Efficient Path Consistency Algorithm for Large Qualitative Constraint Networks

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

Michael Sioutis
CRIL-CNRS UMR 8188
University of Artois
France
sioutis@cril.fr

Sanjiang Li
UTS-AMSS Joint Lab
QCIS, FEIT
University of Technology Sydney
Australia
Sanjiang.Li@uts.edu.au

Abstract
We propose a new algorithm called DPC+ to enforce partial path consistency (PPC) on qualitative constraint networks. PPC restricts path consistency (PC) to a triangulation of the underlying constraint graph of a network. As PC retains the sparseness of a constraint graph, it can make reasoning tasks such as consistency checking and minimal labelling of large qualitative constraint networks much easier to tackle than PC. For qualitative constraint networks defined over any distributive subalgebra of well-known spatio-temporal calculi, such as the Region Connection Calculus and the Interval Algebra, we show that DPC+ can achieve PPC very fast. Indeed, the algorithm enforces PPC on a qualitative constraint network by processing each triangle in a triangulation of its underlying constraint graph at most three times. Our experiments demonstrate significant improvements of DPC+ over the state-of-the-art PPC enforcing algorithm.

1 Introduction
One of the major concerns in AI is dealing with spatio-temporal information, which is involved in many AI applications, such as automatic data maintenance and visualization in Geographic Information Systems (e.g., [Wallgrén, 2012]), robotic navigation (e.g., [Wolter, 2008]), and computer-aided design (e.g., [Bhatt et al., 2009]). Qualitative Spatial and Temporal Reasoning (QSTR) is devoted to processing such kind of information correctly and efficiently. In particular, the techniques in QSTR focus on finding ways to efficiently solve several fundamental reasoning problems, such as the consistency problem, the minimal labelling problem, and the redundancy problem. The consistency problem asks if a constraint network has a solution. The minimal labelling problem requests the strongest implied relations in a constraint network, i.e., the implied relations comprising only tuples that participate in a solution of the network. The redundancy problem [Li et al., 2015] is a newly studied problem in QSTR that aims to simplify the representation of a constraint network by removing its redundant constraints, i.e., the constraints that can be entailed by the rest of the network. We refer to [Cohn and Renz, 2008] for more information regarding fundamental reasoning problems in QSTR.

Path consistency (PC) is an important local consistency condition for constraint networks. Given a constraint network \( \mathcal{N} \), PC can be established in \( \mathcal{N} \) by applying the path consistency algorithm (PC)\(^1\) in [Montanari, 1974; van Beek, 1989], which makes all pairs of variables path consistent with any third variable in \( \mathcal{N} \) by considering the completion of the underlying constraint graph of \( \mathcal{N} \). It has long been known that PC can decide the consistency of maximal tractable subalgebras of several very important qualitative spatial and temporal calculi. Recently, PC became more useful as it was shown to be able to solve the minimal labelling problem and decisively assist in solving the redundancy problem of any distributive subalgebra of several well-known calculi. A subalgebra is called distributive if (weak) composition distributes over non-empty intersections of its relations. The concept of distributive subalgebras was recently proposed in [Li et al., 2015] and further discussed in [Long and Li, 2015]; some previous related work identified certain subalgebras to be distributive as well (cf. [van Beek and Cohen, 1990]).

With the expansion of the internet, the age of Big Data has arrived. Without emphasis on scalability, the current techniques in QSTR have become less suitable to deal with the situation where tens of thousands or even millions of spatiotemporal objects are involved. For example, PC has a time complexity of \( O(n^3) \), which already makes it hardly scalable for large real-world datasets with even less than a few thousands of variables. In the past few years, research in QSTR has focused on trying to deal with this situation. In particular, Chmeiss and Condotta [2011] and Sioutis and Koubarakis [2012] adopted the partial path consistency algorithm (PPC) [Blied and Sam-Haroud, 1999], as a substitute for the original path consistency algorithm PC, for some fundamental reasoning tasks of certain qualitative spatial and temporal calculi. Given a constraint network \( \mathcal{N} \), PPC enforces partial path consistency (PPC) on \( \mathcal{N} \), which is PC restricted to a triangulation of the underlying constraint graph of \( \mathcal{N} \). As such, PPC can exploit the sparseness of a constraint graph, in contrast to PC, and therefore be more efficient than

\(^{1}\)In what follows, we use bold sans serif font to denote an algorithm name (e.g., the PC enforcing algorithm is denoted by PC).
PC, especially when dealing with large and sparsely structured qualitative constraint networks. Note that when the result of a triangulation for a given constraint network $\mathcal{N}$ is a complete graph, PPC will enforce PC on $\mathcal{N}$. Further, it was shown by Sioutis et al. [2015b] and Long and Li [2015] that PPC can have the same reasoning power as PC on the common edges between a triangulation and the completion of the underlying constraint graph of $\mathcal{N}$. In particular, if $\mathcal{N}$ is defined over some distributive subalgebra, the relations on these edges will be minimal after enforcing PPC w.r.t. a triangulation on $\mathcal{N}$, as would be the case with PC. This kind of hybrid restrictions on both the structure of the constraint graph and the allowed relations have also been discussed in the context of constraint satisfaction problems to identify tractable subclasses (see [Cohen et al., 2012]). PPC can also be helpful to sufficiently identify the same set of non-redundant constraints in $\mathcal{N}$ as PC, by only considering the relations on these edges.

In addition, there is a parallel research stream in the field of the Simple Temporal Problem (STP). Xu and Choueiry [2003] realized that the STP has convexity properties analogous to those of convex CSPs and, thus, implemented a particular PPC enforcing algorithm for the STP, much like PPC, which was originally implemented for convex CSPs [Bliek and Sam-Haroud, 1999]. Later, Planken et al. [2008] noticed that the algorithm of Xu and Choueiry is not very efficient, as it is sometimes quadratic in the number of triangles in a triangulation of the underlying constraint graph of a given STP instance, which is also the case with PPC for CSPs and qualitative constraint networks. Therefore, based on an algorithm developed by Dechter et al. [1991], viz., the direction path consistency algorithm (DPC), Planken et al. proposed a new PPC enforcing algorithm, called $P^3C$, which has a worst-case time complexity that is linear in the aforementioned number of triangles.

It is then natural to ask if there is a similar algorithm to $P^3C$ for qualitative spatial and temporal calculi, as it would be quite useful for solving the associated reasoning tasks more efficiently, especially when large and sparsely structured qualitative constraint networks were involved. Our contributions with respect to answering that question are as follows. We develop a similar algorithm to $P^3C$, called DPC+, and prove that it can correctly enforce PPC on a qualitative constraint network that is defined over some distributive subalgebra. DPC+ can achieve PPC (and PC when a complete graph is used as the result of a triangulation) by processing each triangle in a triangulation of the underlying constraint graph of a given qualitative constraint network no more than three times. As PPC (and PC) in general process each such triangle many more times than that, we show that DPC+ is more efficient than PPC (and PC) in theory. Our experimental results with both real-world and synthetic datasets also confirm in practice that DPC+ can be significantly more efficient than PPC (and PC).

The remainder of the paper is organized as follows. After introducing some related concepts and results in Sections 2 and 3 respectively, we present and analyse our new algorithm for achieving PPC in Section 4, and experimentally illustrate its efficiency in Section 5. Section 6 concludes the paper.
remains chordal. The order in which simplicial vertices of sequential subgraphs are successively removed is called a perfect elimination ordering. Suppose that \( (v_0, v_{n-1}, \ldots, v_1) \) is a perfect elimination ordering of \( G \), then we will denote by \( F_k \) the set \( \{v_j : \{v_j, v_k\} \in E \wedge j < k \} \). Note that the subgraph induced by \( F_k \) is a complete subgraph of \( G \).

Let \( M \) be a qualitative calculus. A subalgebra \( S \) of \( M \) contains all basic relations and a subset of non-basic relations in \( M \), and is closed under converse, weak composition, and intersection. The concept of a distributive subalgebra, first proposed in [Li et al., 2015] and further discussed in [Long and Li, 2015], turns out to be useful for developing efficient algorithms to accomplish reasoning tasks such as deciding the consistency or solving the minimal labelling problem of a given qualitative constraint network.

**Definition 2.** A subalgebra \( S \) of \( M \) is distributive if \( R \circ (S \cap T) = (R \circ S) \cap (R \circ T) \) and \( (S \cap T) \circ R = (S \circ R) \cap (T \circ R) \) for any \( R, S, T \in S \) with \( S \cap T \neq \emptyset \).

Throughout the paper, we consider the calculi of PA, IA, RCC5/8, CRA, and BA, and the term “distributive subalgebra” specifically refers to a distributive subalgebra of one of these calculi. When \( M \) is one of these calculi, it satisfies the following property [Dyalla et al., 2013; Düntsch, 2005].

\[ M \] is a relation algebra. (1)

The Peircean Law (also known as the Cycle Law) [Dyalla et al., 2013; Long and Li, 2015] holds for relation algebras.

**Fact 1.** For relations \( R, S, T \) of a relation algebra, the Peircean Law requires that

\[
(R \circ S) \cap T \neq \emptyset \Leftrightarrow (R^{-1} \circ T) \cap S \neq \emptyset
\]

\[
(R \circ S) \cap T \neq \emptyset \Leftrightarrow (T \circ S^{-1}) \cap R \neq \emptyset.
\]

Distributive subalgebras also have a useful property that is closely related to the well-known Helly’s Theorem [Danzer et al., 1963].

**Definition 3.** A subclass \( S \) of a qualitative calculus is called Helly if, for any finite \( n \) relations \( R_1, \ldots, R_n \in S \), we have

\[
\bigcap_{i=1}^{n} R_i \neq \emptyset \quad \text{iff} \quad (\forall 1 \leq i \neq j \leq n) R_i \cap R_j \neq \emptyset.
\]

**Theorem 2** ([Long and Li, 2015]). Suppose \( M \) is a qualitative calculus that is also a relation algebra. Let \( S \) be a subalgebra of \( M \). Then \( S \) is distributive iff it is Helly.

With this property, to check if a set of relations have a non-empty intersection, one only needs to check if the intersection of each pair of relations from that set is non-empty.

### 3 Properties of PC and PPC

Path consistency concerns all triples of variables. This could be an overkill for many reasoning tasks. The idea of enforcing path consistency on a triangulation of the constraint graph of a given CSP was first proposed by Bliik and Sam-Haroud [1999] through the partial path consistency algorithm (PPC). Later, Chmeiss and Condotta [2011] and Sioutis and Koubarakis [2012] adopted PPC for IA and RCC8 respectively, and Amaneddine et al. [2013] and Long and Li [2015] further extended this idea to all of the calculi discussed here.

**Definition 5.** A QCN \( N = (V, C) \) is partially path consistent (PPC) w.r.t. a graph \( G = (V, E) \) iff \( \forall \{v_i, v_j\}, \{v_k, v_j\} \in E \) we have that \( R_{ij} \subseteq R_{ik} \circ R_{kj} \).

Note that PPC is the same as PC when the graph \( G \) in the definition of PPC is complete. In this paper, every considered calculus also satisfies the following property:

Every atomic QCN over \( M \) that is PC is satisfiable. (2)

For a calculus with Properties (1) and (2), PPC has the same reasoning power as PC on the common edges between a triangulation and the completion of the constraint graph of a qualitative constraint network that is defined over a distributive subalgebra, in the following sense.

**Proposition 3** ([Sioutis et al., 2015b; Long and Li, 2015]). Let \( N = (V, C) \) be a QCN that is defined over a distributive subalgebra of a qualitative calculus that satisfies (1) and (2), and \( G = (V, E) \) a chordal graph such that \( G_N \subseteq G \). Then, enforcing PPC w.r.t. \( G \) on \( N \) decides the consistency of \( N \), and results in the same labelling of the relations that correspond to the edges of \( G \) as enforcing PC.

We say that \( N \) is minimal if for each constraint \( (x, y) \) in \( N \), \( R \) is the minimal (or strongest) relation between \( x \) and \( y \) that is entailed by \( N \), where \( (x, y) \) is entailed by \( N \) if every solution of \( N \) satisfies \( (x, y) \).

**Theorem 4** ([Long and Li, 2015; Li et al., 2015]). Let \( S \) be a distributive subalgebra of a qualitative calculus that satisfies (1) and (2). Then every path consistent QCN that is defined over \( S \) is minimal.

The last two results show that given a QCN that is defined over a distributive subalgebra, enforcing PPC on it is able to make the relations corresponding to the edges of a triangulation of its constraint graph become minimal.

PC and PPC are also useful with regard to the redundancy problem. A constraint \( (x, y) \) in a QCN \( N \) is redundant if \( N \setminus \{(x, y)\} \) entails \( (x, y) \).

**Definition 6.** A QCN \( N = (V, C) \) is all-different if \( \forall v_i \neq v_j \in V, N \) does not entail \( (v_i, v_j) \).

The following result regarding the redundancy problem was first shown for RCC5/8, but applies to any of the calculi considered here (cf. [Sioutis et al., 2015b, Appendix]).

**Theorem 5** ([Sioutis et al., 2015b]). Let \( N = (V, C) \) be a satisfiable all-different QCN that is defined over a distributive subalgebra satisfying (1) and (2), and \( G = (V, E) \) a chordal graph such that \( G_N \subseteq G \). Then, if \( N \) is PPC w.r.t. \( G \), a constraint \( (v_i, v_j) \) is non-redundant in the PC subnetwork of \( N \) if and only if we have that \( \{v_i, v_j\} \subseteq E \) and \( R_{ij} \notin \{R_{ik} \circ R_{kj} : \{v_i, v_k\}, \{v_k, v_j\} \in E \} \).
Algorithm 1: DPC(N, α)

Input: A QCN N = (V, C) with n variables, and an ordering α = (v_n, ..., v_1) of V.

Output: True or False, a graph G = (V, E), and an updated N.

1. G ← (V, E ← E(G_N));
2. for v_k from v_n to v_1 do
   3. F_k ← {v_i : (v_i, v_k) ∈ E ∧ s < k};
   4. foreach v_i, v_j ∈ F_k with i < j do
      5. if {v_i, v_j} ∉ E then
         6. E ← E ∪ {{v_i, v_j}};
         7. temp ← R_{ij} ∩ (R_{ik} ∩ R_{kj});
         8. if temp ⊆ R_{ij} then
            9. R_{ij} ← temp;
            10. R_{ij} ← temp^{-1};
      11. if R_{ij} = ∅ then
          12. return (False, G, N);
   13. return (True, G, N);

4.1 Directional Path Consistency

Directional path consistency is another important local consistency condition, which can be used for deciding the consistency of a QCN that is defined over a distributive subalgebra. It was first proposed in the context of the Simple Temporal Problem (STP, [Dechter et al., 1991]).

Definition 7. A QCN N = (V, C) is directionally path consistent (DPC) with respect to an ordering of its variables α = (v_1, ..., v_n) iff for all v_i, v_k, v_j ∈ V with i, j < k we have that R_{ij} ⊆ R_{ik} ∩ R_{kj}.

As noted, DPC is already sufficient to decide the consistency of a QCN that is defined over a distributive subalgebra.

Proposition 6 ([Sioutis et al., 2016]). Let N = (V, C) be a QCN that is defined over a distributive subalgebra of a qualitative calculus that satisfies (1) and (2). Then, if N is DPC with respect to an ordering of its variables and does not contain an empty relation, N is satisfiable.

DPC (Algorithm 1) achieves DPC by using the idea of variable elimination [Sioutis et al., 2016]. It iterates variables with respect to an ordering, and propagates the constrainedness of the constraints involving a variable v_k, to the constraints involving only subsequent variables in the ordering, with the update rule R_{ij} ← R_{ij} ∩ (R_{ik} ∩ R_{kj}). This is as if the variable v_k is “eliminated” from the QCN. As a by-product, DPC also triangulates the constraint graph of a QCN and produces a chordal graph G that has the ordering α as a perfect elimination ordering.

Theorem 7 ([Sioutis et al., 2016]). Let N = (V, C) be a QCN that is defined over a distributive subalgebra of a qualitative calculus satisfying (1) and (2), and α = (v_n, ..., v_1) an ordering of V. Then, DPC returns (True, G, N') if and only if N is satisfiable, where G is a chordal graph such that G_N ⊆ G and α is a perfect elimination ordering of it, and N' is the DPC w.r.t. α subnetwork of N.

4.2 The New Algorithm DPC+

Although enforcing DPC is efficient, it is not guaranteed to achieve PPC or PC and, thus, cannot solve the minimal labelling problem or assist in solving the redundancy problem of a given QCN. We hereby propose a new algorithm that achieves PPC (and PC with a simple modification) by building on the DPC enforcing algorithm DPC. Given a QCN N and an ordering α = (v_n, ..., v_1) of its variables, we first enforce DPC w.r.t. α on N using DPC. Then, we update relations by iterating the variables in reverse order. In particular, for a variable v_k (k is from 1 to n), we consider the set F_k of variables that are adjacent to v_k and preceded by v_k in α. The relation between each v_i ∈ F_k and v_k is updated with ∩_{v_j ∈ F_k} R_{ij} ∩ R_{kj}. The detailed steps are shown in Algorithm 2. We call this new algorithm DPC+.

The following theorem shows that DPC+ establishes PPC in a satisfiable QCN that is defined over a distributive subalgebra. Note that if we replace the graph G in line 1 with the complete graph of the same order, DPC+ will achieve PC.

Theorem 8. Let N = (V, C) be a QCN that is defined over a distributive subalgebra of a qualitative calculus that satisfies (1) and (2), and α = (v_n, ... , v_1) an ordering of V. Then, DPC+ returns (True, G, N') if and only if N is satisfiable, where G is a chordal graph such that G_N ⊆ G and α is a perfect elimination ordering of it, and N' is the PPC w.r.t. G subnetwork of N.
Proof. After calling DPC in line 1, \( N \) becomes DPC w.r.t. \( \alpha \) and we get a chordal graph \( G \) such that \( G_N \subseteq G \) and \( \alpha \) is a perfect elimination ordering of it. In what follows, we denote by \( N(0) \) the DPC network before applying the next steps of DPC+ and by \( N \) the updated network obtained afterwards.

Suppose that \( N_k \) is the partial network of the updated \( N \) restricted to variables \( \{v_k, v_{k-1}, \ldots, v_1\} \), i.e., the updated partial network after considering \( \{v_k, v_{k-1}, \ldots, v_1\} \) in the for loop in line 4 of DPC+. Note that \( N_n = N \). It suffices to show that \( N_k \) is PPC given that \( N_{k-1} \) is PPC. To this end, we only need to consider \( F_i \) and show that \( \forall v_j, v_j \subseteq F_k \), we have \( R_{ik} \neq \emptyset \), \( R_{ij} \subseteq R_{ik} \cup R_{jk} \), and \( R_{ik} \subseteq R_{ij} \cup R_{jk} \).

To simplify our proof, we first adjust the updating rule in line 7 of DPC+ from \( R_{ik} \leftarrow \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk} \) to \( R_{ik} \leftarrow \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk} \), where \( R_{jk} \) is the relation between \( v_j \) and \( v_k \) in the original DPC network, viz., \( N(0) \). We denote the adjusted algorithm by DPC+. We will first prove that the conclusion holds for DPC+.

We first show that \( R_{ik} \neq \emptyset \) for all \( v_k \subseteq F_k \). Note that \( \forall v_j, v_j \subseteq F_k \), by DPC of \( N(0) \), we have \( R_{jk} \subseteq R_{ij} \cup R_{jk} \). By DPC of \( N_{k-1} \), we have \( R_{ij} \subseteq R_{ij} \cup R_{jk} \). Therefore, \( \emptyset \neq R_{ij} \subseteq R_{ij} \cup R_{jk} \). By the Peirce Law, we have \( (R_{ij} \cup R_{jk}) \cap (R_{ij} \cup R_{jk}) \neq \emptyset \).

According to the Helly property of distributive subalgebras, we have \( R_{ik} = \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk} \). No \( \emptyset \neq R_{ij} \subseteq R_{ij} \cup R_{jk} \).

Next, we show that \( R_{ij} \subseteq R_{ik} \cup R_{jk} \). Note that as \( R_{jk} = \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk} \), we have

\[
R_{ik} \cup R_{jk} = \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk} \overset{\text{(3)}}{=} R_{ij} \cup \bigcap_{v_j \subseteq F_k} R_{jk} = R_{ij} \cup R_{jk}.
\]

By distribution of weak composition, we have

\[
R_{ik} \cup R_{jk} = \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk} \overset{\text{(3)}}{=} R_{ij} \cup R_{jk} = (\bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk}) \cap (\bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk}) = R_{ij} \cup R_{jk}.
\]

As \( N(0) \) is DPC, we have \( R_{ij} \cup R_{jk} \subseteq R_{ij} \cup R_{jk} \subseteq R_{ij} \cup R_{jk} \). and as \( N_{k-1} \) is PPC, we have \( R_{ij} \subseteq R_{ij} \cup R_{jk} \). Therefore \( R_{ij} \subseteq R_{ij} \cup R_{jk} \). Since \( v_j \) is arbitrary in \( F_k \) and \( R_{jk} \subseteq R_{jk} \), we also have

\[
R_{ik} = \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk} = \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk}.
\]

Therefore, under the updating rule of DPC+, we have that \( N_k \) is PPC if \( N_{k-1} \) is PPC and, thus, DPC+ enforces PPC on \( N \). Regarding DPC+, its updating rule will update the first \( R_{ik} \) by using relations in \( N_{k-1} \) and \( N(0) \), and then update the following \( R_{jk} \) by using the updated \( R_{ik} \), and so on. By induction, we can prove that each updated \( R_{ik} \) is stronger than \( R_{ik}^{(0)} \) and weaker than \( \bigcap_{v_j \subseteq F_k} R_{ij} \cup R_{jk} \), where \( R_{jk} \) is obtained by DPC+. By (3), the relations obtained by DPC+ are the same as those obtained by DPC+. As DPC+ enforces PPC on \( N \), we have that DPC+ also enforces PPC on \( N \).

\( \square \)

4.3 Analysis of the DPC+ Algorithm

As an adaptation of the \( P^2 C \) algorithm in [Planken et al., 2008] to qualitative spatial and temporal calculi, the DPC+ algorithm only needs to update each triangle in a graph at most three times. This means that the time complexity of DPC+ is linear in the number of triangles \( t \) in the graph, if we assume that \( t \) dominates the number of vertices and edges.

**Theorem 9.** Let \( N = (V, C) \) be a QCN, and \( \alpha = (v_n, \ldots, v_1) \) an ordering of \( V \). Then, DPC+ returns \( (\text{True}, G, N') \) in \( \Theta(t + |V| + |E|) \) time when \( N \) is satisfiable, where \( G = (V, E) \) is a chordal graph such that \( G_N \subseteq G \), and \( t \) is the number of triangles in \( G \).

**Proof.** The DPC algorithm considers each triangle in \( G \) exactly once. For each \( \{v_i, v_k\} \in E \) such that \( i < k \), lines 6–8 in DPC+ will consider all the triangles involving \( v_i \) and \( v_k \) once. Therefore, each triangle \( \{v_i, v_j, v_k\} \) such that \( i, j < k \) and \( \{v_i, v_k\}, \{v_j, v_k\} \in E \) will be considered at most twice and, as such, by iterating \( v_k \) from \( v_i \) to \( v_n \), every triangle in the graph will be considered at most twice. Note that as DPC+ needs to scan through the vertices and edges, DPC+ runs in \( \Theta(t + |V| + |E|) \) time when \( N \) is satisfiable.

\( \square \)

In terms of the maximum vertex degree \( \Delta \) of \( G \), with the above analysis, it is easy to see that DPC+ has a time complexity of \( O(n \Delta^2) \), where \( n \) is the number of variables.

While DPC+ only needs to check each triangle in a graph at most three times, the state-of-the-art PPC enforcing algorithm PPC usually requires to check each such triangle many more times than that (e.g., as many times as \( 3|B| \)). This is also the case with the PC enforcing algorithm PC, since when a complete graph is used as the result of a triangulation, PPC falls back to PC. Therefore, we expect DPC+ to be more efficient than PPC and PC. Indeed, as shown in the following section, for large and sparsely structured networks, the advantage of DPC+ over PPC (and PC) is very significant.

5 Experimental Results

We evaluate the performance of our implementation of the DPC+ algorithm, against an implementation of the state-of-the-art PPC enforcing algorithm (PPC), for QCNs that are defined over a distributive subalgebra. We also employ an implementation of the state-of-the-art DPC enforcing algorithm (DPC) to pinpoint the overhead that DPC+ adds to DPC.

**Technical Specifications.** The experimentation was carried out on a computer with an Intel Core i7-2820QM processor with a 2.30 GHz frequency per CPU core, 8 GB of RAM, and the Trusty Tahr x86.64 OS. All algorithms were coded in Python and run with PyPy 2.2.1 [PyPy Team, 2015], which implements Python 2.7. Only one CPU core was used.
Datasets and Measures. We considered random RCC8 networks generated by the Ba(n, m) model [Barabasi and Albert, 1999], the use of which in qualitative constraint-based spatial and temporal reasoning is well motivated in [Sioutis et al., 2015a], and real-world RCC8 datasets that have been recently used in [Sioutis et al., 2015b; Li et al., 2015]. In particular, we used the Ba(n, m) model to create random scale-free graphs as the constraint graphs of the RCC8 QCNs. We considered 10 satisfiable RCC8 networks of model Ba(n, m) for each order 1000 ≤ n ≤ 10000 of their constraint graphs with a 1000-vertex step and a preferential attachment value of m = 2. The edges of these graphs were labelled with relations from the maximal distributive subclass D8 of RCC8 [Li et al., 2015]. Regarding real-world RCC8 datasets, we employed the ones recently used in [Sioutis et al., 2015b; Li et al., 2015], viz., nuts (nomenclature of territorial units)2 with 2235/3176 variables/constraints (by constraints we mean non-universal relations), adm1 (administrative geography of Great Britain) [Goodwin et al., 2008] with 11,762/44,832 variables/constraints, gadm1 (German administrative units)2 with 42,749/159,600 variables/constraints, gadm2 (the world’s administrative areas) [GeoVocab, 2012] with 276,729/589,573 variables/constraints, adm2 (the administrative geography of Greece)2 with 1,732,999/526,270 variables/constraints, fprints (geographic “footprints” in the South West area of the UK) [Li et al., 2015] with 3470/446,847 variables/constraints, and stareas (statistical areas in Rwanda) [Li et al., 2015] with 1,562,10/101 variables/constraints. These datasets are satisfiable. Each dataset comprises only relations that are contained in one of the maximal distributive subclasses of RCC8 [Li et al., 2015].

The maximum cardinality search algorithm [Tarjan and Yannakakis, 1984] was used to obtain a variable elimination ordering for DPC and DPC+, and a triangulation of the constraint graph of a given QCN for PPC as described in [Sioutis et al., 2015a]. We note that in our case any variable elimination ordering would be adequate for the evaluation to follow, as it would affect all involved algorithms proportionally and would not qualitatively distort the obtained result.

Our experimentation involves the following two measures. The first measure considers the number of constraint checks performed by a local consistency enforcing algorithm implementation. Given a QCN N = (V, C) and vi, vj, vk, vj ∈ V, a constraint check is performed when we compute relation r = Rij ∩ (Rik ∩ Rkj) and check if r ⊆ Rij, so that we can propagate its constrainedness. Weak compositions that yield relation ∗ are disregarded. The second measure concerns the CPU time and is strongly correlated with the first one, as the runtime of these implementations relies heavily on the number of constraint checks performed.

Results. The experimental results for random scale-free RCC8 networks are summarized in Table 1, where a fraction x/y denotes that an approach required x seconds of CPU time and performed y constraint checks on average per dataset of networks during its operation. It is clear that DPC+ performs significantly fewer constraint checks and is much faster than PPC for all considered networks, while retaining the

<table>
<thead>
<tr>
<th>n</th>
<th>DPC</th>
<th>DPC+</th>
<th>PPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>2000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>3000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>4000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>5000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>6000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>7000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>8000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>9000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
<tr>
<td>10000</td>
<td>2068</td>
<td>2068</td>
<td>2068</td>
</tr>
</tbody>
</table>

Table 1: Evaluation with random scale-free RCC8 networks.

<table>
<thead>
<tr>
<th>network</th>
<th>DPC</th>
<th>DPC+</th>
<th>PPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>nuts</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>adm1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>gadm1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>gadm2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>adm2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>fprints</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>stareas</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 2: Evaluation with real-world RCC8 datasets.

good time complexity characteristics of DPC. In particular, DPC+ enforces PPC on a random scale-free RCC8 network of model Ba(n = 10000, m = 2) in around 2 sec, when PPC requires approximately 8 times more time than that for the same task. Regarding real-world RCC8 datasets, the experimental results are summarized in Table 2, where a fraction x/y has the same meaning as before. Again, we can see that DPC+ significantly outperforms PPC with regard to both the CPU time required and the number of constraint checks performed for all datasets (with the exception of nuts, whose constraint graph is almost a tree and, thus, neither of the algorithms is challenged enough), while adding only limited overhead to the performance of DPC. As illustration, DPC+ enforces PPC on the largest of the datasets (adm2) in 8.97 sec, when PPC requires over 44 times more time than that, viz., a total of 398.63 sec, for the same task. In general, we noted much more inference occurring with real-world datasets than with random networks.

6 Conclusion

We proposed a new algorithm, called DPC+, that enforces partial path consistency (PPC) on a qualitative constraint network w.r.t. a triangulation of its underlying constraint graph. In particular, we showed that DPC+ can correctly establish PPC in a qualitative constraint network that is defined over any distributive subalgebra of well-known spatio-temporal calculi, such as the Region Connection Calculus and the Interval Algebra. We also showed that DPC+ only needs to process the triangles in a given graph at most three times, and is therefore much more efficient than the state-of-the-art PPC enforcing algorithm; experimental results with both real-world and synthetic datasets confirm this.
Acknowledgements

We thank the anonymous reviewers for their helpful suggestions. This work was supported by the ARC grants DP120104159 and FT0990811 and a PhD grant from the University of Artois and the Nord–Pas-de-Calais region.

References


