Empirical study of algorithms for qualitative temporal or spatial constraint networks

Jean-François Condotta, Gérard Ligozat, Mahmoud Saade

Empirical study of algorithms for qualitative temporal or spatial constraint networks

Jean-François Condotta[1] **Gérard Ligozat**[2] **Mahmoud Saade**[1]

[1] **CRIL-CNRS, Université d'Artois, 62307 Lens Cedex, France**
[2] **LIMSI-CNRS, Université d'Orsay, 91403 Orsay, France**
ligozat@limsi.fr, {condotta, saade}@cril.univ-artois.fr

Abstract. Representing and reasoning about spatial and temporal information is an important task in many applications of Artificial Intelligence. In the past two decades numerous formalisms using qualitative constraint networks have been proposed for representing information about time and space. Most of the methods used to reason with these constraint networks are based on the weak composition closure method. The goal of this paper is to study some implementations of these methods, including three well known and very used implementations, and two new ones.

1 Introduction

Representing and reasoning about spatial and temporal information is an important task in many applications, such as geographic information systems (GIS), natural language understanding, robot navigation, temporal and spatial planning. Qualitative spatial and temporal reasoning aims to describe non-numerical relationships between spatial or temporal entities. Typically a qualitative calculus [1, 17, 10, 16, 7] uses some particular kind of spatial or temporal objects (subsets in a topological space, points on the rational line, intervals on the rational line,...) to represent the spatial or temporal entities of the system, and focuses on a limited range of relations between these objects (such as topological relations between regions or precedence between time points). Each of these relations refers to a particular temporal or spatial configuration. For instance, consider the well-known temporal qualitative formalism called Allen's calculus [1]. It uses intervals of the rational line for representing temporal entities. Thirteen basic relations between these intervals are used to represent the qualitative situation between temporal entities (see Figure 1). For example, the basic relation *overlaps* can be used to represent the situation where a first temporal activity starts before a second activity and terminates while the latter is still active.

Now the temporal or spatial information about the configuration of a specific set of entities can be represented using a particular kind of constraint networks called qualitative constraint networks (QCNs). Each constraint of a QCN represents a set of acceptable qualitative configurations between some temporal or spatial entities and is defined by a set of basic relations. Given a QCN $\mathcal{N}$, the main problems to be considered are the following ones: decide whether there exists a solution of $\mathcal{N}$ (the consistency problem), find one or several solutions of $\mathcal{N}$; find one or several consistent scenarios of $\mathcal{N}$; determine the minimal QCN of $\mathcal{N}$. In order to solve these problems, methods based on local constraint propagation algorithms have been defined, in particular algorithms based on the $\circ$-closure method (called also the path consistency method) [2, 4, 8, 9, 6, 5, 18, 15] which is the qualitative version of the path consistency method [14, 12] used in the domain of classical CSPs. Roughly speaking the $\circ$-closure method is a constraint propagation method which consists in iteratively performing an operation called the triangulation operation which removes for each constraint defined between two variables the basic relations not allowed w.r.t. a third variable. In following the line of reasoning of van Beek and Manchak [5] and Bessière [6], in this paper we compare different possible versions of the $\circ$-closure method. The algorithms studied are adapted from the algorithms PC1[14] or PC2 [11]. Concerning the algorithms issued of PC2 we use different heuristics, in particular heuristics defined in [5] and we use structures saving pairs of constraints or structures saving triples of constraints. Moreover we introduce two algorithms mixing the algorithm PC1 and the algorithm PC2.

This paper is organized as follows. In Section 2, we give some general definitions concerning the qualitative calculi. Section 3 is devoted to the different $\circ$-closure algorithms studied in this paper. After discussing the realized experimentations in Section 4 we conclude in Section 5.

2 Background on Qualitative Calculi

2.1 Relations

In this paper, we focus on binary qualitative calculi and use very general definitions. A qualitative calculus considers a finite set $\mathcal{B}$ of k binary relations defined on a domain D. These relations are called basic relations. The elements of D are the possible values to represent the temporal or spatial entities. The basic relations of $\mathcal{B}$ correspond to all possible configurations between two temporal or spatial entities. The relations of $\mathcal{B}$ are jointly exhaustive and pairwise disjoint, which means that any pair of elements of D belongs to exactly one basic relation in $\mathcal{B}$. Moreover, for each basic relation $B \in \mathcal{B}$

there exists a basic relation of $\mathcal{B}$, denoted by $B^{\sim}$, corresponding to the converse of B. The set $\mathcal{A}$ is defined as the set of relations corresponding to all unions of the basic relations: $\mathcal{A} = \{\bigcup B : B \subseteq \mathcal{B}\}$. It is customary to represent an element $B_1 \cup \ldots \cup B_m$ (with $0 \leq m \leq k$ and $B_i \in \mathcal{B}$ for each i such that $1 \leq i \leq m$) of $\mathcal{A}$ by the set $\{B_1, \ldots, B_m\}$ belonging to $2^{\mathcal{B}}$. Hence we make no distinction between $\mathcal{A}$ and $2^{\mathcal{B}}$ in the sequel. There exists an element of $\mathcal{A}$ which corresponds to the identity relation on D, we denote this element by Id. Note that this element can be composed of several basic relations. Now we give some well known examples of calculi to illustrate this definition.

The Allen's calculus. As a first example, consider the well known temporal qualitative formalism called Allen's calculus [1]. It uses intervals of the rational line for representing temporal entities. Hence D is the set $\{(x^-, x^+) \in \mathbb{Q} \times \mathbb{Q} : x^- < x^+\}$. The set of basic relations consists in a set of thirteen binary relations corresponding to all possible configurations of two intervals. These basic relations are depicted in Figure 1. Here we have $\mathcal{B} = \{eq, b, bi, m, mi, o, oi, s, si, d, di, f, fi\}$. Each basic relation can be formally defined in terms of the endpoints of the intervals involved; for instance, $m = \{((x^-, x^+), (y^-, y^+)) \in \mathsf{D} \times \mathsf{D} : x^+ = y^-\}$. The set $\{b, m\} \in 2^{\mathcal{B}}$ corresponds to the relation $b \cup m$ of $\mathcal{A}$. Moreover, we have $\mathsf{Id} = \{eq\}$.

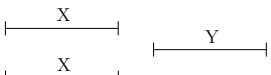
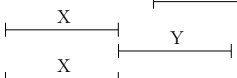
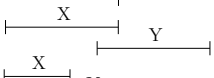
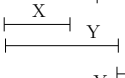
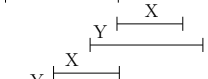
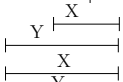
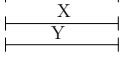
Relation	Symbol	Inverse	Meaning
precedes	b	bi	
meets	m	mi	
overlaps	o	oi	
starts	s	si	
during	d	di	
finishes	f	fi	
equals	eq	eq	

Figure 1. The basic relations of the Allen's calculus.

The point algebra. As second example, consider a temporal qualitative formalism weaker than the Allen's calculus, namely the point algebra. It uses points of the rational line for representing temporal entities. Hence D is the set $\mathbb{Q}$. The set of basic relations consists in three binary relations corresponding to all possible configurations of two points : $\mathcal{B} = \{precedes, follows, same\}$ with $precedes = \{(x, y) \in \mathsf{D} \times \mathsf{D} : x < y\}$, $follows = \{(x, y) \in \mathsf{D} \times \mathsf{D} : x > y\}$ and $same = \{(x, y) \in \mathsf{D} \times \mathsf{D} : x = y\}$. Moreover, we have $\mathsf{Id} = \{same\}$.

The Meiri's calculus. Meiri [13] considers temporal qualitative constraints on both intervals and points. These constraints can correspond to the relations of a qualitative formalism defined in the following way. D is the set of pairs of rational numbers: $\{(x, y) : x \leq y\}$. The pairs (x, y) with $x < y$ correspond to intervals and the pairs (x, y) with $x = y$ correspond to points. Hence, we define to particular basic relations on D : $eq^i = \{((x, y), (x, y)) : x < y\}$ and $eq^p = \{((x, y), (x, y)) : x = y\}$ composing Id. These basic relations allow to constraint an object to be an interval or a point. In addition of these basic relations, the basic relations of the Allen's calculus and those ones of the point algebra are added to $\mathcal{B}$. To close the definition of $\mathcal{B}$ we must include the ten basic relations corresponding to the possible configurations between a point and an interval, see 2 for an illustration of these basic relations.

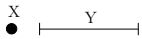
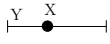
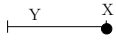
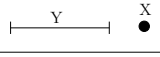
Relation	Symbol	Inverse	Meaning
before	b*	bi*	
starts	s*	si*	
during	d*	di*	
finishes	f*	fi*	
after	a*	ai*	

Figure 2. The basic relations of the Meiri's calculus concerning a point X and an interval Y.

2.2 Fundamental operations

As a set of subsets, $\mathcal{A}$ is equipped with the usual set-theoretic operations including intersection ($\cap$) and union ($\cup$). As a set of binary relations, it is also equipped with the operation of converse ($\sim$) and an operation of composition ($\circ$) sometimes called weak composition or qualitative composition. The converse of a relation R in $\mathcal{A}$ is the relation of $\mathcal{A}$ corresponding to the transpose of R; it is the union of the converses of the basic relations contained in R. The composition $A \circ B$ of two basic relations A and B is the relation $R = \{C : \exists x, y, z \in \mathsf{D}, x\ A\ y, y\ B\ z \text{ and } x\ C\ z\}$. The composition $R \circ S$ of $R, S \in \mathcal{A}$ is the relation $T = \bigcup_{A \in R, B \in S} \{A \circ B\}$. Computing the results of these various operations for relations of $2^{\mathcal{B}}$ can be done efficiently by using tables giving the results of these operations for the basic relations of $\mathcal{B}$. For instance, consider the relations $R = \{eq, b, o, si\}$ and $S = \{d, f, s\}$ of Allen's calculus, we have $R^{\sim} = \{eq, bi, oi, s\}$. The relation $R \circ S$ is $\{d, f, s, b, o, m, eq, si, oi\}$. Consider now the relations $R = \{b^*, s^*\}$ and $S = \{b\}$ of the Meiri's calculus, we have $R \circ S = \{b^*\}$ whereas $S \circ R = \{\}$.

2.3 Qualitative Constraint Networks

A qualitative constraint network (QCN) is a pair composed of a set of variables and a set of constraints. The set of variables represents spatial or temporal entities of the system. A constraint consists of a set of acceptable basic relations (the possible configurations) between some variables. Formally, a QCN is defined in the following way:

Definition 1 *A QCN is a pair $\mathcal{N} = (V, C)$ where:*

- $V = \{v_1, \ldots, v_n\}$ *is a finite set of* n *variables where* n *is a positive integer;*
- C *is a map which to each pair* (v_i, v_j) *of* $V \times V$ *associates a subset* $C(v_i, v_j)$ *of the set of basic relations:* $C(v_i, v_j) \in 2^{\mathcal{B}}$. *In the sequel* $C(v_i, v_j)$ *will be also denoted by* C_{ij}. C *is such that* $C_{ii} \subseteq \mathsf{Id}$ *and* $C_{ij} = C_{ji}^{\sim}$ *for all* $v_i, v_j \in V$.

With regard to a QCN $\mathcal{N} = (V, C)$ we have the following definitions: A *solution* of $\mathcal{N}$ is a map σ from V to D such that $(\sigma(v_i), \sigma(v_j))$ satisfies C_{ij} for all $v_i, v_j \in V$. $\mathcal{N}$ is consistent iff it admits a solution. A QCN $\mathcal{N}' = (V', C')$ is a *sub*-QCN of $\mathcal{N}$ if and only if $V = V'$ and $C'_{ij} \subseteq C_{ij}$ for all $v_i, v_j \in V$. A QCN $\mathcal{N}' = (V', C')$ is *equivalent* to $\mathcal{N}$ if and only if $V = V'$ and both networks $\mathcal{N}$ and $\mathcal{N}'$ have the same solutions. The *minimal* QCN of $\mathcal{N}$ is the smallest (for $\subseteq$) sub-QCN of $\mathcal{N}$ equivalent to $\mathcal{N}$. An *atomic* QCN is a QCN such that each C_{ij} contains a basic relation. A *consistent scenario* of $\mathcal{N}$ is a consistent atomic sub-QCN of $\mathcal{N}$.
Given a QCN $\mathcal{N}$, the main problems to be considered are the following problems: decide whether there exists a solution of $\mathcal{N}$; find one or several solutions of $\mathcal{N}$; find one or several consistent scenarios of $\mathcal{N}$; determine the minimal QCN of $\mathcal{N}$. Most of the algorithms used for solving these problems are based on a method which we call the $\circ$-closure method. The next section is devoted to this method.

3 The $\circ$-closure method and associated algorithms

3.1 Generalities on the $\circ$-closure method

In this section we introduce the path $\circ$-closure property and give the different implementations of this method studied in the sequel. Roughly speaking the $\circ$-closure method is a constraint propagation method which consists in iteratively performing the following operation (the triangulation operation): $C_{ij} := C_{ij} \cap (C_{ik} \circ C_{kj})$, for all variables v_i, v_j, v_k of V, until a fixed point is reached. Just no satisfiable basic relations are removed from these constraints with this method. In the case where the QCN obtained in this way contains the empty relation as a constraint, we can assert that the initial QCN is not consistent. However, if it does not, we cannot in the general case infer the consistency of the network. Hence the QCN obtained in this way is a sub-QCN of $\mathcal{N}$ which is equivalent to it. Moreover, the obtained QCN is $\circ$-closed, more precisely it satisfies the following property: $C_{ij} \subseteq C_{ik} \circ C_{kj}$ for all variables v_i, v_j, v_k of V. Note that this property implies the $(0, 3)$-consistency of the resulting QCN (each restriction on 3 variables is consistent). For several calculi, in particular for the Allen's calculus defined on the rational intervals, the $(0, 3)$-consistency implies the 3 consistency or path consistency [11]. It is why sometimes there exists a confusion between the $\circ$-closure property and the path consistency property.

3.2 Studied Algorithms

There are two well known algorithms in the literature for enforcing the path-consistency of discrete CSPs [11, 14], namely the PC1 and the PC2 algorithms. These algorithms have been adapted on several occasions to the binary qualitative case in order to enforce $\circ$-closure [2, 19, 8, 3, 6].
A possible adaptation of PC1 to the qualitative case is the function WCC1 defined in Algorithm 1. WCC1 checks all triples of variables of the network in a main loop. It starts again this main loop until no changes occur. For each triple of variables the operation of triangulation is made by the function *revise*. Note that in this function the call of $updateConstraints(C_{ij}, R)$ allows to set the constraint C_{ij} with the new relation R and to set the constraint C_{ij} with $R^{\sim}$. For particular situations, the treatment corresponding to lines 7–9 can be avoided. For example, for the QCNs defined from relations of the Allen's calculus this treatment is an useless work in the following cases : $C_{ik} = \mathcal{B}$, $C_{kj} = \mathcal{B}$, $i = k$, $k = j$ or $i = j$. This is respectively due to the facts that $\mathcal{B} \circ R = R \circ \mathcal{B} = \mathcal{B}$ for all non empty relation $R \in \mathcal{A}$, Id is composed by a basic relation (eq) and $\mathsf{Id} \subseteq R \circ R^{\sim}$ for all non empty relation R. Note that these properties are not always true for another calculus, see the Meiri's calculus for example. It is why we introduce a conditional statement at line 6 allowing to avoid fruitless work by defining a good predicate *skippingCondition ad hoc* to the qualitative calculus used. For example, in the case of the Allen's calculus, *skippingCondition* could be defined by the following instruction: *return* ($C_{ik} == \mathcal{B}$ *or* $C_{kj} == \mathcal{B}$ *or* $i == k$ *or* $k == j$ *or* $i == j$). For this calculus this skipping condition can be more elaborated, see [5]. The time complexity of WCC1 is $O(|\mathcal{B}| * n^5)$ whereas its spatial complexity is $O(|\mathcal{B}| * n^3)$.

Algorithm 1

Function WCC1($\mathcal{N}$), with $\mathcal{N} = (V, C)$.
1: **repeat**
2: *change* ← **false**
3: **for** $i \leftarrow 1$ to n **do**
4: **for** $j \leftarrow i$ to n **do**
5: **for** $k \leftarrow 1$ to n **do**
6: **if** *not skippingCondition*(C_{ik}, C_{kj}) **then**
7: **if** *revise*(i, k, j) **then**
8: **if** $C_{ij} == \emptyset$ **then return false**
9: **else** *change* ← **true**
10: **until not** *change*
11: **return true**

Function revise(i, k, j).
1: $R \leftarrow C_{ij} \cap (C_{ik} \circ C_{kj})$
2: **if** $C_{ij} \subseteq R$ **then return false**
3: updateConstraints(C_{ij}, R)
4: **return true**

The functions WCC2_P and WCC2_T defined in respectively Algorithm 2 and Algorithm 3 are inspired by PC2. WCC2_P handles a list containing pairs of variables corresponding to the modified constraints which must be propagated whereas WCC2_P handles a list containing triples of variables corresponding to the operations of triangulation to realize. The using of triples instead of pairs allows to circumscribe more precisely the useful triangulation operations. In the previous algorithms proposed in the literature, the exact nature of the list manipulated is not very clear, this list could be a set, a queue or still a stack. In WCC2_P and WCC2_T the nature of the list is connected with the nature of the object *heuristic* which is commissioned to handle it. The main task of *heuristic* consists in the insertion of a pair or a triple of variables in the list. It must compute a location in the list and places it. If the pair or the triple is already in the list it

can insert it or do nothing. The method *next* always consists in removing and returning the first element of the list. In the sequel we will describe the used heuristics with more details. The predicate *skippingCondition*, like in WCC1, depends on the qualitative calculus used. For the Allen's calculus and for most of the calculi $skippingCondition(C_{ij})$ can be defined by the following instruction: $return\ (C_{ij} == \mathcal{B})$. The time complexity of WCC2_P and WCC2_T is $O(|\mathcal{B}| * n^3)$ whereas the spatial complexity of WCC2_P is $O(n^2)$ and this one of WCC2_T is $O(n^3)$.

Algorithm 2

Function WCC2_P($\mathcal{N}, heuristic$), with $\mathcal{N} = (V, C)$.
1: $Q \leftarrow \emptyset$
2: $initP(\mathcal{N}, Q, heuristic)$
3: **while** $Q \neq \emptyset$ **do**
4: $(i, j) \leftarrow heuristic.next(Q)$
5: **for** $k \leftarrow 1$ to n **do**
6: **if** $revise(i, j, k)$ **then**
7: **if** $C_{ik} == \emptyset$ **then return false**
8: **else** $addRelatedPathsP(i, k, Q, heuristic)$
9: **if** $revise(k, i, j)$ **then**
10: **if** $C_{kj} == \emptyset$ **then return false**
11: **else** $addRelatedPathsP(k, j, Q, heuristic)$
12: **done**
13: **end while**
14: **return true**

Function initP($\mathcal{N}, Q, heuristic$).
1: **for** $i \leftarrow 1$ to n **do**
2: **for** $j \leftarrow i$ to n **do**
3: **if** *not* $skippingCondition(C_{ij})$ **then**
4: $addRelatedPathsP(i, j, Q, heuristic)$

Function addRelatedPathsP($i, j, Q, heuristic$).
1: $heuristic.append(Q, (i, j))$

Algorithm 3

Function WCC2_T($\mathcal{N}$), with $\mathcal{N} = (V, C)$.
1: $Q \leftarrow \emptyset$
2: $initT(\mathcal{N}, Q, heuristic)$
3: **while** $Q \neq \emptyset$ **do**
4: $(i, k, j) \leftarrow heuristic.next(Q)$
5: **if** $revise(i, k, j)$ **then**
6: **if** $C_{ij} == \emptyset$ **then return false**
7: **else** $addRelatedPathsT(i, j, Q, heuristic)$
8: **end while**
9: **return true**

Function initT($N, Q, heuristic$).
1: **for** $i \leftarrow 1$ to n **do**
2: **for** $j \leftarrow i$ to n **do**
3: **if** *not* $skippingCondition(C_{ij})$ **then**
4: $addRelatedPathsT(i, j, Q, heuristic)$

Function relatedPathsT($i, j, Q, heuristic$).
1: **for** $k \leftarrow 1$ to n **do**
2: **if** *not* $skippingCondition(C_{jk})$ **then**
3: $heuristic.append(Q, (i, j, k))$
4: **if** *not* $skippingCondition(C_{ki})$ **then**
5: $heuristic.append(Q, (k, i, j))$
6: **done**

Despite these different complexities, WCC2_P and WCC2_T can perform worse than WCC1. This is mainly due to the fact that WCC2_P and especially WCC2_T must make an expensive initialization of the list Q (line 2). This step can take more time than the subsequent processing of the elements of the list, in particular for no consistent QCNs. This is why we introduce the functions WCCMixed_P and WCCMixed_T (see Algorithm 4 and Algorithm 5) to remedy this drawback. Roughly, these functions realize a first step corresponding to a first loop of WCC1 and then continues in the manner of WCC2_P and WCC2_T.

Algorithm 4

Function WCCMixed_P($\mathcal{N}$), with $\mathcal{N} = (V, C)$.
1: $Q \leftarrow \emptyset$
2: initMixedPair($\mathcal{N}, Q, heuristic$)
3: **while** $Q \neq \emptyset$ **do**
4: $(i, j) \leftarrow heuristic.next(Q)$
5: **for** $k \leftarrow 1$ to n **do**
6: **if** $revise(i, j, k)$ **then**
7: **if** $C_{ik} == \emptyset$ **then return false**
8: **else** $addRelatedPathsPair((i, k), Q, heuristic)$
9: **if** $revise(k, i, j)$ **then**
10: **if** $C_{kj} == \emptyset$ **then return false**
11: **else** $addRelatedPathsP(k, j, Q, heuristic)$
12: **done**
13: **end while**
14: **return true**

Function initMixedP($\mathcal{N}, Q, heuristic$).
1: $change \leftarrow$ **false**
2: **for** $i \leftarrow 1$ to n **do**
3: **for** $j \leftarrow i$ to n **do**
4: **for** $k \leftarrow 1$ to n **do**
5: **if** *not* $skippingCondition(C_{ik}, C_{kj})$
6: **if** $revise(i, k, j)$**then**
7: **if** $C_{ij} == \emptyset$ **then return false**
8: **else** $change \leftarrow$ **true**
9: **done**
10: **if** $(change)$ $addRelatedPathsP(i, j, Q, heuristic)$
11: **done**

4 Experimentation

4.1 Generated instances

To evaluate the performances of the proposed algorithms we randomly generate instances of qualitative constraint networks. A randomly generated QCN will be characterized by five parameters:

- an integer n which corresponds to the number of variables of the network;
- a qualitative calculus *algebra* which is the used qualitative calculus;
- a real $nonTrivialDensity$ which corresponds to the probality of a constraint to be a non trivial constraint (to be different of $\mathcal{B}$);
- a real $cardinalityDensity$ which is the probality of a basic relation to belong to a non trivial given constraint;
- a flag *type* which indicates if the generated network must be forced to be consistent by adding a consistent scenario.

Algorithm 5

Function WCCPCMixedTriple($\mathcal{N}$), with $\mathcal{N} = (V, C)$.
1: $Q \leftarrow \emptyset$
2: initMixedTriple($\mathcal{N}, Q, heuristic$)
3: **while** $Q \neq \emptyset$ **do**
4: $(i, k, j) \leftarrow heuristic.next(Q)$
5: **if** $revise(i, k, j)$ **then**
6: **if** $C_{ij} == \emptyset$ **then return false**
7: **else** $addRelatedPathsT(i, j, Q, heuristic)$
8: **end while**
9: **return true**

Function initMixedT($\mathcal{N}, Q, heuristic$).
1: $change \leftarrow$ **false**
2: **for** $i \leftarrow 1$ to n **do**
3: **for** $j \leftarrow i$ to n **do**
4: **for** $k \leftarrow 1$ to n **do**
5: **if** *not* $skippingCondition(C_{ik}, C_{kj})$
6: **if** $revise(i, k, j)$**then**
7: **if** $C_{ij} == \emptyset$ **then return false**
8: **else** $change \leftarrow$ **true**
9: **done**
10: **if** $(change)$ $addRelatedPathsT(i, j, Q, heuristic)$
11: **done**

The different algorithms have been implemented with the help of the JAVA library QAT[1]. We have conducted an extensive experimentation on a PC Pentium IV 2,4GHz 512mo under Linux. The experiences reported in this paper concern QCNs of the Allen's calculus generated with a *nonTrivialDensity* equals to 0.5 . Performances are measured in terms of the number of revise operations (numberOfRevises), in terms of cpu time (time), in terms of the number of maximum elements in the list (max).

4.2 Heuristics

Almost of the algorithms proposed in the later section use a list which contains the elements (pairs or triples) to be propagated. To improve the efficiency of the algorithms we have to reduce the number of these elements. When a constraint between (i,j) changes we must add all the elements which can be affected by this modification. The order that these elements are processed is very important and can reduce dramatically the number of triangulation operations. The set of the experimented heuristics contains the different heuristics proposed in [5]. The main task of a heuristic consists in the insertion of a pair or a triple of variables in the list after computing its location. If the pair or the triple is already in the list it can insert it or do nothing depending on its policy. All heuritics experimented remove and return the first element of the list. In general, given an heuritics, more it reduces the number of triangulation operations more its time cost and spatial cost are important.

4.3 Experimental results

Stack or Queue. The list used to stock the pairs/the triples can be handles as a stack or a queue, *i.e.* after the changing

[1] This library can be found at http://www.cril.univ-artois.fr/∼saade/.

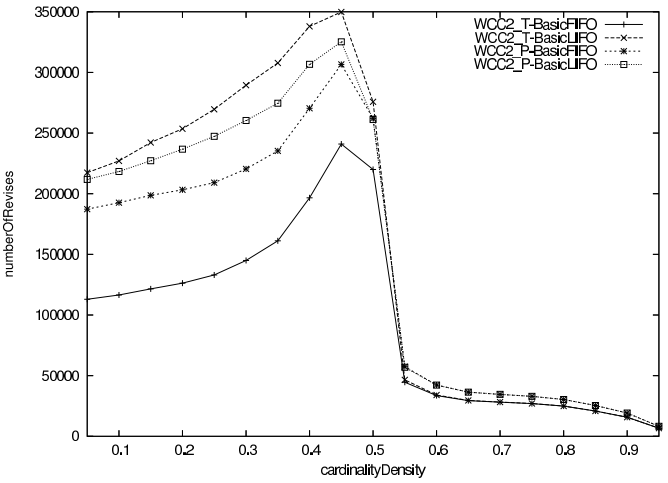


Figure 3. Average time for WCC2_P and WCC2_T using the heuristic *basic* on consistent QCNs (200 instances per data points, with $n = 50$)

of a constraint the pair or the triples corresponding can be added at the head of the list or at its queue (recall that the first element of the list is always treat firstly). After the initialisation of the list, the addition of a pair/a triple is due to the restriction of a constraint. Intuitively, more this constraint is added belatedly, more its cardinality is small and more it will be restrictive for an operation of triangulation. It is why, the using of the list as a stack (a FIFO structure) must perform the using of the list as a queue (a LILO structure). This is confirmed by our experiences, for example consider Figure 3 in which we use WCC2_P and WCC2_T on forced consistent networks with the heuristic *Basic*. Note that for WCCMixed_P and WCCMixed_T the difference is not so important. Actually *Basic* is not really a heuristic, indeed, it just adds an element in the list if it is not present and removes the first element of the list. In the sequel, among the elements which can be returned from the list, the heuristics always choose the more recently added (LIFO handling).

Add or not add a pair/a triple. The main task of *heuristic* is to add a pairs or a triples when a modification raises. As the pair/the triple is already in the list, depending of the used policy, heuristic can or cannot add the pair/the triple. Adding the element could have a prohibitive cost since one must remove the element of the list before add it at the new location. This cost is connected to the heuristic used and the structure used to implement the list. In our case, roughly speaking, we use doubly-linked lists or tables of doubly-linked lists for the more sophisticated heuristics. Moreover, we use tables with 2/3 entries to check the presence of a pair/a triple in the list. Actually, the experiences show that removing and adding the pair/the triple in the case where it is present avoid sufficient revise operations to be more competitive than the case where nothing is done. See for example Figure 4 which shows the behaviour of the heuristic cardinality with these two possible policies ($cardinalityMoving$ for the systematic addition and $cardinalityNoMoving$ for the addition in the case where the pair/the triple is not present). For

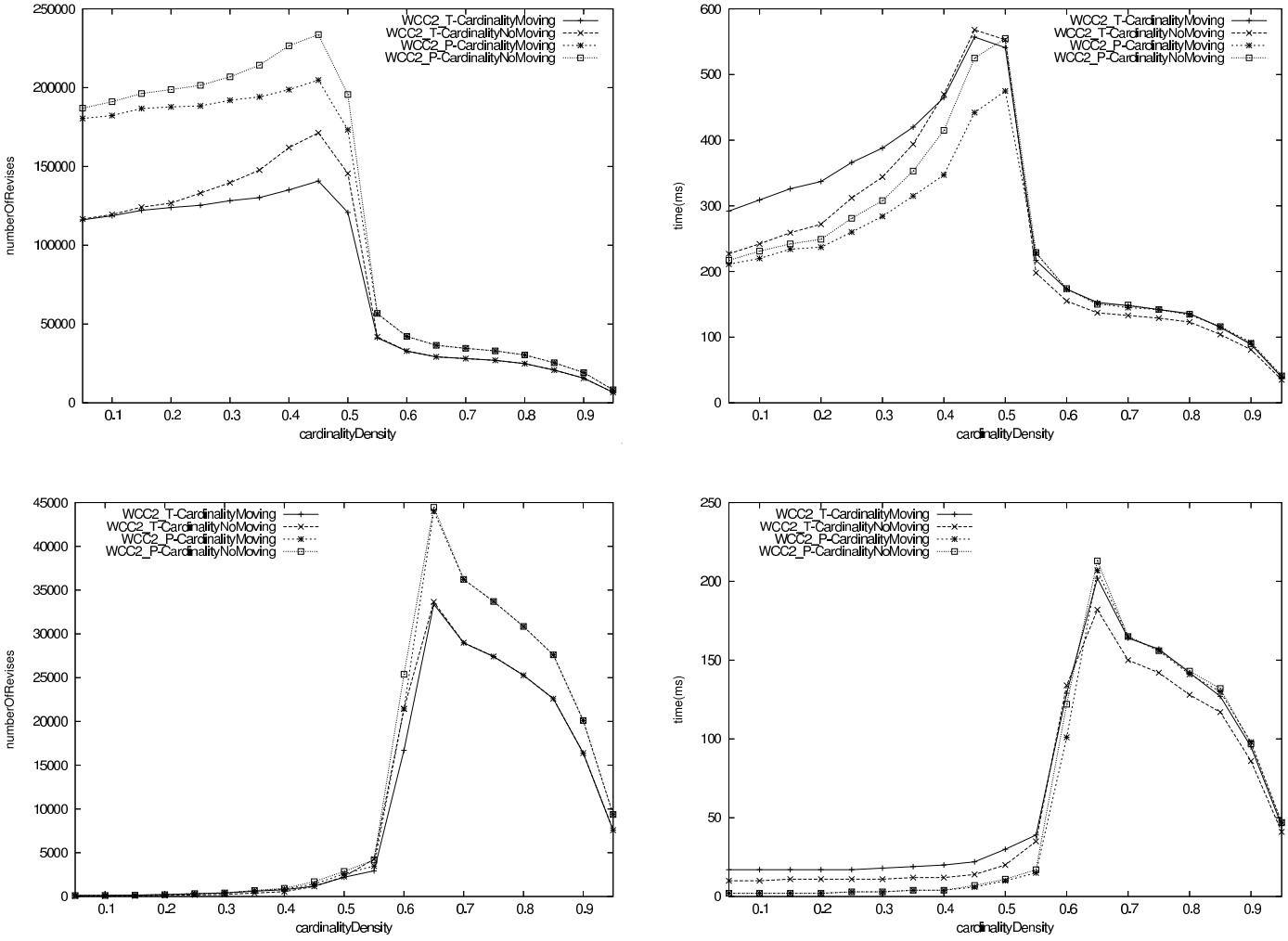


Figure 4. Average number of revises and average time for WCC2_P and WCC2_T using $cardinalityNoMoving$ and $cardinalityMoving$ on consistent (top) and no forcing consistent (bottom) QCNs (200 instances per data points, with $n = 50$)

the methods WCC2_P and WCC2_T $cardinalityMoving$ performs $cardinalityNoMoving$, in particular before the phase transition (cardinalityDensity between 0.3 and 0.55). Conerning the no forced consistent instances, from the fact that the numbers of revises are very near, we have $cardinalityMoving$ which is lightly better in term of time. For the mixed methods, $cardinalityMoving$ and $cardinalityNoMoving$ are very closed in term of time and number of revises. In the sequel we always use the policy which consists in systematically moving the present pair or triple.

The better heuristics. We compared all the heuristics on the different algorithms. Concerning the algorithms manipulating the pairs we compare the heuristics $Basic$ and $Cardinality$ previously presented. Moreover we used the $Weight$ heuristic, this heuristic processes the pair (i, j) following the weight of the constraint C_{ij} in ascending order. The weight of a constraint is the sum of the weights of the basic relations composing it. Intuitively, to obtain the weight of a basic relation B we sum the number of basic relations present in the table of composition at the line and the column corresponding to the entry B then we scaled the obtained numbers to give the value 1 at the basic relations with the smallest numbers, then 2, *etc.* This method is lightly different from this proposed by van Beek and Manchak [5] but it is easy to implement it for all qualitative calculi. For the basic relations of the Allen's calculus we obtain the weight 1 for eq, 2 for m, mi, s, si, f, fi, 3 for d, di, b, bi and 4 for o, oi. In addition to these heuristics, we define heuristics corresponding to combinations of $Cardinality$ and $Weight$: the $SumCardinalityWeight$ heuristic which arranges the pairs (i, j) following the sum of the cardinality and the weight of the constraint C_{ij}, the $CardinalityWeight$ heuristic which arranges the pairs (i, j) following the cardinality of C_{ij} and then, following the weight of C_{ij}, and $WeightCardinality$ which arranges the pairs (i, j) following the weight of C_{ij} and then, following the cardinality of C_{ij}. These heuristics

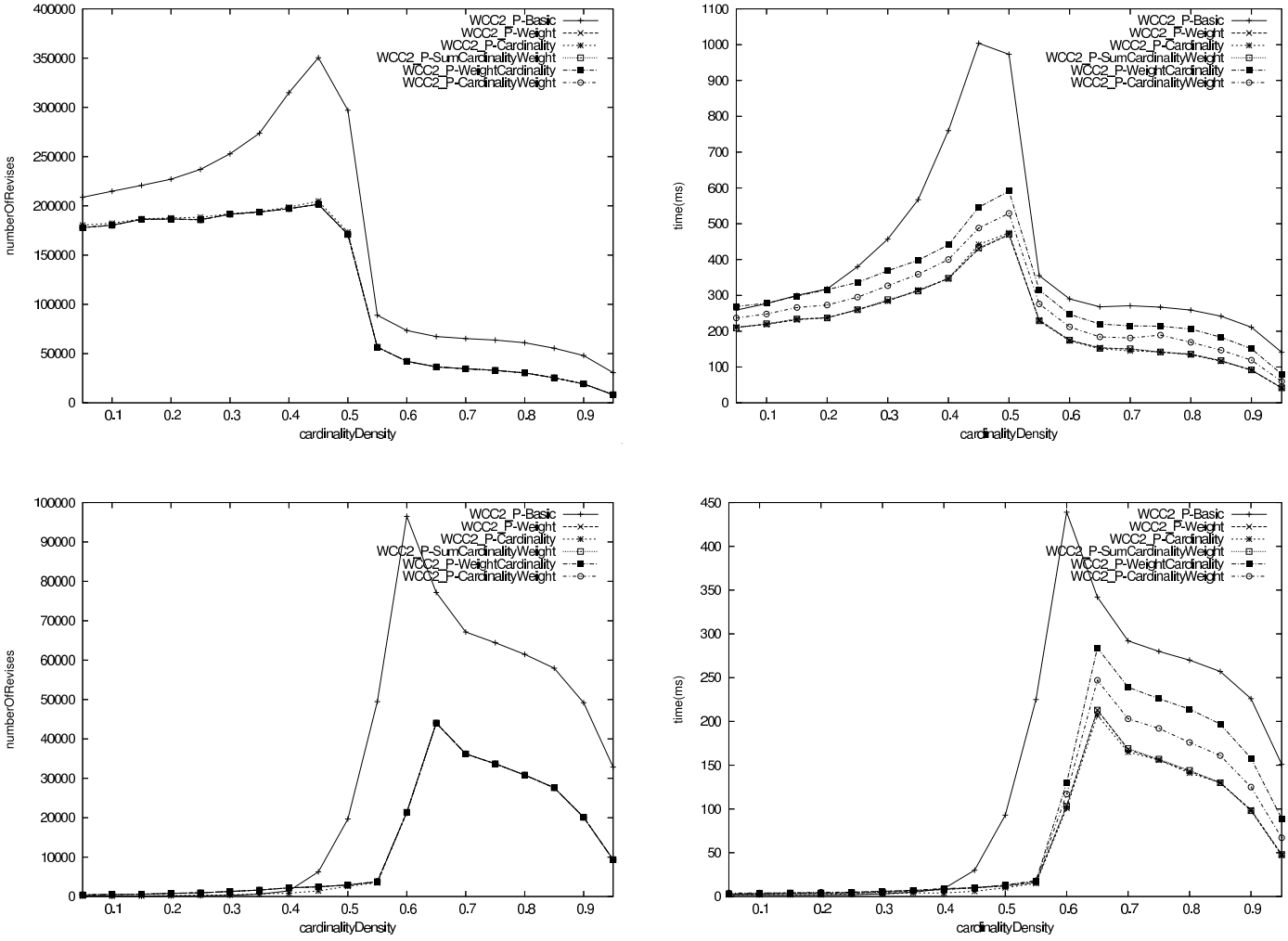


Figure 5. Average number of revises and average time for the heuristics used with WCC2_P on consistent (top) and no forcing consistent (bottom) QCNs (200 instances per data points, with $n = 50$)

are also define for for WCC2_T and WCCMixed_T which use triples instead of pairs. By examining Figure 5, we constate that the number of revises are very closed for all these heuritics (expected for the *Basic* heuristic). In term of cpu time, the heuristics $Cardinality$, $SumCardinalityWeight$ and $Weight$ are very closed and are the more performing heuristics. Due to the using of triples we can define finer heuristics. For example, from the heuristic *cardinality* we have three different heuristics: the *cardinalityI* heuristic which considers the cardinality of C_{ij} for the triples (i, j, k) and (k, i, j) (similarly to the previous *cardinality* heuristic), the *cardinalityII* heuristic which takes into account the sum of the cardinality of C_{ij} and the cardinality of C_{jk} for the triple (i, j, k), and the sum of the cardinality of C_{ij} and the cardinality of C_{ki} for the triple (k, i, j), the *cardinalityIII* heuristic which takes into account the sum of the cardinality of C_{ij}, the cardinality of C_{jk} and the cardinality of C_{ik} for the triple (i, j, k), the sum of the cardinality of C_{ij}, the cardinality of C_{ki} and the cardinality of C_{kj} for the triple (k, i, j). In a same line of reasoning we split the heuristics *weight* and $SumCardinalityWeight$ in six heuristics. By considering the different versions of the $Cardinality$ heuristic (it is the same thing for the *weight* and $SumCardinalityWeight$ heuristics) we can see that the *cardinalityII* heuritic makes the smallest number of revises. Outside the phase transition it performs the other triple *cardianality* heuritics in terms of time. In the phase transition the *cardinalityIII* heuristic performs the *cardinalityI* heuristic and the *cardinalityIII* heuristic. In terms of cpu time, the handling with pairs performs the handling with triples. WCC1/WCC2_P/WCCMixed_P/WCC2_T/WCCMixed_T Now we compare all the algorithms we the more competitive heuristics. We can constate that in general WCC1 is the algorithm the less competitive algorithm, see Figure 7. The most favorable case for WCC1 is the case where the instances are inconsistent QCNs. Generally, in particular for the forced con-

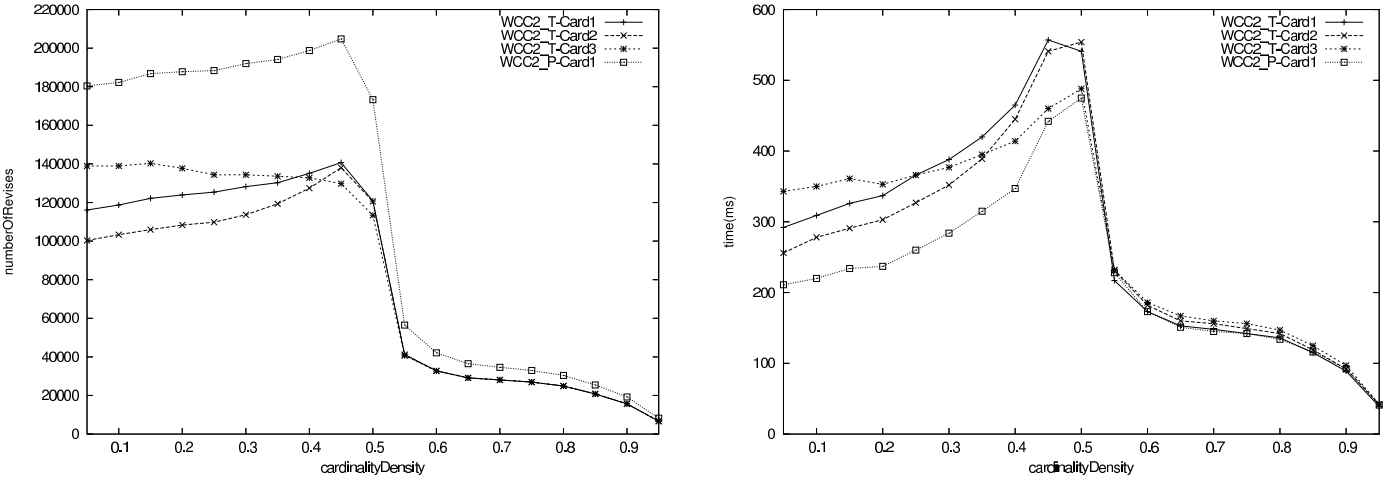


Figure 6. Average number of revises and average time for the different *cardinality* heuristics used with WCC2_T and WCCMixed_T on forced consistent QCNs (200 instances per data points, with $n = 50$)

sistent instances, the algorithms based on triples make less revise operations than the algorithms based on pairs. Despit it we can see that the last ones are more speed than the first ones. The reason is that the handling of triples is most cost than the handling of pairs in term of time. Moreover the number of elements which must be stocked is very important for the triples contrary to pair case (see the last figures of Figure 7). For the forced consistent instances we can see that the mixed versions of the algorithms are less performing than the no mixed versions, note that the difference is not very important. Concerning the no forced consistent instances we have the same result for the cardinality density comprise between 0.5 and 0.6. For the densities strictly greater than 0.6 we have an inverion and the mixed versions are more competitive. By examing the number corresponding to the maximum of elements in the list we can see that the mixed versions reduced dramatically this number for the triples.

5 Conclusions

In this paper we study empirically several algorithms enforcing the $\circ$-closure on qualitative constraint networks. The algorithms studied are adapted from the algorithms PC1 and PC2. Concerning the algorithms issued of PC2 we use different heuristics, in particular heuristics defined in [5] and we use structures saving pairs of constraints or structures saving triples of constraints. We showed that using triples reduces dramatically the number of revises compared with an handling with pairs. Despite it, the versions using pairs are more competitive in term of time. We introduced two algorithms mixing the algorithm PC1 and the algorithm PC2. These algorithms seem to be a good compromise between a PC1 version which consumes lot of time and a PC2 version which consumes lot of space. Currently, we continue our experimentations on QCNs with a larger size in term of variables and on other qualitative calculus (in particular on INDU which is based on 25 basic relations and the cyclic point algebra which is a ternary calculus).

REFERENCES

[1] J. F. Allen. An interval-based representation of temporal knowledge. In *Proc. of the Seventh Int. Joint Conf. on Artificial Intelligence (IJCAI'81)*, pages 221–226, 1981.

[2] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[3] P. Van Beek. Reasoning About Qualitative Temporal Information. *Artificial Intelligence*, 58(1-3):297–326, December 1992.

[4] Peter van Beek. Approximation algorithms for temporal reasoning. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1291–1296, Detroit, MI, USA, August 1989. Morgan Kaufmann.

[5] Peter Van Beek and Dennis W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4:1–18, 1996.

[6] Christian Bessire. A Simple Way to Improve Path Consistency Processing in Interval Algebra Networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, volume 1, pages 375–380, 1996.

[7] Amar Isli and Anthony G. Cohn. A new approach to cyclic ordering of 2D orientations using ternary relation algebras. *Artificial Intelligence*, 122(1–2):137–187, 2000.

[8] P. B Ladkin and A. Reinefeld. Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57(1):105–124, september 1992.

[9] Peter B. Ladkin and Alexander Reinefeld. A symbolic approach to interval constraint problems. In *AISMC*, pages 65–84, 1992.

[10] Gérard Ligozat. Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, 1(9):23–44, 1998.

[11] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence, 1977*, 8:99–118, 1977.

[12] A. K. Mackworth and E. C. Freuder. The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problem. *Artificial Intelligence*, 25(1):65–74, 1985.

[13] Itay Meiri. Combining qualitative and quantitative constraints in temporal reasoning. In Thomas Dean and Kathleen McKeown, editors, *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 260–267, Menlo Park, California, 1991. American Association for Artificial Intelligence, AAAI Press.

[14] U. Montanari. Networks of constraints: Fundamental properties and application to picture processing. *Information Sciences*, 7(2):95–132, 1974.

[15] Bernhard Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficienty of using the ord-horn class. In *Proceeding of the Twelfth Conference on Artificial Intelligence (ECAI'96)*, 1996.

[16] Arun K. Pujari, G. Vijaya Kumari, and Abdul Sattar. Indu: An interval and duration network. In *Australian Joint Conference on Artificial Intelligence*, pages 291–303, 1999.

[17] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proc. of the 3rd Conf. on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176, 1992.

[18] J. Renz and B. Nebel. Efficient methods for qualitative spatial reasoning. In *Proceedings of the workshop on spatial reasoning (ECAI'98)*, 1998.

[19] M. Vilain and H. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *Proc. of the Fifth Nat. Conf. on Art. Int. (AAAI'86)*, pages 377–382, 1986.

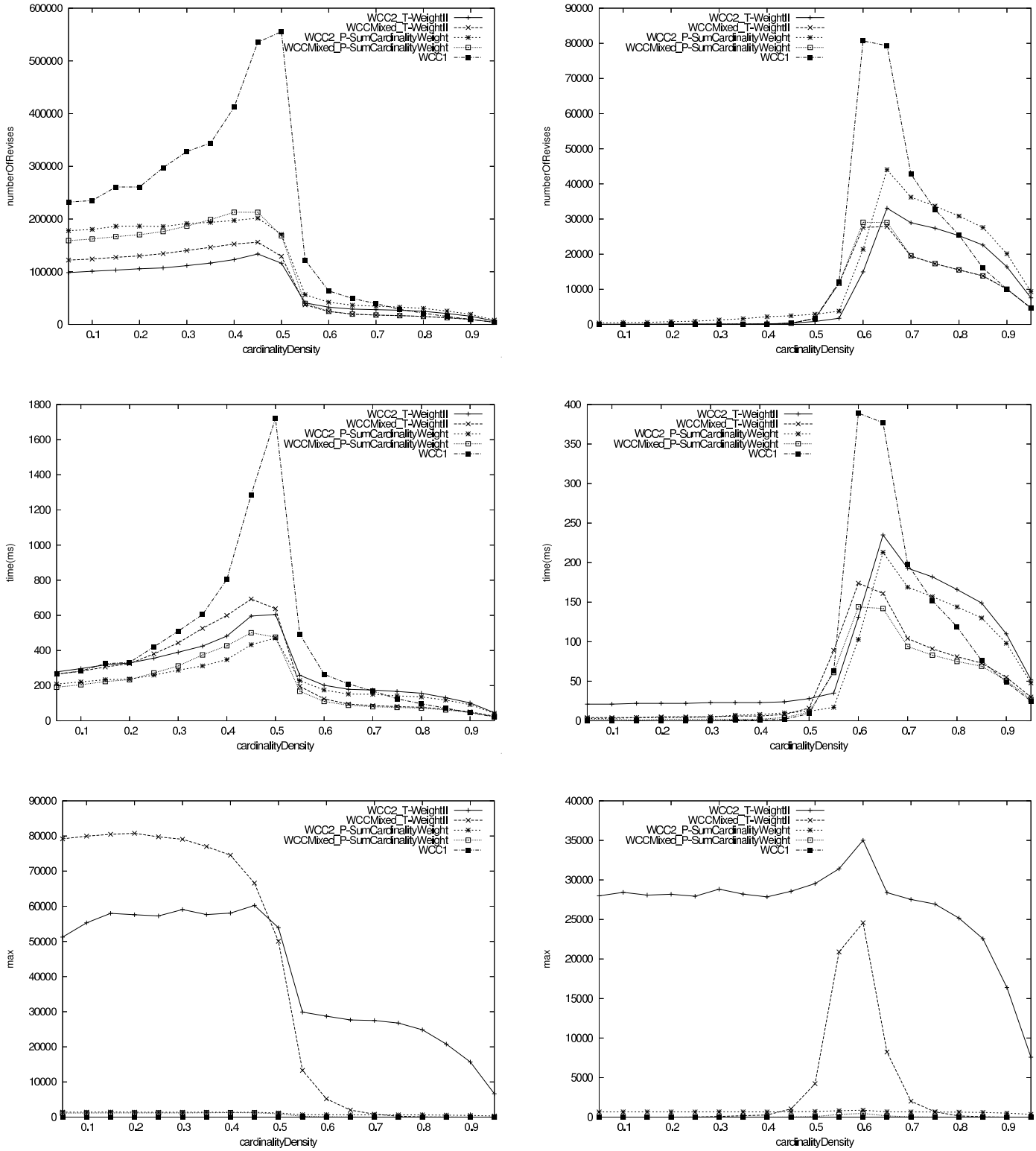


Figure 7. Average number of revises, average time and average maximum elements in the list for all algorithms with a competitive heuristic on consistent (left) and no forcing consistent (right) QCNs (200 instances per data points, with $n = 50$)