bhatt, Frank Dylla, and Joana Hois. Spatio-terminological Inference for the Design of Ambient Environments. In *COSIT*, pages 371–391, 2009.

4 Nicolas Chleq. Efficient Algorithms for Networks of Quantitative Temporal Constraints. In *CONSTRAINT@FLAIRS Conference*, pages 901–907, 1995.

5 Jean-François Condotta, Issam Nouaouri, and Michael Sioutis. A SAT Approach for Maximizing Satisfiability in Qualitative Spatial and Temporal Constraint Networks. In *KR*, pages 432–442, 2016.

6 Krishna Sandeep Reddy Dubba, Anthony G. Cohn, David C. Hogg, Mehul Bhatt, and Frank Dylla. Learning Relational Event Models from Video. *J. Artif. Intell. Res.*, 53:41–90, 2015.

7 Frank Dylla, Jae Hee Lee, Till Mossakowski, Thomas Schneider, André van Delden, Jasper van de Ven, and Diedrich Wolter. A Survey of Qualitative Spatial and Temporal Calculi: Algebraic and Computational Properties. *ACM Comput. Surv.*, 50(1):7:1–7:39, 2017.

8 Frank Dylla, Till Mossakowski, Thomas Schneider, and Diedrich Wolter. Algebraic Properties of Qualitative Spatio-temporal Calculi. In *COSIT*, pages 516–536, 2013.

9 Frank Dylla and Jan Oliver Wallgrün. Qualitative Spatial Reasoning with Conceptual Neighborhoods for Agent Control. *J. Intell. Robotic Syst.*, 48(1):55–78, 2007.

10 Max J. Egenhofer. Toward the semantic geospatial web. In *ACM-GIS*, pages 1–4, 2002.

11 Delbert Ray Fulkerson and Oliver Alfred Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15(3):835–855, 1965.

12 Martin Charles Golumbic and Ron Shamir. Complexity and Algorithms for Reasoning about Time: A Graph-Theoretic Approach. *J. ACM*, 40(5):1108–1133, 1993.

13 Georg Gottlob. On minimal constraint networks. *Artif. Intell.*, 191-192:42–60, 2012.

14 Thiago Pedro Donadon Homem, Paulo Eduardo Santos, Anna Helena Reali Costa, Reinaldo Augusto da Costa Bianchi, and Ramón López de Mántaras. Qualitative case-based reasoning & learning. *Artif. Intell.*, 283:103258, 2020.

15 Zina Ibrahim, Honghan Wu, and Richard Dobson. Modeling Rare Interactions in Time Series Data Through Qualitative Change: Application to Outcome Prediction in Intensive Care Units. In *ECAI*, pages 1826–1833, 2020.

16 Orestis Kostakis, Nikolaj Tatti, and Aristides Gionis. Discovering recurring activity in temporal networks. *Data Min. Knowl. Discov.*, 31(6):1840–1871, 2017.

17 Manolis Koubarakis, Manos Karpathiotakis, Kostis Kyzirakos, Charalampos Nikolaou, and Michael Sioutis. Data Models and Query Languages for Linked Geospatial Data. In *Reasoning Web*, pages 290–328, 2012.

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

19 Gérard Ligozat. *Qualitative Spatial and Temporal Reasoning*. ISTE Series. Wiley, 2011.

20 Gérard Ligozat and Jochen Renz. What Is a Qualitative Calculus? A General Framework. In *PRICAI*, pages 53–64, 2004.

21 Zhiguo Long and Sanjiang Li. On Distributive Subalgebras of Qualitative Spatial and Temporal Calculi. In *COSIT*, pages 354–374, 2015.

22 Zhiguo Long, Michael Sioutis, and Sanjiang Li. Efficient Path Consistency Algorithm for Large Qualitative Constraint Networks. In *IJCAI*, pages 1202–1208, 2016.

23 David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In *KR*, pages 165–176, 1992.

24 Jochen Renz. A Canonical Model of the Region Connection Calculus. *J. Appl. Non Class. Logics*, 12(3-4):469–494, 2002.

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

26 Michael Sioutis, Zhiguo Long, and Sanjiang Li. Leveraging Variable Elimination for Efficiently Reasoning about Qualitative Constraints. *Int. J. Artif. Intell. Tools*, 27(4):1–37, 2018.

27 Michael Sioutis and Diedrich Wolter. Qualitative Spatial and Temporal Reasoning: Current Status and Future Challenges. In *IJCAI*, pages 4594–4601, 2021.

28 Philip A. Story and Michael F. Worboys. A Design Support Environment for Spatio-Temporal Database Applications. In *COSIT*, pages 413–430, 1995.

29 Jakob Suchan, Mehul Bhatt, and Srikrishna Varadarajan. Out of Sight But Not Out of Mind: An Answer Set Programming Based Online Abduction Framework for Visual Sensemaking in Autonomous Driving. In *IJCAI*, pages 1879–1885, 2019.