Incrementally Building Partially Path Consistent Qualitative Constraint Networks[†]

Michael Sioutis and Jean-François Condotta

Université Lille-Nord de France, Artois, CRIL-CNRS UMR 8188
Lens, France
{sioutis,condotta}@cril.fr

Abstract. The Interval Algebra (IA) and a fragment of the Region Connection Calculus (RCC), namely, RCC-8, are the dominant Artificial Intelligence approaches for representing and reasoning about qualitative temporal and topological relations respectively. In this framework, one of the main tasks is to compute the path consistency of a given Qualitative Constraint Network (QCN). We concentrate on the partial path consistency checking problem problem of a QCN, i.e., the path consistency enforced on an underlying chordal constraint graph of the QCN, and propose an algorithm for maintaining or enforcing partial path consistency for growing constraint networks, i.e., networks that grow with new temporal or spatial entities over time. We evaluate our algorithm experimentally with QCNs of IA and RCC-8 and obtain impressive results.

Keywords: qualitative constraint language, chordal graph, triangulation, qualitative reasoning, BA model, partial path consistency

1 Introduction

Spatial and temporal reasoning is a major field of study in Artificial Intelligence; particularly in Knowledge Representation. This field is essential for a plethora of areas and domains that include, but are not limited to, ambient intelligence, dynamic GIS, cognitive robotics, and spatiotemporal design [7]. The Interval Algebra (IA) [1] and a fragment of the Region Connection Calculus [16], namely, RCC-8, are the dominant Artificial Intelligence approaches for representing and reasoning about qualitative temporal and topological relations respectively.

The state-of-the-art techniques to enforce *partial path consistency* [8] on a set of IA or RCC-8 relations consider a fixed size constraint network to represent and reason with the relations. However, it may be the case that temporal intervals (the case for IA) or regions (the case for RCC-8) are not known a priori, but arrive continuously within different fragments of time. This is a real problem and has not been addressed before in literature. The term *"incremental"* has been used to describe the problem of maintaining or enforcing partial path consistency for a fixed size network when new constraints among existing nodes

[†]This work was funded by a PhD grant from Université d'Artois and region Nord-Pas-de-Calais.

are added or existing constraints are tightened. This approach is well described in the work of Gerevini [10] for qualitative temporal reasoning (where complete underlying constraint graphs are considered and, thus, partial path consistency is identical to path consistency [8, chapt. 6]) and the work of Planken et al. for the Simple Temporal Problem (STP) [15], and differs from our approach in that we consider extensions of a given network with new temporal or spatial entities. In a recent theoretical work, Huang showed that IA and RCC-8 have *canonical solutions* [13], i.e., path consistent IA or RCC-8 networks with relations from some maximal tractable subset of their signatures can be extended arbitrarily with the addition of new temporal or spatial entities respectively. In a more practical view, until recently state-of-the-art techniques made use of a matrix to represent a constraint network [20]. Growing a constraint network represented by an adjacency matrix requires $O(|V|^2)$ for every variable addition.

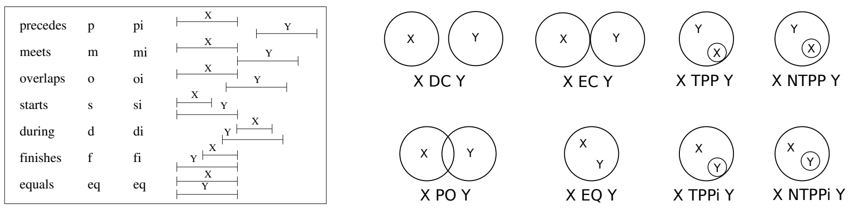
In this paper, we concentrate on the problem of maintaining or enforcing partial path consistency for growing constraint networks and make the following contributions: (i) we present an algorithm that maintains or enforces partial path consistency for an initial partially path consistent constraint network augmented by a new set of temporal or spatial entities and their accompanying constraints, (ii) we implement our algorithm making use of chordal graphs and a hash table based adjacency list to represent and reason with the QCNs as described in [20], (iii) we evaluate our algorithm experimentally with random and real QCNs of IA and RCC-8 and obtain quite interesting results.

The paper is organized as follows. Section 2 introduces the theoretical background of our work. In Section 3 we present our algorithm that maintains or enforces partial path consistency for an initial partially path consistent QCN augmented by a new set of temporal or spatial entities and their accompanying constraints. In Section 4 we use large QCNs of IA and RCC-8 to experimentally compare our algorithm with the state-of-the-art, one-shot partial path consistency algorithm that considers the whole network size all at once, and, finally, in Section 5 we conclude and give directions for future work.

2 Preliminaries

In this section we formally introduce the IA and RCC-8 constraint languages and partial path consistency, and chordal graphs along with triangulation.

The IA and RCC-8 constraint languages. A (binary) qualitative temporal or spatial constraint language [18] is based on a finite set B of *jointly exhaustive and pairwise disjoint* (JEPD) relations defined on a domain D, called the set of base relations. The set of base relations B of a particular qualitative constraint language can be used to represent definite knowledge between any two entities with respect to the given level of granularity. B contains the identity relation Id, and is closed under the converse operation ($^{-1}$). Indefinite knowledge can be specified by unions of possible base relations, and is represented by the set containing them. Hence, 2^{B} will represent the set of relations. 2^{B} is equipped



(a) The base relations of IA (b) The base relations of RCC-8

Fig. 1: IA and RCC-8 constraint languages

with the usual set-theoretic operations (union and intersection), the converse operation, and the weak composition operation. The converse of a relation is the union of the converses of its base relations. The weak composition $\diamond$ of two relations s and t for a set of base relations B is defined as the strongest relation $r \in 2^{\mathsf{B}}$ which contains $s \circ t$, or formally, $s \diamond t = \{b \in \mathsf{B} \mid b \cap (s \circ t) \neq \emptyset\}$, where $s \circ t = \{(x, y) \mid \exists z : (x, z) \in s \wedge (z, y) \in t\}$ is the relational composition.

The set of base relations of IA [1] is the set $\{eq, p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$. These thirteen relations represent the possible relations between *time intervals*, as depicted in Figure 1a. The set of base relations of RCC-8 [16] is the set $\{dc, ec, po, tpp, ntpp, tppi, ntppi, eq\}$. These eight relations represent the binary topological relations between *regions* that are non-empty regular subsets of some topological space, as depicted in Figure 1b (for the $2D$ case). IA and RCC-8 networks are qualitative constraint networks (QCNs), with relation eq being the identity relation in both cases.

Definition 1. *A RCC-8, or IA, network is a pair $\mathcal{N} = (V, C)$ where V is a non empty finite set of variables and C is a mapping that associates a relation $C(v, v') \in 2^{\mathsf{B}}$ to each pair (v, v') of $V \times V$. C is such that $C(v, v) \subseteq \{eq\}$ and $C(v, v') = (C(v', v))^{-1}$.*

Note that we always regard a QCN as a complete network. The underlying (constraint) graph of a QCN $\mathcal{N} = (V, C)$ is a graph $G = (V, E)$, for which we have that $(v, v') \in E$ iff $C(v, v') \neq \mathsf{B}$. Given two QCNs $\mathcal{N} = (V, C)$ and $\mathcal{N}' = (V', C')$, and their respective underlying graphs $G = (V, E)$ and $G' = (V', E')$ where $\forall (v, u) \in E'$ we have that $v \in V' \setminus V$ or $u \in V' \setminus V$, $\mathcal{N} \uplus \mathcal{N}'$ denotes the QCN $\mathcal{N}'' = (V'', C'')$, where $V'' = V \cup V'$, $C''(v, v') = \mathsf{B}$ for all $(v, v') \in (V \setminus V') \times (V' \setminus V)$, $C''(v, v') = C(v, v')$ for all $(v, v') \in (V \times V)$, and $C''(v, v') = C'(v, v')$ for all $(v, v') \in (V' \setminus V) \times V'$. The underlying graph of $\mathcal{N} \uplus \mathcal{N}'$ is graph $G \cup G' = (V \cup V', E \cup E')$. In what follows, $C(v_i, v_j)$ will be also denoted by C_{ij}. Checking the consistency of a QCN of IA or RCC-8 is $\mathcal{NP}$-complete in general [14, 19]. However, there exist large maximal tractable subclasses of IA and RCC-8 for which consistency checking can be done in polynomial time, $O(n^3)$ in particular, with a path consistency algorithm. These maximal tractable subclasses are the classes $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$ for RCC-8 [17] and $\mathcal{H}_{\mathsf{IA}}$ for IA [14]. When path consistency is enforced on the underlying constraint graph of an input QCN, we refer to it as *partial path consistency*. In our case, and throughout this paper, we enforce path consistency on the underlying *chordal* constraint graph of a given QCN, thus, whenever we use the term partial path consistency we implicitly consider underlying chordal constraint graphs. Partial path consistency was originally introduced for finite domain CSPs in [8] and it was most recently used in the case of IA and RCC-8 networks in [9] and [21] respectively.

Chordal graphs and Triangulation. We begin by introducing the definition of a chordal graph. More results regarding chordal graphs, and graph theory in general, can be found in [11].

Definition 2 ([11]). *Let $G = (V, E)$ be an undirected graph. G is* chordal *or* triangulated *if every cycle of length greater than* 3 *has a chord, which is an edge connecting two non-adjacent nodes of the cycle.*

Chordality checking can be done in (linear) $O(|V| + |E|)$ time for a given graph $G = (V, E)$ with the maximum cardinality search algorithm which also constructs an elimination ordering α as a byproduct [4]. If a graph is not chordal, it can be made so by the addition of a set of new edges, called *fill edges*. This process is usually called *triangulation* of a given graph $G = (V, E)$ and can run as fast as in $O(|V|+(|E \bigcup F(\alpha)|))$ time, where $F(\alpha)$ is the set of fill edges that result by following the elimination ordering α, eliminating the nodes one by one, and connecting all nodes in the neighborhood of each eliminated node, thus, making it simplicial in the elimination graph. If the graph is already chordal, following the elimination ordering α means that no fill edges are added, i.e., α is actually a *perfect elimination ordering* [11]. In a QCN fill edges correspond to universal relations, i.e., non-restrictive relations that contain all base relations (hence, the universal relation is equivalent to B). Chordal graphs become relevant in the context of qualitative reasoning due to the following result obtained in [2, 21] that states that partial path consistency is equivalent to path consistency in terms of consistency checking of tractable QCNs:

Proposition 1 ([2, 21]). *For a given* RCC-*8, or* IA*, network $\mathcal{N} = (V, C)$ with relations from the maximal tractable subclasses $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$, or $\mathcal{H}_{\mathsf{IA}}$, respectively, and for $G = (V, E)$ its underlying chordal graph, if $\forall(i, j), (i, k), (j, k) \in E$ we have that $C_{ij} \subseteq C_{ik} \diamond C_{kj}$, then $\mathcal{N}$ is consistent.*

In general, partial path consistency can be significantly faster than path consistency as the latter considers much more triangles of relations for a given QCN [9, 21]. (A chordal graph has less edges than a complete graph in general.)

3 The iPPC+ algorithm

In this section we present a new algorithm, viz., iPPC+, that enforces partial path consistency incrementally, together with an auxiliary algorithm, viz., GiPPC, that uses iPPC+ to simulate the construction of a QCN of n entities. Symbol + is only used to differentiate iPPC+ from the iPPC algorithm for the STP of Planken et al. [15], as we do not consider subsequent edge tightenings within a fixed size QCN, but rather extensions of a given QCN with new temporal or spatial entities accompanied by new sets of constraints. iPPC+ is structurally close to the one-shot partial path consistency algorithm presented in [9] and in [21] for IA and RCC-8 respectively. In what follows, we will refer to the one-shot partial path consistency algorithm simply as PPC.

```
Function iPPC+($\mathcal{N} \uplus \mathcal{N}'$, $G$, $G'$)
in     : A QCN $\mathcal{N} \uplus \mathcal{N}' = (V'', C'')$, and two chordal graphs $G = (V, E)$ and
         $G' = (V', E')$.
output : False if network $\mathcal{N} \uplus \mathcal{N}'$ results in a trivial inconsistency (contains the
         empty relation), True if the modified network $\mathcal{N} \uplus \mathcal{N}'$ is partially path
         consistent.
1  begin
2      $Q \leftarrow \{(i,j) \mid (i,j) \in E'\}$;
3      while $Q \neq \emptyset$ do
4          $(i,j) \leftarrow Q.pop()$;
5          foreach $k$ such that $(i,k),(k,j) \in E \cup E'$ do
6              $t \leftarrow C''_{ik} \cap (C''_{ij} \diamond C''_{jk})$;
7              if $t \neq C''_{ik}$ then
8                  if $t = \emptyset$ then return False;
9                  $C''_{ik} \leftarrow t$; $C''_{ki} \leftarrow t^{-1}$;
10                 $Q \leftarrow Q \cup \{(i,k)\}$;
11             $t \leftarrow C''_{kj} \cap (C''_{ki} \diamond C''_{ij})$;
12             if $t \neq C''_{kj}$ then
13                 if $t = \emptyset$ then return False;
14                 $C''_{kj} \leftarrow t$; $C''_{jk} \leftarrow t^{-1}$;
15                 $Q \leftarrow Q \cup \{(k,j)\}$;
16     return True;
```

iPPC+ receives as input a QCN $\mathcal{N} \uplus \mathcal{N}' = (V'', C'')$, where $\mathcal{N} = (V, C)$ is the initial partially path consistent QCN augmented by a new QCN $\mathcal{N}' = (V', C')$, and $G = (V, E)$ and $G' = (V', E')$ are their respective underlying chordal graphs where $\forall (v, u) \in E'$ we have that $v \in V' \setminus V$ or $u \in V' \setminus V$. The output of algorithm iPPC+ is False if network $\mathcal{N} \uplus \mathcal{N}'$ results in a trivial inconsistency (contains the empty relation), and True if the modified network $\mathcal{N} \uplus \mathcal{N}'$ is partially path consistent and not trivially inconsistent. The queue data structure is instatiated by the set of edges E' (line 2), i.e., the set of edges corresponding to the underlying graph of the new QCN $\mathcal{N}'$. Path consistency is then realised by iteratively performing the following operation until a fixed point $\overline{C''}$ is reached: $\forall i, j, k$ do $C''_{ij} \leftarrow C''_{ij} \cap (C''_{ik} \diamond C''_{kj})$, where edges $(i,k),(k,j) \in E \cup E'$ (line 5). Within the set of edges E' we implicitly consider a small number of edges that maintain chordality of the underlying constraint graph $G \cup G'$ of QCN $\mathcal{N} \uplus \mathcal{N}'$ (as it is possible that E' introduces cycles). These edges can be found before each appliance of the iPPC+ algorithm in time at most linear in the number of vertices [5]. In this paper, we always first construct the graph consisting of all temporal or spatial entities and constraint edges that will be added and triangulate it once, as it is also done in [15, chap. 5], thus reserving incrementally maintaining chordality for future work. The aforementioned path consistency operation will result in a partially path consistent network $\mathcal{N}'$, and a (possibly) modified partially path consistent network $\mathcal{N}$ after constraint tightenings that might occur. Since $G \cup G'$ is a chordal graph, it follows that QCN $\mathcal{N} \uplus \mathcal{N}'$ is partially path consistent with respect to its underlying chordal graph $G \cup G'$, and, thus, due to Proposition 1 we can assert the following theorem:

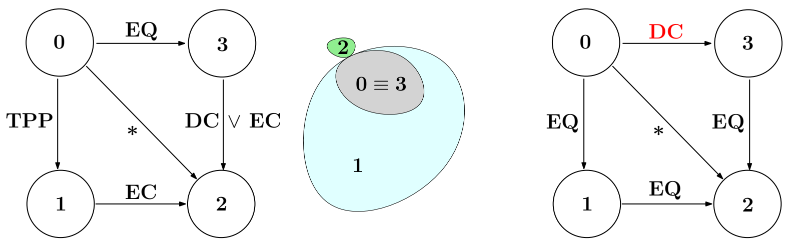
Theorem 1. *For a given* RCC-*8, or* IA*, network $\mathcal{N} \uplus \mathcal{N}' = (V'', C'')$ with relations from classes $\hat{\mathcal{H}}_8, \mathcal{C}_8$, and $\mathcal{Q}_8$, or $\mathcal{H}_{\mathsf{IA}}$, respectively, where $\mathcal{N} = (V, C)$ is the initial partially path consistent* QCN *augmented by a new* QCN *$\mathcal{N}' = (V', C')$, and $G = (V, E)$ and $G' = (V', E')$ are their respective underlying chordal graphs where $\forall(v, u) \in E'$ we have that $v \in V' \setminus V$ or $u \in V' \setminus V$, function* iPPC+ *decides the consistency of* QCN *$\mathcal{N} \uplus \mathcal{N}'$ with respect to chordal graph $G \cup G'$.*

Regarding data structures, the QCNs are represented by a hash table based adjacency list as described in [20]. This recent technical advancement in qualitative reasoning allows us to extend a QCN with new temporal or spatial entities in constant time.

Function GiPPC($\mathcal{N}$, G)

```
in     : A QCN $\mathcal{N} = (V, C)$, and a chordal graph $G = (V, E)$.
output : False if network $\mathcal{N}$ results in a trivial inconsistency, True if the
         modified network $\mathcal{N}$ is partially path consistent.
begin
    $\mathcal{N}_1 \uplus \mathcal{N}_2 \uplus \ldots \uplus \mathcal{N}_i \leftarrow \mathcal{N}$; $\mathcal{N}' \leftarrow \mathcal{N}_1$;
    foreach $k \leftarrow 2$ to $i$ do
        if ! iPPC+($\mathcal{N}' \uplus \mathcal{N}_k, G', G_k$) then return False;
        $\mathcal{N}' \leftarrow \mathcal{N}' \uplus \mathcal{N}_k$;
    $\mathcal{N} \leftarrow \mathcal{N}'$;
    return True;
```

GiPPC receives as input a QCN $\mathcal{N}$ together with its underlying chordal graph G, and applies iPPC+ iteratively (line 4) on a decomposition of $\mathcal{N}$ (line 2). This decomposition can be any partition of the set of variables. If we start with a single-entity QCN $\mathcal{N}_1$ and extend it with a new QCN $\mathcal{N}_i$ of one new entity at a time applying iPPC+ in total $n-1$ times (thus, $2 \leq i \leq n$), it follows that we will perform $O(\delta_2 \cdot |E_2| + \ldots + \delta_n \cdot |E_n|)$ intersection and composition operations for constructing a QCN of n temporal or spatial entities, where δ_i is the maximum degree of a vertex of chordal graph $((G_1 \cup G_2) \cup \ldots) \cup G_i$ and $O(|E_i|)$ is the number of constraints that the new QCN $\mathcal{N}_i$ contributes to QCN $((\mathcal{N}_1 \uplus \mathcal{N}_2) \uplus \ldots) \uplus \mathcal{N}_i$. As $\delta_2 \leq \ldots \leq \delta_n$ and $E_2 \cup \ldots \cup E_n = E$ (i.e., the no. of edges in the n entities constraint network after the n^{th} entity is added), it follows that the complexity of iPPC+ is asymptotically upper bounded by $O(\delta_n \cdot |E|)$, which is the complexity of PPC. Thus, we increase on average the performance of applying partial path consistency, as we will also find out in the experimentation to follow, and retain the same worst-case complexity.



(a) consistent QCN along with a solution (b) inconsistent QCN

Fig. 2: QCNs with respect to their constraint graphs

3.1 Running Example

Before moving on to our running example, it is important to explain the notions of *processed edges* and *consistency checks*. An edge is processed whenever it is popped out of the queue (line 4), and a consistency check takes place whenever we apply the intersection operator ($\cap$) between two constraints (lines 6 and 11). In our running example we will demonstrate how iPPC+ is able to perform better than the one-shot partial path consistency algorithm (PPC) originally presented in [9] for the case of IA and in [21] for the case of RCC-8. In what follows, we always give the chordal graph $G \cup G'$ of QCN $\mathcal{N} \uplus \mathcal{N}'$ as input to iPPC+ to facilitate description, as the initial network $\mathcal{N}$, along with graph G, and its augmentation $\mathcal{N}'$, along with graph G', are easily identifiable at each step.

Consistent case. Let us consider the consistent RCC-8 network in Figure 2a. We will first build a partial path consistent version of this network incrementally, beginning with node 0 and adding nodes 1, 2, and, 3, one at each step. We always pop edges from the left of the queue and push to the right (FIFO), and whenever needed we use the converse relation corresponding to an edge. First, $(\{0\}, \emptyset)$ is given as input to iPPC+ with no edges whatsoever, the queue is initialized by an empty set, no edges are processed, and, thus, no consistency checks occur. Then, $(\{0, 1\}, \{(0, 1)\})$ is given as input to iPPC+, the queue is initialized with the set of edges $\{(0, 1)\}$, a single edge is processed, and no consistency checks occur as there are no triangles in the network. Then, $(\{0, 1, 2\}, \{(0, 1), (0, 2), (1, 2)\})$ is given as input to iPPC+, the queue is initialized with the set of edges $\{(1, 2)\}$, viz., the constraint edges that accompany the newly inserted spatial entity (region) 2. Edge $(0, 2)$ is not included in the queue as it corresponds to the universal relation $*$ and we do not consider it at all during initialization (this detail is not provided in algorithm iPPC+). (The intersection of $*$ with any other relation leaves the latter relation intact.) Edge $(1, 2)$ is popped out of the queue. Two consistency checks take place among edges $(0, 1)$, $(0, 2)$, and $(1, 2)$, leading to the pruning of the universal relation $*$ for edge $(0, 2)$ into the relation $DC \vee EC$, and, edge $(0, 2)$ is inserted in the queue which now holds the set of edges $\{(0, 2)\}$. Edge $(0, 2)$ is popped out of the queue. Two consistency checks take place among edges $(0, 1)$, $(0, 2)$, and $(1, 2)$, leading to no pruning of relations for edges $(0, 1)$ and $(1, 2)$. Finally, $(\{0, 1, 2, 3\}$, $\{(0, 1)$, $(0, 2)$, $(0, 3)$,

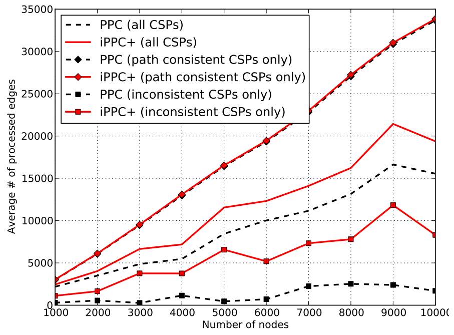
$(1,2)$, $(2,3)\}$) is given as input to iPPC+, the queue is initialized with the set of edges $\{(0,3),(2,3)\}$. Both edges are popped out of the queue, each one leading to two consistency checks, with no further pruning of relations. In total we have processed 5 edges and performed 8 consistency checks.

We now proceed with PPC which is fairly easier to describe. First, ($\{0,1,2,3\}$, $\{(0,1)$, $(0,2)$, $(0,3)$, $(1,2)$, $(2,3)\}$) is given as input to PPC. The queue is initialized with the set of edges $\{(0,1)$, $(0,3)$, $(1,2)$, $(2,3)\}$. Edge $(0,1)$ is popped out of the queue. Two consistency checks take place among edges $(0,1)$,$(0,2)$, and $(1,2)$, leading to the pruning of the universal relation $*$ for edge $(0,2)$ into the relation $DC \vee EC$. Edge $(0,2)$ is inserted in the queue which now holds the set of edges $\{(0,3)$, $(1,2)$, $(2,3)$, $(0,2)\}$. All edges are popped out of the queue with no further pruning of relations. Edges $(0,3)$, $(1,2)$, and $(2,3)$ lead to two consistency checks each, and edge $(0,2)$ to four, as it is part of two triangles. In total we have processed 5 edges and performed 12 consistency checks.

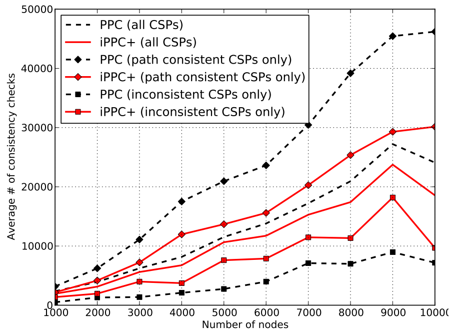
PPC proccesses the same number of edges as iPPC+, but performs 4 more consistency checks. The numbers may vary a bit depending on the order of the edges in the initialized queue (in our running example we have considered a sorted initial queue of edges), but the trend is that for consistent QCNs, iPPC+ will perform less consistency checks than PPC, and will process only slightly more edges than PPC, depending on whether an edge already exists in the queue or not. (Since PPC works with a large queue, an edge might not have to be popped and pushed often as it may already exist in queue.)

Inconsistent case. Let us consider now the inconsistent QCN in Figure 2b. Regions 0, 1, 2, and 3 are all equal to each other (they are essentially the one same region), thus, regions 0 and 3 can not be disconnected. We will not go into detail as we did with the consistent case, by now the reader should be able to verify (assuming a sorted initial queue of edges in every case) that iPPC+ processes in total 4 edges and performes 5 consistency checks, while PPC processes in total 2 edges and performes 3 consistency checks. In fact, Figure 2b describes the worst case scenario for iPPC+; an inconsistency that occurs when the last temporal or spatial entity is added in the network. By that point iPPC+ will have already fully reasoned with all previous entities and their accompanying constraints. On the other hand, PPC will always do a first iteration of the queue, and might be able to immediately capture the inconsistency, as with our running example. Again, the numbers may vary a bit depending on the order of the edges in the initialized queue, but the trend is that for inconsistent QCNs iPPC+ will perform more consistency checks than PPC, and will process more edges than PPC.

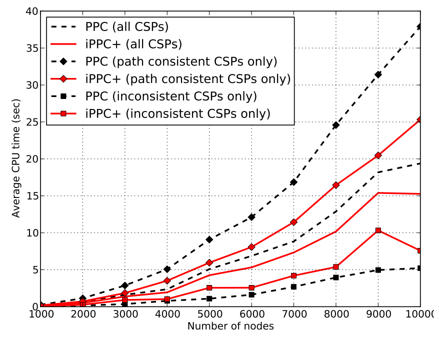
Concluding our running example, we have observed that iPPC+ should perform better than PPC in the case of consistent QCNs and worse than PPC in the case of inconsistent QCNs. Further, iPPC+ works with a very small queue at each step. It is natural that a path consistency algorithm will run faster when there is an inconsistency in the input network, as the inconsistency will not allow the algorithm to reason exhaustively with the network relations. Thus, we expect that the overall performance of iPPC+ should be better than that of PPC in the average case. We are about to experimentally verify this in the next section.



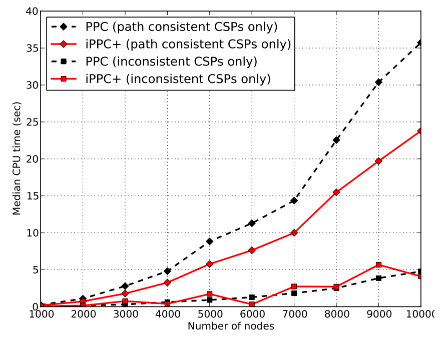
(a) Average # of processed edges



(b) Average # of consistency checks



(c) Average CPU time (sec)



(d) Median CPU time (sec)

Fig. 3: Performance comparison of iPPC+ and PPC for RCC-8 networks

4 Experimental evaluation

We considered random datasets consisting of *large* IA and RCC-8 networks generated by the BA(n, m) model [3], the use of which is well motivated in [20], and the standard A(n, d, l) model [19], used extensively in literature. In short, BA(n, m) creates random scale-free-like networks of size n and a preferential attachment value m, and A(n, d, l) creates random networks of size n, degree d, and an average number l of IA and RCC-8 relations per edge. For model BA(n, m) the average number of IA or RCC-8 relations per edge defaults to $|\mathsf{B}|/2$, where B is the set of base relations of IA or RCC-8 respectively. We also considered real RCC-8 datasets that consist of `admingeo` [12] and `gadm-rdf` (`http://gadm.geovocab.org/`) comprising 11761/77907 nodes/edges and 276728/590865 nodes/edges respectively. In short, `admingeo` describes the administrative geography of Great Britain using RCC-8 relations, and `gadm-rdf` the world's administrative areas likewise. The experiments were carried out on a computer with a CPU frequency of 2.00 GHz, 4 GB RAM, and the Lucid Lynx x86_64 OS (Ubuntu Linux). The implementations of iPPC+ and PPC were run with the CPython interpreter (`http://www.python.org/`), which implements Python 2.6. Only one of the CPU cores was used for the experiments. Regarding iPPC+, we begin with a single node and grow the network one node at a time. All tools and datasets used in this paper can be found online in the following address: `http://www.cril.fr/~sioutis/work.php`.

Random datasets. BA(n, m) model: For RCC-8 we considered 30 networks for each size between 1000 and 10000 nodes with a 1000 step and a preferential attachment value of $m = 2$. For this specific value of m and for the network sizes considered, the networks lie within the *phase transition* region, where it is equally possible for networks to be consistent or inconsistent, thus, they are harder to solve [20]. We assess the performance of iPPC+ and PPC using the following parameters: average CPU time, median CPU time, average number of processed edges, and average number of consistency checks. On the average case, i.e., when all networks are considered, iPPC+ processes around 26.8% more edges than PPC, as shown in Figure 3a, iPPC+ performs around 15.3% less consistency checks than PPC, as shown in Figure 3b, and, finally, regarding average CPU time, iPPC+ runs around 18.9% faster than PPC, and 21.3% faster in the final step where networks of 10000 nodes are considered, as shown in Figure 3c; for the networks of 10000 nodes iPPC+ runs in an average time of 15.3 sec, and PPC in 19.4 sec. In Figure 3d we can also see the median CPU time for path consistent and inconsistent networks. The interesting thing to note is that in the case of inconsistent networks the median allows us to get rid of some outlying measurements that influence the average CPU time in Figure 3c. As our dataset consists of a little more than 50% inconsistent networks for almost all network sizes, the diagram for the median CPU time for the combined dataset of path consistent and inconsistent networks was very close to that of the inconsistent case, as in almost all cases the median would correspond to the CPU processing time of an inconsistent network. Thus, we did not include this diagram as it was pretty erratic and did not offer any additional information.

For IA we considered 30 networks for each size between 500 and 5000 nodes with a 500 step and a preferential attachment value of $m = 3$. We found that for this specific value of m and for the network sizes considered, the networks lie within the *phase transition* region, as is the case with RCC-8. However, we note that the phase transition for IA occurs for a different value of m (viz., $m = 3$) than the value of m for RCC-8 (viz., $m = 2$). This is probably because IA is a bigger calculus than RCC-8, containing 13 base relations instead of 8 respectively, which allows for consistent networks to be denser as there are more relations to be pruned and more relations that can *support* consistency in the network. On the average case, iPPC+ runs around 27.4% faster than PPC, and 34.4% faster in the final step where networks of 5000 nodes are considered; for the networks of 5000 nodes iPPC+ runs in an average time of 44 sec, and PPC in 67.1 sec.

A(n, d, l) model: Due to space constraints it is not possible to give analytical figures regarding this model. However, it should suffice to note that experimentation using this model yielded qualitatively similar results with those using the BA(n, m) model, i.e., we had roughly the same trends and speed-ups for both RCC-8 and IA calculi. In particular, for 50 RCC-8 networks of 1000 nodes and 50

IA networks of 500 nodes in the phase transition region there was a speed-up of around 20% and 30% respectively.

Real datasets. Both `admingeo` and `gadm-rdf` are consistent RCC-8 networks comprising 11761/77907 nodes/edges and 276728/590865 nodes/edges respectively. iPPC+ was able to enforce partial path consistency on `admingeo` in 267.6 sec, and PPC in 349.08 sec. Hence, regarding `admingeo`, iPPC+ runs 22.6% faster than PPC. Regarding `gadm-rdf`, iPPC+ outruns PPC even more significantly. In particular, iPPC+ was able to enforce partial path consistency on `gadm-rdf` in 6.27 sec, and PPC in 10.88 sec. This translates to iPPC+ running 42.4% faster than PPC regarding `gadm-rdf`. We note that our findings concerning real datasets agree with the findings concerning random datasets. Surprisingly, both algorithms run the `gadm-rdf` experiment faster than the `admingeo` one, but this is due to more relations being inferred in the latter case as a result of dataset particularities that affect the reasoning process.

At this point we conclude our experimentation. We have demonstrated that iPPC+ performs better than PPC on average for random QCNs of IA and RCC-8, and real QCNs of RCC-8. It should be noted that since iPPC+ works with a small queue at each incrementation step (as it considers the edges of a subnetwork of the whole network), as opposed to a full queue utilized by PPC (as it considers the edges of the whole network), iPPC+ is also much more memory efficient. To the best of our knowledge, the networks of IA and RCC-8 used in this paper are the biggest ones to date of all others that exist in literature (which scale up to a few hundred nodes only).

5 Conclusion and Future work

In this paper we presented an algorithm, viz., iPPC+, for maintaining or enforcing partial path consistency for growing constraint networks, i.e., networks that grow with new temporal or spatial entities over time. Through a complexity analysis, and thorough experimental evaluation, we showed that iPPC+ is able to perform better than the state-of-the-art one-shot partial path consistency algorithm (PPC) originally presented in [9] for IA and in [21] for RCC-8, which is an advancement in the field of qualitative reasoning. The importance of our results can be roughly compared with those of Bessiére in [6], in that we also present an improvement of the state-of-the-art path consistency algorithm that on average increases its performance by avoiding redundant consistency checks, and, thus, table lookups. However, in our case, our approach is both more time and memory efficient and appropriate for all network sizes. Thus, state-of-the-art reasoners can immediately gain a performance boost by opting for iPPC+ as the preprocessing step in their backtracking algorithms for general networks, but also for solving tractable QCNs.

Future work consists of exploring if we can also have an approach for incrementally building consistent QCNs by using iPPC+ as the consistency checking step of a backtracking algorithm, i.e., we would like to investigate if in backtracking there is any benefit in reasoning with a smaller network, as a result of reversing the incremental building process. Finally, iPPC+ is tailored for dynamic spatial and temporal reasoning and, in this regard, it can become completely online by implementing a mechanism to incrementally maintain chordality [5].

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. CACM 26, 832–843 (1983)
2. Amaneddine, N., Condotta, J.F., Sioutis, M.: Efficient Approach to Solve the Minimal Labeling Problem of Temporal and Spatial Qualitative Constraints. In: IJCAI (2013)
3. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. Science (New York, N.Y.) 286, 509–512 (1999)
4. Berry, A., Blair, J.R.S., Heggernes, P.: Maximum Cardinality Search for Computing Minimal Triangulations. In: WG (2002)
5. Berry, A., Heggernes, P., Villanger, Y.: A vertex incremental approach for maintaining chordality. Discrete Mathematics 306 (2006)
6. Bessière, C.: A Simple Way to Improve Path Consistency Processing in Interval Algebra Networks. In: AAAI/IAAI (1996)
7. Bhatt, M., Guesgen, H., Wölfl, S., Hazarika, S.: Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions. Spatial Cognition & Computation 11, 1–14 (2011)
8. Bliek, C., Sam-Haroud, D.: Path consistency on triangulated constraint graphs. In: IJCAI (1999)
9. Chmeiss, A., Condotta, J.F.: Consistency of Triangulated Temporal Qualitative Constraint Networks. In: ICTAI (2011)
10. Gerevini, A.: Incremental qualitative temporal reasoning: Algorithms for the point algebra and the ord-horn class. Artif. Intell. 166(1-2), 37–80 (2005)
11. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Elsevier Science, 2nd edn. (2004)
12. Goodwin, J., Dolbear, C., Hart, G.: Geographical Linked Data: The Administrative Geography of Great Britain on the Semantic Web. TGIS 12, 19–30 (2008)
13. Huang, J.: Compactness and its implications for qualitative spatial and temporal reasoning. In: KR (2012)
14. Nebel, B.: Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class. In: ECAI (1996)
15. Planken, L., de Weerdt, M., Yorke-Smith, N.: Incrementally Solving STNs by Enforcing Partial PC. In: ICAPS (2010)
16. Randell, D.A., Cui, Z., Cohn, A.: A Spatial Logic Based on Regions and Connection. In: KR (1992)
17. Renz, J.: Maximal Tractable Fragments of the Region Connection Calculus: A Complete Analysis. In: IJCAI (1999)
18. Renz, J., Ligozat, G.: Weak Composition for Qualitative Spatial and Temporal Reasoning. In: CP (2005)
19. Renz, J., Nebel, B.: Efficient Methods for Qualitative Spatial Reasoning. JAIR 15, 289–318 (2001)
20. Sioutis, M., Condotta, J.F.: Tackling large Qualitative Spatial Networks of scale-free-like structure. In: SETN (2014)
21. Sioutis, M., Koubarakis, M.: Consistency of Chordal RCC-8 Networks. In: ICTAI (2012)