Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

Encoding Large RCC8 Scenarios Using Rectangular Pseudo-Solutions

Zhiguo Long*
QCIS, FEIT
University of Technology Sydney
Australia
Zhiguo.Long@student.uts.edu.au

Steven Schockaert
School of Computer Science & Informatics
Cardiff University
United Kingdom
SchockaertS1@cardiff.ac.uk

Sanjiang Li
QCIS, FEIT
University of Technology Sydney
Australia
Sanjiang.Li@uts.edu.au

Abstract

Most approaches in the field of qualitative spatial reasoning (QSR) use constraint networks to encode spatial scenarios. The size of these networks is quadratic in the number of variables, which has severely limited the real-world application of QSR. In this paper, we propose another representation of spatial scenarios, in which each variable is associated with one or more rectangles. Instead of requiring these rectangles to define a solution of the corresponding constraint network, we construct sequences of rectangles that define partial solutions to progressively weaker constraint networks. We present experimental results that illustrate the effectiveness of this strategy.

Introduction

Qualitative spatial relations (e.g., part of, adjacent to, and overlapping with) offer a convenient interface between geometric representation and natural language, and as such form the corner stone of most spatial query languages. The vast majority of existing methods for representing and reasoning about qualitative spatial relations, however, do not readily scale to large datasets with e.g. tens of thousands of regions. The main culprit is that these methods typically represent spatial scenarios as a constraint network, whose size is quadratic in the number of regions. A constraint network is a tuple $N = (V, \mathcal{D}, \mathcal{C})$, where V is a set of variables, $\mathcal{D}$ is their domain, and $\mathcal{C}$ is a set of spatial or temporal constraints between the variables. In this paper, we are interested in the topological relations between pairs of geographical regions. This kind of information can be encoded in the popular RCC8 calculus, which is a fragment of the Region Connection Calculus (Randell, Cui, and Cohn 1992) supporting the definition of relations such as *disjoint*, *touch*, and *partially overlap*. To represent the RCC8 relations that hold between n regions $a_1, \cdots, a_n$, in general we need an RCC8 constraint for every pair of regions a_i and a_j with $i \neq j$.

On the other hand, Geographical Information Systems (GISs) typically rely on a geometric representation of spatial scenarios, where each region is represented as a complex polygon which may have holes and/or multiple components. This geometric representation scales linearly with the number of regions. However, the online computation of spatial relationships between polygons can nonetheless be computationally expensive. In particular, it requires a computation time that is proportional to the number of vertices, which can be quite large (e.g. in our test data we have encountered polygons with more than 30,000 vertices for encoding administrative areas). Moreover, for some regions, we may only know how they qualitatively relate to other regions, without having access to precise boundaries. For example, this is often the case with vernacular places (Vögele, Schlieder, and Visser 2003; Vasardani, Winter, and Richter 2013), as well as in applications that rely on extracting spatial information from natural language (Schockaert et al. 2008).

Therefore, the question arises: is there any representation from which the RCC8 relation between any two regions can be determined more efficiently than by direct geometric computation, while only requiring an amount of space which is linear in the number of regions, and which can handle geometric and qualitative representations (or a combination of both) as input?

The solution we propose in this paper is to construct a sequence of axis-aligned rectangles which define partial solutions of progressively weaker RCC8 constraint networks. Here we say that a rectangle in the plane is *axis-aligned* if each of its edges is parallel to either the x- or y-axis.

Example 1. Consider the regions shown in Figure 1(a). Clearly it is not possible to find a rectangular solution for the set N of all the RCC8 constraints induced by these regions. However, we can easily find a mapping S_0 defining rectangles for the regions in $V_0 = \{o_1, \ldots, o_5\}$ such that the constraints $N|_{V_0}$ among these regions are satisfied (see Figure 1(b)). We then weaken the RCC8 network for the regions by removing the constraints in $N|_{V_0}$. Now we can consider the mapping S_1 from the regions in $V_1 = \{o_1, \ldots, o_6\}$ to a second set of rectangles that satisfy all the remaining RCC8 constraints $N \setminus (N|_{V_0})$ (see Figure 1(c)). Note that any of the RCC8 constraints in N is satisfied by either S_0 or S_1. In a similar way, for any spatial configuration we can find a sequence of mappings $\langle S_0, \ldots, S_k \rangle$, such that the rectangles given by S_i satisfy the RCC8 constraints between regions in V_i,

*Corresponding author.

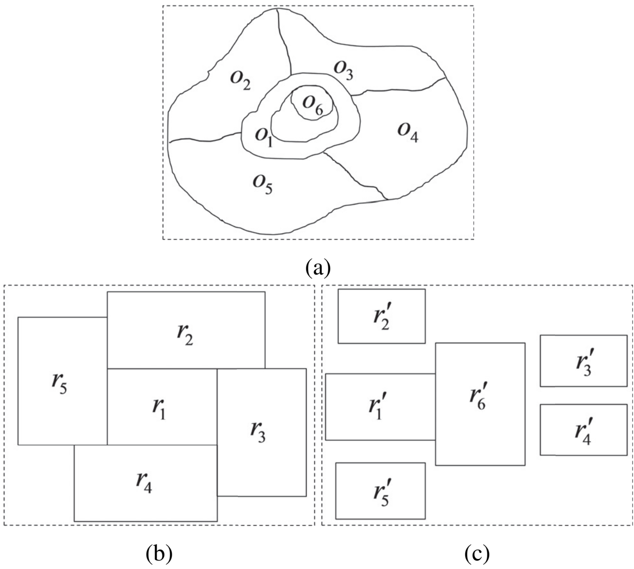


Figure 1: (a) Geometric representation of a spatial scenario with six regions. (b) The set of rectangles assigned by S_0 to regions in $V_0 = \{o_1, \ldots, o_5\}$. (c) The set of rectangles assigned by S_1 to regions in $V_1 = \{o_1, \ldots, o_6\}$.

which is a subnetwork of $N \setminus (\bigcup_{j=0}^{i-1} N|_{V_j})$, and moreover $N \setminus (\bigcup_{j=0}^{k} N|_{V_j}) = \emptyset$.

We call the sequence of mappings $\langle S_0, \ldots, S_k \rangle$ a *rectangular pseudo-solution* of N. With pseudo-solutions, we can determine the RCC8 relation between any two regions in N, by using a method that is different from the traditional way of retrieving the relation from a constraint network or calculating it from the polygons. For example, for o_1 and o_6, by finding out the first mapping that covers both o_1 and o_6, which is S_1 in Example 1, we will use the two rectangles (r'_1 and r'_6) defined by S_1 to calculate the RCC8 relation. When starting from large geometric representations, however, it might not be feasible to even consider the initial RCC8 network, due to its quadratic size. We address this issue by first partitioning the geometric dataset and then looking for rectangular pseudo-solutions of the RCC8 subnetworks induced by each partition. In the experiments, we show that the pseudo-solution representation allows for a more compact representation than existing baseline methods and can be used to encode RCC8 networks with over 100,000 regions. Moreover, since the construction is based on rectangles, retrieving the RCC8 relation from the rectangular pseudo-solution remains efficient.

The remainder of the paper is organized as follows. First, we provide an overview of related work in the next section, after which we recall some preliminaries from qualitative spatial reasoning. Then, we describe our algorithm for constructing the proposed representations. Subsequently, we present several improvements of our method. Finally, we empirically demonstrate the effectiveness of our approach.

Related Work

Dozens of relation models (also known as qualitative calculi) have been proposed for representing qualitative spatial and temporal information. Well-known examples include Point Algebra (PA) (Vilain and Kautz 1986), Interval Algebra (IA) (Allen 1983), Region Connection Calculus (Randell, Cui, and Cohn 1992), Cardinal Direction Calculus (CDC) (Goyal and Egenhofer 1997), and Rectangle Algebra (RA) (Guesgen 1989; Balbiani, Condotta, and Fariñas del Cerro 1999). In the past decades, significant research efforts have been spent on the central reasoning problem, viz. the consistency problem. For example, van Beek (1992) proposed an efficient $O(n^2)$ algorithm for deciding the consistency of a PA constraint network, Gerevini (2005) proposed an $O(n^3)$ time Incremental Path Consistency algorithm (IPC), and Balbiani et al. (1999) found a large tractable subclass of RA. We refer the reader to Cohn and Renz (2008) for more information.

Most of these works assume that a qualitative constraint network has been explicitly given in advance. In practice, this is rarely the case. Suppose there are one hundred thousand regions in a geometric dataset. Then a complete constraint network would involve billions of constraints.

Recently, several authors have started addressing the challenging problem of compactly encoding the qualitative relations that hold in a large geometric dataset S. For example, Fogliaroni (2012) introduced a framework called *spatial clustering index* that exploits the property of *clustering relations* to compactly encode the RCC8 scenario extracted from S. A relation R is a clustering relation if for any pair of objects $(a, b) \in R$ and any objects $a' \subseteq a, b' \subseteq b$ we have $(a', b') \in R$. To reduce the number of stored constraints in the representation, it organizes variables into clusters and then omits the constraints between individual variables belonging to two clusters whose corresponding geometric shapes (e.g., rectangles) are in a clustering relation. Fogliaroni discussed two implementations of the framework: one in which the clusters correspond to the cells of a grid, and one in which the clusters are obtained using an R-tree. Both implementations can be applied to RCC8 and CDC relations. However, for RCC8, the only clustering relation is the relation for disconnectedness ($\mathbf{DC}$) and therefore the omitted constraints are all $\mathbf{DC}$ constraints. Al-Salman (2014) proposed another implementation, which works well for CDC constraints but whose effectiveness for RCC8 is unclear.

Long et al. (2015) developed an algorithm, called the *MBR-based approach*, which uses the minimal bounding rectangles (MBRs) to compactly represent RCC8 and CDC constraints between regions. For RCC8 (resp. CDC), it only stores constraints between regions whose MBRs have no common points (resp. common interior points). It was shown that the representation thus obtained consistently leads to more compact representations for both RCC8 and CDC than the two implementations proposed by Fogliaroni.

The above methods start from a geometric dataset and sometimes still need to store a large number of constraints. In fact, for RCC8, there is a naive method called Non-$\mathbf{DC}$ which simply removes all the $\mathbf{DC}$ constraints. Despite its

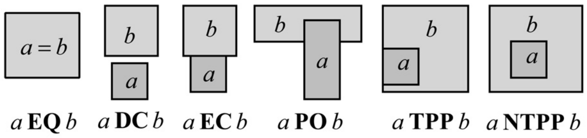


Figure 2: Illustration of basic RCC8 relations.

simplicity, this method is often surprisingly effective, since in many real-world datasets the majority of region pairs are indeed disconnected. It is also easy to see that the resulting representation is at least as compact as the results of all the aforementioned approaches (which are, however, not restricted to RCC8). In this paper, we will therefore use the Non-$\mathbf{DC}$ method as our main baseline.

Li et al. (2015) proposed the *prime subnetwork approach* which starts from a constraint network and derives a subnetwork that has no redundant constraints and is equivalent to the original network. For any consistent RCC8 network N defined over a distributive subalgebra (Long and Li 2015), they proved that N has a unique prime subnetwork and proposed an $O(n^3)$ algorithm for identifying the prime subnetwork. However, the only method which is available for retrieving the removed constraints is to use qualitative reasoning (e.g. enforcing path-consistency), which does not scale to large networks. Therefore, in our evaluation, we will not compare our approach with the prime subnetwork approach.

Preliminaries

We first recall some general notions about qualitative calculi, after which we briefly discuss a number of particular calculi that will be used in this paper.

Suppose $\mathcal{U}$ is a domain of spatial or temporal entities. A set of relations on $\mathcal{U}$ is *jointly exhaustive and pairwise disjoint* (JEPD) if any pair of entities from $\mathcal{U}$ is contained in exactly one of these relations. A *qualitative calculus* $\mathcal{M}$ on $\mathcal{U}$ is a finite Boolean subalgebra of (binary) relations on $\mathcal{U}$ such that the atom set of $\mathcal{M}$, written as $\mathcal{B}_\mathcal{M}$, is JEPD, closed under converse, and contains the identity relation $id_\mathcal{U}$. We call each relation in $\mathcal{B}_\mathcal{M}$ a *basic* relation and denote by $\star$ the universal relation $\mathcal{U} \times \mathcal{U}$. Since $\mathcal{B}_\mathcal{M}$ is JEPD, $\star$ is the union of all basic relations of $\mathcal{M}$.

RCC8 is a fragment of the Region Connection Calculus (Randell, Cui, and Cohn 1992). It is widely used for representing topological relations between regions. A region in this context is a bounded non-empty regular closed set in the plane, where a set is *regular closed* if it is identical to the closure of its interior. RCC8 has eight basic relations, i.e. $\mathbf{DC}$ (disconnected), $\mathbf{EC}$ (touch), $\mathbf{PO}$ (partially overlap), $\mathbf{NTPP}$ (non-tangential proper part) and its converse $\mathbf{NTPPi}$, $\mathbf{TPP}$ (tangential proper part) and its converse $\mathbf{TPPi}$, and $\mathbf{EQ}$ (equal). Figure 2 illustrates these relations. The same set of relations has also been identified for simple regions (i.e. regions without holes) in geographic information science (Egenhofer and Herring 1991; Egenhofer and Franzosa 1991; Smith and Park 1992).

PA (Vilain and Kautz 1986) models qualitative temporal relations between points on the real line. It uses the natural orderings of real numbers as basic relations, i.e. $<$, $>$, and $=$. IA (Allen 1983) represents the temporal relation between two events I_1, I_2 (represented as closed intervals on the real line) by considering the four PA relations between the endpoints of I_1 and I_2. RA (Guesgen 1989; Balbiani, Condotta, and Fariñas del Cerro 1999) characterizes the relation between two *axis-aligned rectangles* a and b by considering the IA relations between their x- and y-projections. A basic RA relation can then be represented as $\alpha \otimes \beta$, where α, β are two basic IA relations. In this paper, rectangles are always assumed to be axis-aligned.

Many qualitative reasoning tasks operate on qualitative constraint networks. Let $\mathcal{M}$ be a qualitative calculus. A *qualitative constraint* over $\mathcal{M}$ has the form (xRy), where R is a relation from $\mathcal{M}$. A *qualitative constraint network* (QCN or simply *network*) over $\mathcal{M}$ is a tuple $N = (V, \mathcal{D}, \mathcal{C})$, where V is a set of variables, $\mathcal{D}$ is the domain of $\mathcal{M}$, and $\mathcal{C}$ is a set of spatial or temporal constraints defined over $\mathcal{M}$. We say that a network N over $\mathcal{M}$ is a *scenario* if every relation in N is a basic relation from $\mathcal{M}$. A *solution* of a network $N = (V, \mathcal{D}, \mathcal{C})$ is an instantiation of the variables in V to entities in $\mathcal{D}$ such that every constraint in $\mathcal{C}$ is satisfied. A *partial solution* of a network N is a solution of the subnetwork $N|_{V_0}$ of N for some $V_0 \subset V$, where

$$N|_{V_0} = \{(uRv) \mid (uRv) \in N \text{ and } u, v \in V_0\}.$$

We say that a solution of an RCC8 network is *rectangular* if all variables are assigned to axis-aligned rectangles.

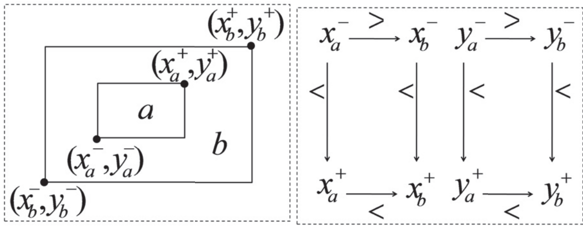
PA Representations of Basic RCC8 Relations

When restricted to the set of axis-aligned rectangles, each basic RCC8 relation is the union of one or more basic RA relations (cf. (Papadias et al. 1995, Figure 4) and (Li and Cohn 2012, Figure 7)). For example, $\mathbf{NTPP}$, when restricted to rectangles, is identical to the basic RA relation $\mathsf{d} \otimes \mathsf{d}$ (see Figure 3(a) for an illustration), where d is the basic IA relation which specifies that one interval is *during* the other. For $\mathbf{NTPPi}$ and $\mathbf{EQ}$, we similarly find that they correspond to a unique basic RA relation, but each of the other five basic RCC8 relations contains more than one basic RA relations. In fact, the union of these basic RA relations contained in each of **DC**, **EC**, **PO**, and **TPP/TPPi** is a non-basic RA relation which is outside the largest known tractable subclass of RA. This suggests that it might be NP-hard to determine if a basic RCC8 network has a rectangular solution.

In this paper, we decompose each of $\mathbf{DC}$, $\mathbf{EC}$, $\mathbf{PO}$, and $\mathbf{TPP}/\mathbf{TPPi}$ into RA relations that are PA representable. Recall that an interval relation ρ is *PA representable* or *pointisable* (Ladkin and Maddux 1988; van Beek and Cohen 1990) if there exists a PA network N over $\{x_1^-, x_1^+, x_2^-, x_2^+\}$ such that $(x_1^- < x_1^+)$ and $(x_2^- < x_2^+)$ are in N and ρ is identical to the solution set $\mathsf{sol}(N)$ of N, i.e.

$$\rho = \{([a^-, a^+], [b^-, b^+]) \mid \langle a^-, a^+, b^-, b^+ \rangle \in \mathsf{sol}(N)\}.$$

We say that an RA relation is PA representable if it is the product of two IA relations that are PA representable.



(a) (b)

Figure 3: (a) Two rectangles in a basic RA relation $\mathsf{d} \otimes \mathsf{d}$. (b) The corresponding PA relations for the endpoints.

Definition 1. Let R be a basic RCC8 relation. Suppose N is a PA network over $V = \{x_1^-, x_1^+, x_2^-, x_2^+, y_1^-, y_1^+, y_2^-, y_2^+\}$ such that $(x_i^- < x_i^+)$ and $(y_i^- < y_i^+)$ are in N for $i = 1, 2$. We say that N is a *PA representation* of R if there is an RA relation contained in R that is PA representable by N, i.e. identical to the solution set of N.

In the algorithm we will present in the next section, each of **NTPP/NTPPi** and **EQ** has exactly one PA representation (Figure 3(b) shows the one for **NTPP**); **DC**, **EC**, and **TPP/TPPi** each have four PA representations; and **PO** has 16 PA representations. These PA representations correspond to maximal RA relations which are PA representable and contained in the corresponding basic RCC8 relation.

Pseudo-Solutions

A network N can be *weakened* by removing one or several constraints from N.

Definition 2. Given an RCC8 scenario N over V, a *pseudo-solution* of N is a sequence $\mathcal{L} = \langle S_0, \ldots, S_k \rangle$ of assignments to $V_0, \ldots, V_k \subseteq V$ such that there exists a sequence of progressively weakened networks $\mathcal{N} = \langle N_0, \ldots, N_{k+1} \rangle$ which satisfy:

- $N_0 = N$, $N_{k+1} = \emptyset$, and $N_i = N \setminus (\bigcup_{j=0}^{i-1} N|_{V_j})$ $(1 \leq i \leq k+1)$;
- S_i is a partial solution of N_i satisfying the constraints in $N_i|_{V_i}$ for every $0 \leq i \leq k$.

Note that if $\mathcal{L}$ is a pseudo-solution, then every constraint in N is satisfied by at least one of the partial solutions in $\mathcal{L}$. However, a pseudo-solution is not necessarily a solution of N, and the scenario N might not even be consistent. Nevertheless, a pseudo-solution $\mathcal{L}$ allows us to retrieve the basic RCC8 relation between any two variables a and b from N, by determining the basic RCC8 relation between the corresponding objects for a and b defined by the first partial solution in $\mathcal{L}$ which involves both. In this paper, we are particularly interested in *rectangular* pseudo-solutions of RCC8 networks, which consist of partial solutions that assign axis-aligned rectangles to variables.

Constructing Pseudo-Solutions

In this section, we present the algorithm for representing any RCC8 scenario N by a rectangular pseudo-solution of N. While our method could either deal with an RCC8 scenario directly or with a scenario that is implicitly induced by a set of regions, we will mainly consider the latter, since we are mostly interested in the cases where the number of regions is too large to be explicitly represented as an RCC8 scenario.

Recall that, when restricted to rectangles, each basic RCC8 relation is the union of several PA representable RA relations as defined in last section. Therefore, we can incrementally build a PA network in a greedy fashion as follows. We consider the RCC8 constraints one at a time. For each RCC8 constraint, we select a corresponding RA relation which is compatible with the PA network being constructed. If such an RA relation exists, the corresponding PA constraints are added to the PA network. From this PA network, we can easily construct a partial rectangular solution of the RCC8 network using topological sort. We then weaken the RCC8 network by removing the RCC8 constraints between the variables that are included in this partial solution. By repeating this process several times for the progressively weakened RCC8 networks, all the RCC8 constraints in the scenario will be satisfied and we obtain a rectangular pseudo-solution of the RCC8 scenario. The details of this process are presented in Algorithms 1 and 2.

Algorithm 1: ExtractPseudoSol(N)

Input: N, an RCC8 scenario with variables V.
Output: $\mathcal{L}$, a pseudo-solution of N (initially empty).

while $N \neq \emptyset$ **do**
  $(S, N') \leftarrow$ ConstructPartialSol(N);
  $\mathcal{L}$.add(S);
  $N \leftarrow N'$;

Algorithm 2: ConstructPartialSol(N)

Input: N, an RCC8 network with variables V.
Output: A pair (S, N') where S is a rectangular partial solution S of N and N' is a correspondingly weakened network.

$P_s \leftarrow \emptyset$;
$V_s \leftarrow \emptyset$;
for *each variable* $v_0 \in V$ **do**
  v_0.*feasible* $\leftarrow$ *true*;
  $P'_s \leftarrow P_s$;
  for *each variable* $v_i \in V_s$ **do**
    if $\exists \mathsf{PA}_{v_0 R v_i}$ *consistent with* P'_s **then**
      P'_s.add($\mathsf{PA}_{v_0 R v_i}$);
    else
      v_0.*feasible* $\leftarrow$ *false*;
      break;
  if v_0.*feasible* **then**
    V_s.add(v_0);
    $P_s \leftarrow P'_s$;
$S \leftarrow$ solution(P_s);
$N' \leftarrow N \setminus N|_{V_s}$;

In particular, on Line 2, Algorithm 1 repeatedly calls Algorithm 2 to construct partial solutions of progressively weakened RCC8 networks. When all constraints have been removed from N, we know that any constraint from the original scenario is satisfied by some partial solution, and hence that $\mathcal{L}$ is a pseudo-solution of the original scenario. To construct a rectangular partial solution (Algorithm 2) of a (weakened) RCC8 network N, we repeat the following steps for each variable v_0:

- Line 4: For each basic RCC8 constraint $(v_0 R v_i)$ between v_0 and the variables V_s that have already been considered:
  - Line 5: choose a PA representation $\mathsf{PA}_{v_0 R v_i}$ for $(v_0 R v_i)$ that is consistent with the current PA network P'_s being built (P'_s is a copy of P_s);
  - Line 8: if no such choice is possible, then move to the next variable.
- Lines 9 and 10: If no inconsistencies have occurred, update the PA network P_s with P'_s to which all the chosen PA representations have been added.

After this process, we obtain a consistent PA network P_s corresponding to a subnetwork of the RCC8 network N. By using topological sort (van Beek 1992), we can obtain a solution of P_s, from which a rectangular partial solution S of the RCC8 network N can easily be constructed. On Line 13, we then weaken the network N to N' by removing all the constraints between variables in V_s. Note that these constraints are satisfied by S. In fact, it is easy to see that the following conclusion holds.

Proposition 1. *Let S_i be the partial solution that is constructed in the i-th iteration of the while loop in Algorithm 1 and let V_i be the corresponding set of variables. It holds that any constraint (uRv) from the original RCC8 network N is satisfied by the partial solution S_{i_0}, where i_0 is the smallest index for which $u, v \in V_{i_0}$.*

After each iteration of the while loop in Algorithm 1, at least one constraint is removed. Therefore the algorithm will terminate after at most $O(n^2)$ iterations, where $|V| = n$. A larger number of constraints would typically be removed in each iteration, hence we can expect the required number of iterations to be much smaller in practice. The number of operations taken by Algorithm 2 is polynomial in the number of variables, because we add PA constraints corresponding to at most $O(n^2)$ RCC constraints. By using the consistency checking algorithm by van Beek (1992), which is quadratic in the number of variables, the total number of the operations is bounded by $O(n^4)$.

Clustering

When the number of variables becomes very large, it would not be feasible to directly apply our algorithm to the complete RCC8 scenario, and we may not even be able to represent the corresponding constraint network. To address this, we propose to cluster the variables and apply Algorithm 1 to each of these clusters. Note that this is different from the use of clusters by Fogliaroni (2012), where the idea is to remove constraints between variables in different clusters which are in a clustering relation. In contrast, we use clusters to decompose the large scenario into smaller subnetworks, so that Algorithms 1 and 2 can be efficiently applied to compactly encode the constraints between the variables in each cluster.

When the RCC8 scenario is implicitly induced from a set of regions, we use the idea of Quadtree (Finkel and Bentley 1974) to obtain a suitable clustering. In particular, the space of the regions is first split into $N \times N$ grid cells of equal size, where N is called the initial grid size. A region is assigned to a grid cell if it has a common point with that cell. If the number of regions assigned to a single grid cell exceeds a given limit K, then we split that cell into four cells of equal size, and repeat the procedure until either all grid cells are associated with less than K regions or the maximum number of splits M has been arrived at. Note that there might be points that belong to more than K of the regions, in which case the resulting clusters will always contain more than K regions. Furthermore note that if two regions are connected, there is at least one cluster to which they both belong.

We then use Algorithm 1 to generate a sequence of rectangular pseudo-solutions for all the clusters. Moreover, assuming the clusters are ordered in some way, we only consider an RCC8 constraint (uRv) in the first cluster that contains both u and v, i.e. we remove this constraint from the RCC8 networks of all succeeding clusters.

Answering Queries

To retrieve the RCC8 relation between v_i and v_j, we first need to determine the first cluster that contains both v_i and v_j. In the pseudo-solution corresponding to this cluster, we then need to find the first partial solution that covers both v_i and v_j. If there are no clusters which contain both variables, it means that the RCC8 relation between them is $\mathbf{DC}$.

To allow for efficient query answering, we store the information about each variable v_i as follows.

- An array with the indices of the clusters that contain v_i, sorted in ascending order. We call this array the *cluster array* of v_i.
- With each entry in the cluster array of v_i, corresponding to a cluster $\mathcal{C}_k$, we associate an array with the indices of the partial solutions (for cluster $\mathcal{C}_k$) which contain a rectangle for v_i, sorted in ascending order. We call the array a *partial solution array* of v_i w.r.t. the cluster $\mathcal{C}_k$.

The total storage size will then be proportional to the total number of rectangles. To reduce the storage size, in the next section we will discuss how we can (i) reduce the average number of clusters in which a given variable appears and (ii) reduce the average number of rectangles that is constructed for a variable in a pseudo-solution. We now discuss in more detail how the proposed encoding can be used to answer questions efficiently.

The relevant cluster for a given pair of regions (v_i, v_j) can be found, using the corresponding cluster arrays A_i and A_j, in at most $O(|A_i| + |A_j|)$ steps. Let us write $|A_i| = a_i$. The average number of comparisons is given by:

$$O(\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{i\neq j}(a_i+a_j)) = O(\frac{1}{n}\sum_{i}a_i),$$
$$= O(\frac{1}{n}\sum_{k}|\mathcal{C}_k|),$$

where n is the number of clusters and $|\mathcal{C}_k|$ is the number of regions in cluster $\mathcal{C}_k$. In other words, the average number of comparisons is proportional to the total cardinality of the clusters.

After determining the index of the first common cluster, we need to determine the first partial solution in the corresponding pseudo-solution which specifies a rectangle for both variables. Let b_i be the number of partial solutions in the pseudo-solution corresponding to $\mathcal{C}_k$, which specify b_i rectangles for v_i. Similar as for the cluster index, we find that on average we need $O(\frac{1}{t}\sum_i b_i)$ comparisons to find the first partial solution which is common to two variables v_i and v_j, where t is the number of variables in the considered cluster. In other words, the average number of comparisons for determining the relevant partial solutions is proportional to the total number of specified rectangles in the considered pseudo-solution.

Implementation Details

In this section, we discuss some implementation details, and improvements of the main algorithm, which affect the overall performance.

For clustering variables, we restrict K, the (soft) limit of the number of regions in each cluster, to be 100. In practice, this limit should be chosen as large as possible to reduce the total cardinality of clusters and hence the total number of rectangles. For each dataset, we cluster the variables with several different values of the initial split size N, to see which value gives smaller total cardinality of clusters. Note that $N = 1$ is not always optimal. In fact, for some datasets used in our experiments, the optimal value of N would be larger values such as 11. For each of the clusters obtained from the optimal value of N, we generate a sequence of rectangular partial solutions.

Let us now consider the algorithm for generating partial solutions. First, note that some variables in V_s, in Algorithm 2, might not have constraints with any of the other variables in V_s, as these constraints might already have been satisfied by earlier partial solutions. In such a case, it is not necessary to include a rectangle for these variables in the new partial solution. Therefore, before determining the partial solution for V_s, we can remove these variables from V_s. This will always reduce the total number of rectangles without affecting the correctness of the algorithm. We call this operation **REMOVE**.

Furthermore, there are three critical steps in Algorithm 2 that affect the number of rectangles:

1. Line 2: How to choose v_0?
2. Line 4: How to choose v_i?
3. Line 5: How to choose a PA representation of $(v_0 R v_i)$?

In a naive implementation of the algorithm, which we will refer to as **NAIVE**, we simply consider a random ordering of the variables, and choose the first PA representation according to a static ordering.

However the constraints between a variable v and the variables for which a PA representation has already been chosen will affect the likelihood that a rectangle for v can be found in the current partial solution. For example, suppose v_0 has $\mathbf{EC}$ constraints with variables $v_1, \ldots, v_4$ which are pairwise $\mathbf{DC}$. If we first choose and add PA representations to the PA network for $v_1, \ldots, v_4$, it might be impossible to find a rectangle for v_0 such that all the $\mathbf{EC}$ constraints are satisfied. This is because the rectangle of v_0 should touch the rectangles of $v_1, \ldots, v_4$, but some specific relative positions of the disjoint rectangles of $v_1, \ldots, v_4$ will make it impossible to find such a rectangle. The case for $\mathbf{PO}$ constraints is similar. This suggests that we should order variables based on the type of constraints in which they are involved. For example, a good strategy seems to be to consider $\mathbf{NTPP}$ and $\mathbf{TPP}$ constraints before others, as the relative position of one rectangle involved in an $\mathbf{NTPP}$ or $\mathbf{TPP}$ constraint has a great influence on the relative position of the other. On the other hand, it is usually trivial to find a rectangle that satisfies a given $\mathbf{DC}$ constraint.

Based on this intuition, we propose the following improvement. For every variable, we count the number of times it is involved in each of the basic RCC8 relations. We then order the variables as follows. For variable v, let the vector $(n^v_{\mathbf{NTPP(i)}}, n^v_{\mathbf{TPP(i)}}, n^v_{\mathbf{PO}}, n^v_{\mathbf{EC}}, n^v_{\mathbf{DC}})$ contain the number of $\mathbf{NTPP/NTPPi}$, $\mathbf{TPP/TPPi}$, $\mathbf{PO}$, $\mathbf{EC}$, and $\mathbf{DC}$ constraints for v. To order variables, we use the lexicographic order between these vectors. In other words, we first order the variables according to the number of $\mathbf{NTPP/NTPPi}$. To break ties, we first consider the number of $\mathbf{TPP/TPPi}$, etc. We will refer to this improvement as **LABEL**.

We also consider the following alternative. With each variable v we associate a score s_v defined as:

$$s_v = 10n^v_{\mathbf{NTPP(i)}} + 5n^v_{\mathbf{TPP(i)}} + 2n^v_{\mathbf{PO}} + 1n^v_{\mathbf{EC}},$$

where the numbers in the formula were determined based on results of some random test sets. We can then order the variables according to these scores. This improvement will be referred to as **WEIGHT**.

Finally, consider the third point, i.e. how to choose a PA representation, for RCC8 relations that do not correspond to a unique PA representation. Recall that we are primarily interested in applications where the input is a geographical dataset. As a heuristic approach to choosing a PA representation, we consider the minimum bounding rectangles (MBRs) of the geometric representations of each region. In particular, note that a PA representation of an RA relation corresponds to some configurations of rectangles, and these rectangles have relative positions such as *left*, *right*, *up* or *down*. We then determine the relative position of the MBRs. When the MBRs are in the same relation as the regions, this can readily be determined; otherwise we look at the relative position of their centre points. The PA representations that have the same relative position as the MBRs will be considered first. We will refer to this improvement as **Type**.

Experimental Results

In the experiments, we use the Non-**DC** method (cf. the related work section) as our main baseline, since it leads to more compact representations than the spatial clustering index and MBR based approaches mentioned in the section on related work. Given a set of n regions, let $N^c = \{v_i R_{ij} v_j : R_{ij} \neq \mathbf{DC}, 1 \leq i < j \leq n\}$ be the set of non-**DC** constraints. Note that on average, each region intersects with $2|N^c|/n$ regions in the dataset. In the following, we will refer to $|N^c|/n$ as the Intersection Measurement Index (IMI), and we will compare the storage size of our method against this value. The storage size of our algorithm is measured by the average number of rectangles which are stored for each variable. Note that applying **Remove** will always yield better performance, and hence in all the cases we will only show the results of implementations of our algorithm with **Remove** applied.

In the following, we first compare the performance of different optimizations of our algorithm, which shows that **Label** performs best; then we compare the performance of **Label** and some of the state-of-the-art techniques, where **Label** outperforms them significantly in most cases; finally we have a look at the performance of **Label** to answer queries, which turns out to be much faster than direct computation from polygons.

Comparison of Optimizations

To compare the performance of the different variants of our algorithm, we have generated a number of synthetic datasets of convex polygons, with cardinalities of 30, 30, 50, 50, 100, and 100, and IMIs of 5, 9, 9, 14, 10, and 25. To generate a convex polygon in the dataset, we sample some elements from a set of points with integer coordinates, say $\{(i, j) : 0 \leq i, j \leq d\}$, and calculate the convex hull of the selected points. To sample the points, we first randomly select a "feed" point, which induces a Gaussian distribution on the integer points around it, and then we randomly sample the other points from this Gaussian. For these datasets, we did not cluster the variables as they are small in number.

Table 1 shows the average number of rectangles for each variant of our algorithm. We can see from the table that **Weight** and **Label** perform similarly, and substantially outperform **Naive**. In most cases, **Type** alone is not as effective as **Weight** and **Label**, while the combination of **Label/Weight** and **Type** usually leads to the best performance.

We also compared the implementations on five small real-world datasets containing habitat distribution regions, with cardinalities of 600, 610, 605, 611, and 604, and IMIs of 12, 30, 46, 52, and 74. These datasets were also used by Long et al. (2015). We clustered the variables by setting the limit size of a cluster as 100.

Table 2 shows the average number of rectangles for each implementation of our algorithm. In contrast to the results from Table 1, **Label** consistently outperforms **Weight** in this case, while surprisingly, **Type** leads to results that are even worse than **Naive**. Also, **Label+Type** and **Weight+Type** did not improve **Label** and **Weight**. Note that in this case many regions consist of multiple connected components, where the MBRs or centre points of MBRs seems less effective to reflect the correct relative positions and hence **Type** would be likely to order PA representations in a wrong way. Note that all these implementations outperform the Non-**DC** method in all cases, in the sense that the number of required rectangles is smaller than the number of relations that need to be stored by the Non-**DC** method, with the latter being equal to the IMI value. In the following experiments, we will use the implementation **Label**.

#var	IMI	N	W	L	T	W+T	L+T
30	5	6.20	2.57	2.60	3.40	2.47	2.67
30	9	5.90	2.67	2.00	2.00	2.00	2.00
50	9	8.12	3.08	3.58	4.40	3.24	2.66
50	14	6.92	3.56	3.54	3.62	3.34	3.38
100	10	15.32	4.46	4.47	7.51	4.97	4.17
100	25	15.44	5.23	5.05	8.23	4.84	4.78

Table 1: Comparison on synthetic datasets of the average number of rectangles needed for different implementations, where "N" stands for **Naive**, "W" for **Weight**, "L" for **Label**, and "T" for **Type**.

#var	IMI	N	W	L	T	W+T	L+T
600	12	10.76	6.01	5.95	12.67	7.71	7.11
610	30	17.98	10.64	10.17	20.79	11.99	11.95
605	46	19.91	12.29	12.15	24.68	14.67	14.65
611	52	30.04	18.35	17.51	36.48	21.88	21.48
604	74	35.30	23.79	23.63	40.61	27.64	27.07

Table 2: Comparison on small real-world datasets of the average number of rectangles needed for different implementations.

Comparison With Baseline Methods

To test the performance of our method on large real-world datasets, we have used four datasets about species distribution and habitat from the European Environment Agency (EEA)[1], as well as a dataset with county subdivisions of the USA[2], a dataset of school catchment areas in the USA[3], and the combination of the last two. The four datasets from EEA contain 5,322, 6,258, 10,061 and 11,613 regions (after removing duplicates), with an IMI of respectively 63.92, 61.83, 121.54, and 119.91. We will denote these four datasets by HEU, HMS, SEU and SMS. The dataset of county subdivisions, denoted by CS, contains 36,702 regions, with an IMI of 3.10. The dataset of school catchment areas, denoted by SC, contains 65,192 regions (after removing duplicates), with an IMI of 9.36. The combination of the last two (CS+SC) contains 101,894 regions resulting in an IMI of 7.11. As the number of variables in these datasets is very large, we will cluster the

[1] http://www.eea.europa.eu/

[2] http://www.census.gov/

[3] http://nces.ed.gov/surveys/sdds/sabs/

	HEU	HMS	SEU	SMS	CS	SC	CS+SC
LABEL	20.75	21.05	34.68	34.20	3.41	6.27	5.23
Non-**DC**	63.92	61.83	121.54	119.91	3.10	9.36	7.11
MBR-C	118.10	113.93	205.82	205.27	4.37	9.03	7.35
MBR-**DC**	74.22	69.94	127.32	122.79	4.50	9.04	7.41

Table 3: Comparison of storage sizes of LABEL, Non-**DC**, MBR-C, and MBR-**DC** for large real-world datasets.

variables by setting the limit size of a cluster as 100.

In addition to comparing our results against the Non-**DC** method, we also present results of the two improvements of the MBR-based approach, i.e. MBR-C and MBR-**DC** from Long et al. (2015). Note that in general, the Non-**DC** method is not guaranteed to outperform these two methods. Compared to the original MBR-based approach, MBR-C further removes any constraints such that the relation between the variables is the same as the relation between the corresponding MBRs. Based on the result of MBR-C, MBR-**DC** tests if the MBRs of the connected components in a region are all disjoint with those of another region, and it removes all constraints for such regions. Both of them need to store the MBRs of regions besides the constraints, and MBR-**DC** also needs to store the MBRs of connected components of the regions. Therefore, we measure the storage size of both methods by calculating the average number of stored MBRs and constraints for each variable.

It is worth noting that we did not include the prime subnetwork technique (Li et al. 2015) as a baseline here. The reason is that, to find out the relation between two variables in a prime subnetwork, sometimes we need to do qualitative reasoning on the whole network, which can be very inefficient when the number of variables becomes large. For example, an experiment on the datasets involved in Table 2 shows that even for prime subnetworks with about 600 variables, it takes 10^7ns to calculate the relation between two variables whose constraint has been characterized as redundant and removed in the prime subnetwork. Therefore, we do not consider this method as a competitive representation that supports efficient query answering.

Table 3 shows the results on these datasets. We can see that for all datasets apart from CS, LABEL generates pseudo-solutions with fewer rectangles than there are non-DC constraints, which is also smaller than the storage size required by MBR-C and MBR-**DC**. Note that for CS, the number of rectangles generated by LABEL is slightly larger than the number of non-DC constraints, which is what we may expect in cases where the IMI is small and the number of regions is large. On the other hand, when the IMI becomes larger the differences become more pronounced. For example, for each of the SEU and SMS datasets, the total number of non-**DC** constraints is about 1,300,000, while the total number of rectangles generated by LABEL is only about 370,000.

Note that the average number of rectangles generated by LABEL grows as the IMI grows. To analyse the relationship between IMI and the number of rectangles generated by LABEL, we performed an additional experiment on

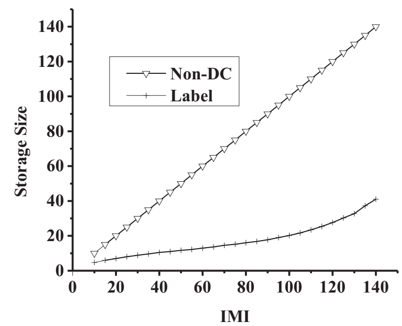


Figure 4: Illustration of the growth rate of LABEL when IMI increases.

synthetic data. For each IMI value from 10 to 140 with a step size of 5, we generated 10 datasets of 1000 convex polygons. When clustering, since the largest IMI is 140, there are many cases where more than 100 regions have a common point and need to be clustered into the same cluster. If we set the limit size of a cluster to be 100, some clusters have to be split many times which will make clustering less efficient. Therefore we set the limit size of a cluster as 120. Figure 4 illustrates the number of rectangles generated by LABEL, in relation to the IMI of the dataset. Note that each data point is the average over 10 datasets. We can see that until the IMI reaches about 80, the growth rate for the number of rectangles is sub-linear. However, as IMI reaches the cluster size limit, the growth rate increases, which is due to the fact that an increasing number of variables will then be included in several clusters.

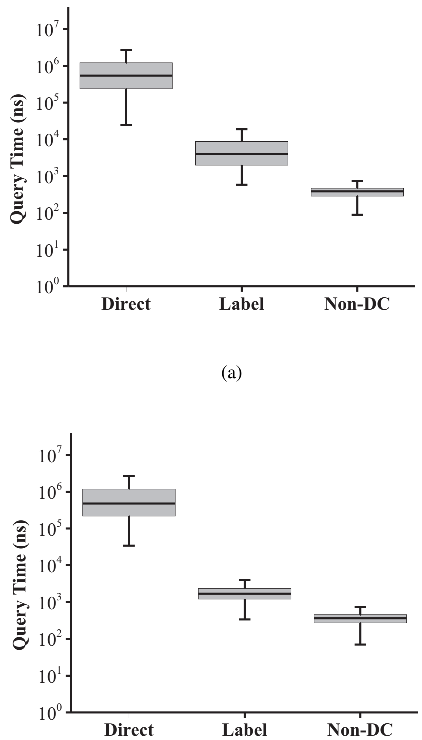
IMI (Non-**DC**)	24	34	44	54	64	74	84
LABEL	19.1	21.73	23.56	25.45	26.71	28.13	29.30

Table 4: Comparison of storage sizes of LABEL and Non-**DC** for networks generated using the BA model.

We also tested on some randomly generated RCC8 constraint networks by using the Barabási-Albert (BA) model which is proposed by Barabási and Albert (1999) and first exploited in QSR by Sioutis, Condotta, and Koubarakis (2015). In particular, for each IMI from 24 to 84, with a step size of 10, we extracted 10 complete basic RCC8 networks of 200 variables from the scale-free structured networks generated by the BA model (with preferential attachment of 2). The results are shown in Table 4. We can easily see that, although the average number of rectangles generated by LABEL grows when the IMI grows, it grows much slower, in accordance with our earlier results.

Answering Queries

Next we consider the computation time which is needed to determine the RCC8 relation between two given variables. This corresponds to the evaluation of queries such as “does species A live in an area where species B is also present” or “is neighbourhood X is the catchment area of



(a)

(b)

Figure 5: (a) Query times on SEU dataset. (b) Query times on CS+SC dataset.

school Y". The queries considered here are fundamental ones, and more complex queries can be answered by integrating our approach with other techniques, such as using the pseudo-solution representation as a compact back-end representation of the R-tree method in (Papadias et al. 1995). In the following, we will compare the performance of three methods: (i) determining the relation by comparing the geometric representation of the boundaries of the regions by using JTS[4] (Direct), (ii) using the pseudo-solution produced by the method **Label**, and (iii) using an RCC8 network without $\mathbf{DC}$ constraints (Non-$\mathbf{DC}$). We assume that all the information, including geometries of regions, MBRs, constraints, and rectangles are stored in memory. Specifically, the constraints are stored in a hash table indexed by the identifiers of variables, and the rectangles are stored as explained before. The experiment was conducted on a computer with Intel® Xeon™ E5-2450L 1.8 GHz CPU and 128 GB RAM.

Figure 5 presents the results of $10{,}000$ random queries for the following two datasets: SEU (which has the largest IMI) and CS+SC (which has the largest number of variables). The queries only involve pairs of variables whose MBRs intersect, and the $10{,}000$ pairs are chosen by randomly sampling in the set of all such pairs. This is motivated by the observation that when geometric information is available, it is easy to apply a pretest for intersection of the MBRs by all methods. For these two datasets, **Label** exhibits a clearly better performance than Direct and is reasonably efficient compared to Non-$\mathbf{DC}$, which simply needs to retrieve the constraint from a hash table. The median query time for **Label** is about 3,300ns for SEU and about $1{,}700$ns for CS+SC, while for Direct it is around $470{,}000$ns for both datasets and 500ns for Non-DC. In fact, in both datasets there are about 3000 queries for which the Direct method needs more than 10^6ns. Note that the fact that **Label** generates more rectangles for the SEU dataset than for the CS+SC dataset translates into a higher query time for the former dataset. Finally, we should note that when information is stored on disk rather than in memory, the performance of all methods would be affected.

[4]http://www.vividsolutions.com/jts/JTSHome.htm

Conclusion

In this paper, we proposed an encoding for large RCC8 scenarios. The main novelty of our approach is the use of pseudo-solutions, which correspond to sequences of partial solutions from which the correct RCC8 relation between any two variables can be derived. Such pseudo-solutions are easier to obtain than actual solutions, and because they are less constrained they allow for more compact representations. Our experimental results have shown that this approach is indeed effective in practice. For example, the pseudo-solution of an RCC8 network with 10,000 variables and 1,300,000 non-DC relations only required 35 rectangles per variable.

There are some cases where the representation might still be quadratic in the number of variables. For example, when all pairs of regions are $\mathbf{EC}$, the current algorithm would result in a representation with $O(n^2)$ number of rectangles. How to modify our approach to deal with these cases would then be one of our future work.

Besides its advantages and limitations, the idea of using pseudo-solutions for representing qualitative spatial information opens up several promising areas for future work. For example, while we have focused on RCC8, it seems likely that good results can be achieved in a similar way for other qualitative calculi. It would also be interesting to generalize our method to RCC8 networks that contain non-basic relations. While it would be difficult to handle arbitrary RCC8 relations, it seems that most of the non-basic RCC8 relations that are likely to be encountered in practical applications could be encoded by associating with every variable a nested pair of rectangles, denoting an upper and lower approximation of the unknown boundaries, as in the Egg-Yolk calculus (Cohn and Gotts 1996). For example, to encode the constraint $(u\{\mathbf{EC},\mathbf{PO}\}v)$, we can assign to u the rectangles r_1 and r_2 and to v the rectangle r_3 such that $r_1\mathbf{NTPP}r_2$, $r_1\mathbf{EC}r_3$, and $r_2\mathbf{PO}r_3$. This shows the potential of extending this method for indefinite information. Another example would be that we can extend the idea of using rectangles to intervals or higher-dimensional hyper-rectangles, which could possibly further improve the representation. Finally, it will be very interesting to investigate the possibility of using our concept of pseudo-solution in the inconsistency handling of qualitative constraint networks.

Acknowledgements

We thank the anonymous reviewers for their helpful suggestions. This work was supported by the ARC grants DP120104159 and FT0990811, and an ERC Starting Grant (637277). The work was completed during the visit of Zhiguo Long to Dr Steven Schockaert at Cardiff University, which was supported by the FEIT HDR International Experience Program of University of Technology Sydney.

References

Al-Salman, R. 2014. *Qualitative Spatial Query Processing : Towards Cognitive Geographic Information Systems*. Ph.D. Dissertation, Universität Bremen.

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

Balbiani, P.; Condotta, J.-F.; and Fariñas del Cerro, L. 1999. A new tractable subclass of the rectangle algebra. In *IJCAI-99*, 442–447.

Barabási, A.-L., and Albert, R. 1999. Emergence of scaling in random networks. *Science* 286(5439):509–512.

Cohn, A. G., and Gotts, N. M. 1996. The 'egg-yolk' representation of regions with indeterminate boundaries. *Geographic objects with indeterminate boundaries* 2:171–187.

Cohn, A. G., and Renz, J. 2008. Qualitative spatial representation and reasoning. In *Handbook of Knowledge Representation*. Elsevier, Amsterdam, Netherlands. 551–596.

Egenhofer, M. J., and Franzosa, R. D. 1991. Point-set topological spatial relations. *International Journal of Geographical Information System* 5(2):161–174.

Egenhofer, M. J., and Herring, J. 1991. Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical report, Department of Surveying Engineering, University of Maine.

Finkel, R. A., and Bentley, J. L. 1974. Quad trees a data structure for retrieval on composite keys. *Acta Informatica* 4(1):1–9.

Fogliaroni, P. 2012. *Qualitative spatial configuration queries – Towards next generation access methods for GIS*. Ph.D. Dissertation, University of Bremen.

Gerevini, A. 2005. Incremental qualitative temporal reasoning: Algorithms for the Point Algebra and the ORD-Horn class. *Artificial Intelligence* 166(1-2):37–80.

Goyal, R. K., and Egenhofer, M. J. 1997. The direction-relation matrix: A representation for directions relations between extended spatial objects. In *The Annual Assembly and the Summer Retreat of University Consortium for Geographic Information Systems Science*, 22–81.

Guesgen, H. W. 1989. Spatial reasoning based on Allen's temporal logic. Technical report, International Computer Science Institute, Berkeley, USA.

Ladkin, P. B., and Maddux, R. D. 1988. On binary constraint networks. Technical report, Kestrel Institute, Palo Alto, Calif.

Li, S., and Cohn, A. G. 2012. Reasoning with topological and directional spatial information. *Computational Intelligence* 28(4):579–616.

Li, S.; Long, Z.; Liu, W.; Duckham, M.; and Both, A. 2015. On redundant topological constraints. *Artificial Intelligence* 225:51–78.

Long, Z., and Li, S. 2015. On distributive subalgebras of qualitative spatial and temporal calculi. In *COSIT-2015*, 354–374.

Long, Z.; Duckham, M.; Li, S.; and Schockaert, S. 2015. Indexing large geographic datasets with compact qualitative representation. *International Journal of Geographical Information Science* (Available Online, DOI: 10.1080/13658816.2015.1104535).

Papadias, D.; Theodoridis, Y.; Sellis, T. K.; and Egenhofer, M. J. 1995. Topological relations in the world of minimum bounding rectangles: A study with R-trees. In *SIGMOD-95*, 92–103.

Randell, D. A.; Cui, Z.; and Cohn, A. G. 1992. A spatial logic based on regions and connection. In *KR-92*, 165–176.

Schockaert, S.; Smart, P. D.; Abdelmoty, A. I.; and Jones, C. B. 2008. Mining topological relations from the web. In *DEXA-2008*, 652–656.

Sioutis, M.; Condotta, J.-F.; and Koubarakis, M. 2015. An efficient approach for tackling large real world qualitative spatial networks. *International Journal on Artificial Intelligence Tools* (In press).

Smith, T. R., and Park, K. K. 1992. Algebraic approach to spatial reasoning. *International Journal of Geographical Information Systems* 6(3):177–192.

van Beek, P., and Cohen, R. 1990. Exact and approximate reasoning about temporal relations. *Computational Intelligence* 6(3):132–147.

van Beek, P. 1992. Reasoning about qualitative temporal information. *Artificial Intelligence* 58:297–326.

Vasardani, M.; Winter, S.; and Richter, K.-F. 2013. Locating place names from place descriptions. *International Journal of Geographical Information Science* 27(12):2509–2532.

Vilain, M. B., and Kautz, H. A. 1986. Constraint propagation algorithms for temporal reasoning. In *AAAI-86*, 377–382.

Vögele, T.; Schlieder, C.; and Visser, U. 2003. Intuitive modelling of place name regions for spatial information retrieval. In *Spatial Information Theory. Foundations of Geographic Information Science*, volume 2825. Springer, Berlin-Heidelberg, Germany. 239–252.