Embarrassingly Greedy Inconsistency Resolution of Qualitative Constraint Networks

Michael Sioutis
LIRMM UMR 5506, Université de Montpellier & CNRS, France

Abstract

In this paper, we deal with inconsistency resolution in qualitative constraint networks (QCNs). This type of networks allows one to represent and reason about spatial or temporal information in a natural, human-like manner, e.g., by expressing relations of the form x {*is north of* $\vee$ *is east of*} y. On the other hand, inconsistency resolution involves maximizing the amount of information that is consistent in a knowledge base; in the context of QCNs, this translates to maximizing the number of constraints that can be satisfied, via obtaining a qualitative solution (scenario) of the QCN that ignores/violates as few of the original constraints as possible. To this end, we present two novel approaches: a greedy constraint-based and an optimal Partial MaxSAT-based one, with a focus on the former due to its simplicity. Specifically, the greedy technique consists in adding the constraints of a QCN to a new, initially empty network, one by one, all the while filtering out the ones that fail the satisfiability check. What makes or breaks this technique is the ordering in which the constraints will be processed to saturate the empty QCN, and for that purpose we use many different strategies to form a portfolio-style implementation. The Partial MaxSAT-based approach is powered by Horn theory-based maximal tractable subsets of relations. Finally, we compare the greedy approach with the optimal one, commenting on the trade-off between obtaining repairs that are optimal and obtaining repairs in a manner that is fast, and make our source code available for anyone to use.

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

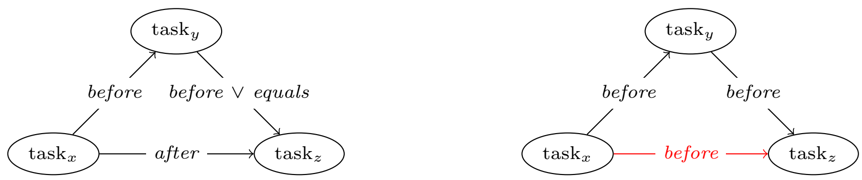
Keywords and phrases Spatial and Temporal Reasoning, Qualitative Constraints, Inconsistency Resolution, Maximizing Satisfiability, Greedy Algorithm, Partial MaxSAT Solver

Digital Object Identifier 10.4230/LIPIcs.TIME.2023.11

Funding *Michael Sioutis*: The work was partially funded by the Agence Nationale de la Recherche (ANR) for the "Hybrid AI" project that is tied to the chair of Dr. Sioutis, and the I-SITE program of excellence of Université de Montpellier that complements the ANR funding.

1 Introduction

Qualitative Spatio-Temporal Reasoning (QSTR) is a rich symbolic AI framework that deals with representing and reasoning about abstract, qualitative spatio-temporal information [8, 15]. Specifically, QSTR allows one to spatially or temporally relate one object with another object or oneself by using everyday, human-like natural language descriptions, and perform reasoning with those descriptions; as an example, consider a relation of the form x {*is north of* $\vee$ *is east of*} y, which abstracts from numerical information and yet is very intuitive. Such QSTR descriptions or relations, and disjunctions thereof, can be modeled as a qualitative constraint network (QCN), a simplified example of which is provided in Figure 1a. Spatial or temporal information in the QSTR framework can, in general, pertain to any spatial or temporal aspects in the physical world. However, the literature has been deeply invested in point/interval-based calculi, with Allen's Interval Algebra being the most representative example [1], as intervals can be used to represent and reason about anything from durative actions in planning or tasks in robotics [18] to temporal abstractions in multivariate time series classification [17], among other applications; the interested reader is invited to explore the discussion in [26, 28, 9, 3].



(a) An inconsistent plan as a simplified QCN. **(b)** An *optimal* scenario of the simplified QCN.

Figure 1 An illustration of the MAX-QCN problem of a qualitative constraint network (QCN) [6] and the terminology used here; the QCN in Figure 1a is inconsistent, and one solution of the MAX-QCN problem, viz., an *optimal* scenario, is depicted in Figure 1b, where task$_x$ $\{$*before*$\}$ task$_z$ is the only relation that does not satisfy the respective constraint in Figure 1a..

Context & Motivation

In this paper, we focus on the problem of maximizing satisfiability in a qualitative constraint network, formally called the MAX-QCN problem [6]. Specifically, given a QCN $\mathcal{N}$, the MAX-QCN problem is the problem of obtaining a spatial or temporal configuration that maximizes the number of satisfied constraints in $\mathcal{N}$; see also Figure 1 for an example. The motivation behind studying this problem lies in the fact that representing spatial or temporal information may inevitably lead to inconsistencies, due to e.g. human error and/or inaccurate classifiers. As illustration, timetabling is an instance of scheduling where inconsistencies can naturally form due to the lack of resources for certain tasks, among other reasons [14]. Specifically, in timetabling the goal is to associate temporal intervals with a number of tasks requiring limited resources. In the context of a hospital, for example, an inconsistency can occur when two surgeons are allocated the same operating room in overlapping temporal intervals; the inconsistency must then be repaired by considering available temporal intervals and preferences alike, and minimizing changes so as to perturb the structure of the timetable as little as possible. In the broader context of neuro-symbolic AI architectures [13], classifiers may construct inconsistent spatio-temporal knowledge bases due to inaccurate predictions, and minimizing inconsistency (i.e., maximizing satisfiability) is an essential step of logical abduction (or other type of reasoning) in the neuro-symbolic cycle, see, e.g., Figure 1 in [31].

State of the Art & Contribution

The state of the art in solving the MAX-QCN problem with respect to *constraints* and *SAT encodings* consists of the works in [6] and in [7], respectively. Specifically, both of these approaches try to obtain a refinement of the input QCN that maximizes the number of satisfied constraints in the QCN. In doing so, they are trying to solve two problems of different nature at the same time: extracting a scenario of the QCN, whilst ensuring that the extracted scenario is optimal. This is particularly crippling for the performance of the constraint-based approach in [6], as, should the constraint not be part of an optimal scenario in the end, taking a refinement of it in the beginning might create a huge branch in the search tree that is useless to explore. The clause learning of the SAT-based approach in [7] circumvents this issue, but, on the other hand, [7] does not exploit tractability properties for QCNs, viz., *Horn theories* and/or *maximal tractable subsets* of relations [22]; nevertheless, it significantly outperforms [6]. Here, with respect to the previous discussion, we make the following contributions:

(i) We offer a greedy constraint-based approach for tackling the MAX-QCN problem that treats the constraints of the input QCN in whole and, hence, may avoid—to a relatively

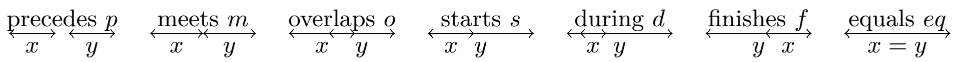


Figure 2 A representation of the 13 base relations b of IA, each one relating two potential intervals x and y as in $x\ b\ y$; the converse of b, i.e., b^{-1}, can be denoted by bi and is omitted in the figure.

greater extent—redundant exploration of search space;

(ii) We introduce one of the most compact to date Partial MaxSAT encodings for the MAX-QCN problem by extending the SAT encoding of [20] (see also [30]), fully utilizing tractability properties (alongside chordal completions of the constraint graphs of QCNs);

(iii) We pit the two approaches against each other in an experimental evaluation, and comment on the trade-off between obtaining repairs in an inconsistent QCN in a way that is optimal and coming close to a solution of the MAX-QCN problem in a manner that is fast, making our source code available for any interested researcher to use.

Organization

The rest of the paper is organized as follows. In Section 2 we provide definitions and notations regarding QSTR and the MAX-QCN problem that are necessary for following and understanding the paper. Then, Sections 3–5 expand on the contribution points (i)–(iii), respectively, that were listed earlier. Finally, in Section 6 we conclude and give some directions for future work.

2 Preliminaries

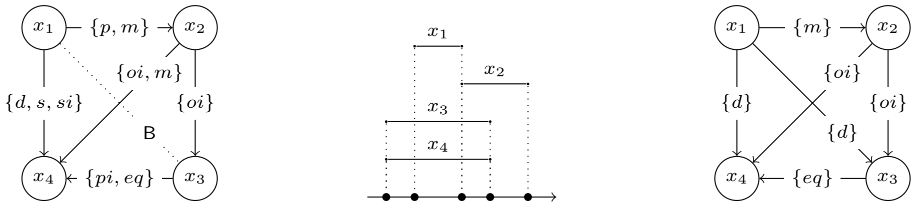
A binary qualitative spatial or temporal constraint language is based on a finite set B of *jointly exhaustive and pairwise disjoint* relations, called *base relations* [15] and defined over an infinite domain D (e.g., $\mathbb{R}$). The base relations of a particular qualitative constraint language can be used to represent the definite knowledge between any two of its entities with respect to the level of granularity provided by the domain D. The set B contains the identity relation Id, and is closed under the *converse* operation $({}^{-1})$. Indefinite knowledge can be specified by a union of possible base relations, and is represented by the set containing them. Hence, 2^{B} represents the total set of relations. The set 2^{B} is equipped with the usual set-theoretic operations of union and intersection, the converse operation, and the *weak composition* operation denoted by the symbol $\diamond$ [15]. For all $r \in 2^{\mathsf{B}}$, we have that $r^{-1} = \bigcup\{b^{-1} \mid b \in r\}$. The weak composition $(\diamond)$ of two base relations $b, b' \in \mathsf{B}$ is defined as the smallest (i.e., most restrictive) relation $r \in 2^{\mathsf{B}}$ that includes $b \circ b'$, or, formally, $b \diamond b' = \{b'' \in \mathsf{B} \mid b'' \cap (b \circ b') \neq \emptyset\}$, where $b \circ b' = \{(x, y) \in \mathsf{D} \times \mathsf{D} \mid \exists z \in \mathsf{D}$ such that $(x, z) \in b \wedge (z, y) \in b'\}$ is the (true) composition of b and b'. For all $r, r' \in 2^{\mathsf{B}}$, we have that $r \diamond r' = \bigcup\{b \diamond b' \mid b \in r, b' \in r'\}$.

As illustration, consider the well-known qualitative temporal constraint language of Interval Algebra (IA) [1]. IA considers time intervals on the real line, and the set of base relations $\mathsf{B} = \{eq\ (= \mathsf{Id}), p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$ to encode knowledge about the temporal relations between such intervals, as described in Figure 2.

Representing and reasoning about qualitative spatio-temporal information pertaining to a set of base relations B can be facilitated by a *qualitative constraint network* (QCN):

▶ **Definition 1.** *A* qualitative constraint network (QCN) *is a tuple* (V, C) *where:*

- $V = \{v_1, \ldots, v_n\}$ *is a non-empty finite set of variables (representing entities in* D*);*
- *and* C *is a mapping* $C : V \times V \to 2^{\mathsf{B}}$ *such that,* $\forall v \in V$, $C(v, v) = \{\mathsf{Id}\}$*, and,* $\forall v, v' \in V$, $C(v, v') = (C(v', v))^{-1}$.



(a) A satisfiable QCN $\mathcal{N}$. **(b)** A solution σ of $\mathcal{N}$. **(c)** A scenario $\mathcal{S}$ of $\mathcal{N}$.

Figure 3 Figurative examples of QCN terminology using Interval Algebra (IA).

An example QCN of IA is shown in Figure 3a; for conciseness, converse relations or Id loops are not shown in the figure.

▶ **Definition 2.** *Let $\mathcal{N} = (V, C)$ be a* QCN *(Figure 3a), then:*

- *a* solution *of $\mathcal{N}$ is a mapping $\sigma : V \to$ D such that, $\forall(u, v) \in V \times V$, $\exists b \in C(u, v)$ such that $(\sigma(u), \sigma(v)) \in b$; and $\mathcal{N}$ is* satisfiable *iff it admits a solution (see Figure 3b);*
- *a* sub-QCN *(also known as* refinement*) $\mathcal{N}'$ of $\mathcal{N}$, denoted by $\mathcal{N}' \subseteq \mathcal{N}$, is a* QCN (V, C') *such that, $\forall u, v \in V$, $C'(u, v) \subseteq C(u, v)$;*
- *$\mathcal{N}$ is* atomic *iff, $\forall v, v' \in V$, $C(v, v')$ is a* singleton relation*, i.e., a relation $\{b\}$ with $b \in$* B*;*
- *a* scenario *$\mathcal{S}$ of $\mathcal{N}$ is an atomic satisfiable sub-*QCN *of $\mathcal{N}$ (see Figure 3c);*
- *the* constraint graph *of $\mathcal{N}$, denoted by* G($\mathcal{N}$)*, is the graph (V, E) where $\{u, v\} \in E$ iff $C(u, v) \neq$ B and $u \neq v$;*
- *for $V' \subseteq V$, $\mathcal{N}{\downarrow_{V'}}$ denotes $\mathcal{N}$ restricted to V';*
- *$\mathcal{N}$ is denoted by $\mathcal{N}_\top$ when each of its constraints is* universal*, i.e., iff, $\forall v, v' \in V$ with $v \neq v'$, $C(v, v') =$ B.*

The MAX-QCN problem

The MAX-QCN problem has been introduced in the context of QSTR in [6]. Given a QCN $\mathcal{N}$ over a set of variables V, the MAX-QCN problem is the problem of finding a scenario over V that maximizes the number of satisfied constraints in $\mathcal{N}$, or, equivalently, the problem of finding a scenario over V that minimizes the number of unsatisfied constraints in $\mathcal{N}$. Such scenarios are called *optimal* scenarios of $\mathcal{N}$. Clearly, if a QCN $\mathcal{N}$ is satisfiable, any scenario of $\mathcal{N}$ is also an optimal scenario of $\mathcal{N}$. The reader is kindly asked to revisit Figure 1 in the introduction for a simplified example of the MAX-QCN problem and a solution of it. Solving the MAX-QCN problem is clearly at least as difficult as solving the satisfiability checking problem of a QCN, which is NP-hard in general for most calculi [8].

3 Greedy Constraint-based Approach

In this section, we present a greedy approach to come close to, or even exactly identify, a maximum satisfiable subset of constraints of an original input QCN $\mathcal{N} = (V, C)$ and, hence, tackle the MAX-QCN problem. This approach is presented in Algorithm 1, and it consists in consistently saturating a universal QCN (lines 4–13) with as many constraints as possible from $\mathcal{N}$, by using and iterating various different orderings of the constraints of $\mathcal{N}$ (line 5). Given a QCN $\mathcal{N} = (V, C)$, with $E = E(\mathsf{G}(\mathcal{N}))$ denoting the set of edges in its constraint graph, GREEDUS runs in $O(|E| \cdot \beta)$ time, where β is the runtime of a SAT oracle call. The SAT oracle here can be any solver that can solve the satisfiability checking problem of a QCN, be it SAT- or qualitative constraint-based; in our implementation of the algorithm,

Algorithm 1 GREEDUS($\mathcal{N}, \mathcal{A}$)

in : A QCN $\mathcal{N} = (V, C)$ and a set $\mathcal{A}$ of bijections $\alpha : E \rightarrow \{0, 1, \ldots, |E| - 1\}$, where $E = E(\mathsf{G}(\mathcal{N}))$ (i.e., roughly, a set of orderings of the constraints in $\mathcal{N}$)
out : A subset $p \subseteq E(\mathsf{G}(\mathcal{N}))$ corresponding to feasible constraints in $\mathcal{N}$

1 $P \leftarrow \emptyset$;
2 **foreach** $\alpha \in \mathcal{A}$ **do**
3     $p \leftarrow \emptyset$;
4     $\mathcal{N}' = (V, C') \leftarrow \mathcal{N}_{\top}$;
5   **for** i **from** 0 **to** $|E(\mathsf{G}(\mathcal{N}))| - 1$ **do**
6         $\{u, v\} \leftarrow \alpha^{-1}(i)$;
7         $C'(u, v) \leftarrow C(u, v)$;
8         $C'(v, u) \leftarrow C(v, u)$;
9   **if** $SAT(\mathcal{N}')$ **then**
10             $p \leftarrow p \cup \{\{u, v\}\}$;
11   **else**
12             $C'(u, v) \leftarrow \mathsf{B}$;
13             $C'(v, u) \leftarrow \mathsf{B}$;
14     $P \leftarrow P \cup \{p\}$;
15 **return** $p \in \arg\max_{p' \in P}(|p'|)$;

we opted for a qualitative constraint-based one, since it made the implementation of the algorithm more straightforward. Of course, we assume here that the size of the set $\mathcal{A}$ of some orderings of the constraints in $\mathcal{N}$ is upper bounded by a small constant k that is equal to the number of different strategies that will be used to obtain these orderings in the first place (a discussion on such strategies follows immediately after); this would be naturally the case, as exploring all possible orderings would defeat the purpose of being greedy. Finally, it is important to know that each iteration of the loop in line 5 can be run in parallel, as the calculation of a satisfiable subset of constraints p by the end of an iteration is completely independent to any other such p; in the end, the largest such p is returned. However, in our implementation we maintained the sequential nature of the algorithm.

Constraint Ordering Strategies

Given a QCN $\mathcal{N} = (V, C)$, the effectiveness of GREEDUS relies heavily on the set $\mathcal{A}$ of some orderings of the constraints in $\mathcal{N}$ that will be provided as part of its input, as this set has a direct effect on the quality of the satisfiable subset of constraints that will be obtained in the end. It is worth noting that the efficiency of GREEDUS does not rely all that much on $\mathcal{A}$, as the algorithm will go through all constraints anyway (of course, in the sequential version of the algorithm, some optimizations can be achieved by passing information from one iteration to the next one, to early stop the loop, for example).

Intuition: We would like to delay the encounter of a constraint that causes inconsistency (line 9 in the algorithm) for as long as possible, as this should allow us to maximize the size of the set of satisfiable constraints. So, intuitively, we should order constraints from *more permissive* to *less permissive*, as this should increase our chances of a relatively more successful outcome. In the sequel, we list ways to assess the permissiveness of a constraint.

In qualitative constraint-based reasoning, the satisfiability checking of a QCN is done via the use of a backtracking algorithm [22], where the selection of the next constraint to process follows the *minimum remaining values* principle in traditional constraint programming [23] (commonly known as MRV); specifically, heuristics are used to select the more restrictive constraints first, as this should help the algorithm to explore a relatively sparser search tree. Here, we simply reverse the use of such constraint selection heuristics, making small adaptations where necessary, which we explain in what follows.

In sum, among other heuristics, we use the *local model* counting-based heuristics of [25], as well as the weighting-based ones of [27, 21], to order the constraints from more permissive to less permissive (or, equivalently, from less restrictive to more restrictive).

First, we need to recall and slightly adapt the definition of a local model from [25].

▶ **Definition 3** (local model, cf. [25]). *Given a* QCN $\mathcal{N} = (V, C)$ *and an edge* $\{v, v'\} \in E(\mathsf{G}(\mathcal{N}))$*, a* local model *of a base relation* $b \in C(v, v')$ *is a scenario* $S = (V', C')$ *of* $\mathcal{N}\downarrow_{V'}$*, where* $V' = \{v, v', u\}$ *with* $u \in V$ *(*V' *is a triple of variables in* V*), and* $C'(v, v') = \{b\}$*.*

Now, we are ready to list all of the used constraint ordering strategies in this work. It is clear that, given a QCN $\mathcal{N} = (V, C)$, an exhaustive application of either of the following strategies for each of the (non-universal) constraints of $\mathcal{N}$ provides an ordering of the constraints of $\mathcal{N}$; we can then represent those orderings with bijections $E \rightarrow \{0, 1, \ldots, |E|-1\}$, where $E = E(\mathsf{G}(\mathcal{N}))$, and form the required set of orderings for Greedus.

- **max**: choose the constraint that contains the base relation with the most local models.
- **min**: choose the constraint for which the base relation with the fewest local models has the most local models compared to such base relations of the rest of the constraints.
- **avg**: choose the constraint with the highest average count of local models (i.e., each of its base relations contributes a count and we take the average of these counts).
- **sum**: choose the constraint with the highest cumulative count of local models. (i.e., each of its base relations contributes a count and we take the sum of these counts).
- **weight**: choose the constraint with the largest weight; see, e.g., Figure 9 in [27] (the larger the weight, the more permissive the constraint).
- **card**: choose the constraint whose smallest decomposition into sub-relations of a (maximal) tractable subset $\mathcal{S} \in 2^{\mathsf{B}}$ [21] (e.g., the ORD-Horn set for IA [20]) is the largest one.
- **card + weight**: the card heuristic, with the weight heuristic acting as tie-breaker (this is very typical in the literature e.g., [21]).
- **random**: choose a constraint randomly.

The reader can note that the aforementioned strategies are very different to one another, even contradictory at times (e.g., max and min). In fact, such a mix of different strategies ensures that our portfolio-style approach is diverse enough; diversity is an important aspect of any portfolio-based method.

4 Optimal Partial MaxSAT-based Approach

In this section, we introduce a Partial MaxSAT encoding for the MAX-QCN problem by extending the SAT encoding of [20]; we note that the aforementioned encoding pertains to the IA calculus, but the approach itself may be adapted to any calculus by using the hard clauses to encode a theory of the calculus and the soft ones to encode the constraints of an input QCN over that calculus—e.g., a similar encoding exists for RCC8 in [29]. It must be noted that, contrary to the approach of [7], which does not take into account a theory of a calculus and aims to provide a generic approach that is based solely on the weak composition rules of that calculus, our extension may take full advantage of tractability properties for QCNs, viz., Horn theory-based *maximal tractable subsets* of relations [22], and is thus one of the most compact encodings for the MAX-QCN problem to date, see also Table 1 in [30].

First, we briefly introduce some notions about the Partial MaxSAT problem. A literal is a propositional variable or its negation, and a clause is a disjunction of literals. The maximum satisfiability problem (MaxSAT) is the problem of finding an assignment that satisfies as many clauses of a given set of clauses as possible [12]. Hence, the MAX-QCN problem can already be viewed as a version of the MaxSAT problem for QCNs. The Partial MaxSAT problem is an extension of the MaxSAT problem defined as follows: an instance Ω of Partial MaxSAT [16, 5] is a set of clauses composed of hard and soft clauses, and a solution ω of Ω is an assignment that satisfies the hard clauses and maximizes the number of satisfied soft clauses. For the MAX-QCN problem, certain hard clauses are necessary to ensure the completeness of the approach, in particular, the clauses that pertain to a provided theory of a given calculus, as we will demonstrate in the sequel.

We first introduce our Partial MaxSAT encoding for a given QCN in an abstract way, and then give an example based on IA and the SAT encoding in [20]. Given a QCN $\mathcal{N} = (V, C)$ over some calculus $\mathcal{C}$, the hard clauses in the Partial MaxSAT encoding are the ones encoding a theory of $\mathcal{C}$, the set of these clauses being denoted by $\mathsf{Th}_{\mathcal{C}}(\mathcal{N})$, and the soft clauses in the Partial MaxSAT encoding are the ones encoding the constraints of $\mathcal{N}$, the set of these clauses being denoted by $\mathsf{In}_{\mathcal{C}}(\mathcal{N})$. Specifically, regarding $\mathsf{In}_{\mathcal{C}}(\mathcal{N})$, the soft clauses can be viewed as follows (an explanation of the symbols follows immediately after):

$$\bigwedge_{(i,j)\in E(\mathsf{G}(\mathcal{N}))\text{ s.t. } i<j} (r_{ij} \rightarrow \bigwedge_{l=1}^{m} c_l) \tag{1}$$

With respect to Equation (1) above, r_{ij} is an auxiliary variable associated with every $(i, j) \in E(\mathsf{G}(\mathcal{N}))$ s.t. $i < j$, and complementing every clause c_l of a CNF formula $c_1 \wedge c_2 \wedge \ldots \wedge c_m$ corresponding to the constraint $C(i, j)$ (here, m is some small constant that is particular to the CNF encoding of a constraint in a given calculus). The soft part in Equation (1) is simply the set of these r_{ij} unit clauses: maximizing the number of satisfied clauses of the form r_{ij} corresponds to maximizing the number of satisfied constraints of the form $C(i, j)$.

Let us ground the presentation so far in IA to facilitate the reader. A Horn theory of IA can be based on that of partial orders, as is done in [20]. We present this theory as follows:

$$\begin{array}{cc} x \leq z \wedge z \leq y \rightarrow x \leq y & x = y \rightarrow x \leq y \\ x \leq y \wedge y \leq x \rightarrow x = y & x = y \rightarrow y \leq x \\ x = y \wedge x \neq y \rightarrow \bot & x \neq x \rightarrow \bot \end{array}$$

Then, we consider the usual domain D of IA, which is defined as the set of intervals on the real line, i.e., $\mathsf{D} = \{x = (x^-, x^+) \in \mathbb{R} \times \mathbb{R} \mid x^- < x^+\}$, where x^- and x^+ denote the starting point and ending point of an interval x, respectively.

Given a QCN $\mathcal{N} = (V, C)$ over IA, every interval variable $x \in V$ can be translated with regard to the theory of partial orders as follows (remember that, $\forall x \in V$, $x^- < x^+$):

$$x^- \leq x^+ \wedge x^- \neq x^+$$

In addition, for all distinct interval variables $x, y, z \in V$, we need to enforce the theory of partial orders mentioned earlier and obtain the respective translations for all of their starting and ending points (with respect to a chordal completion of $E(\mathsf{G}(\mathcal{N}))$).

The hard clauses of $\mathsf{Th}_{\mathrm{IA}}(\mathcal{N})$ can then be straightforwardly obtained by associating, for all $s \in \{-,+\} \times \{\leq,=\} \times \{-,+\}$, the propositional variables p^s_{xy} with every pair of interval variables $x, y \in V$, and retrieving the SAT enconding of the aforementioned translations.

For example the formula corresponding to an interval variable (viewed within the theory of partial orders as above, viz., $x^- \leq x^+ \wedge x^- \neq x^+$) is as follows:

$$p_{xx}^{(-,\leq,+)} \wedge \neg p_{xx}^{(-,=,+)}$$

With respect to the soft clauses of $\mathsf{In}_{\mathrm{IA}}(\mathcal{N})$, and the SAT encoding of the constraints in particular, it can be easily obtained by considering the definition of each base relation of IA with respect to the starting and ending points of two intervals, and its subsequent translation with regard to the theory of partial orders. For example, the base relation *during* between two intervals x and y is defined as $\{(x, y) \in \mathsf{D} \times \mathsf{D} \mid y^- < x^- \wedge x^+ < y^+\}$; we already saw earlier how $<$ corresponds to $\leq \wedge \neq$ with regard to the theory of partial orders, so the translation is obvious. By extension, the SAT encoding of composite relations (disjunctions of base relations) can be obtained via the disjunction of the SAT encodings of the base relations in the composite relation (which can then be transformed to CNF).

5 Experimentation

In this section, with respect to tackling the MAX-QCN problem, we perform an experimental evaluation between and in-house implementation of GREEDUS introduced in Section 3 (Algorithm 1), and an implementation of the Partial PaxSAT encoding introduced in Section 4 using the PySAT toolkit [10] and the RC2 MaxSAT solver offering there [11].

▶ Note 4. All the code is available at: `https://msioutis.gitlab.io/software/`

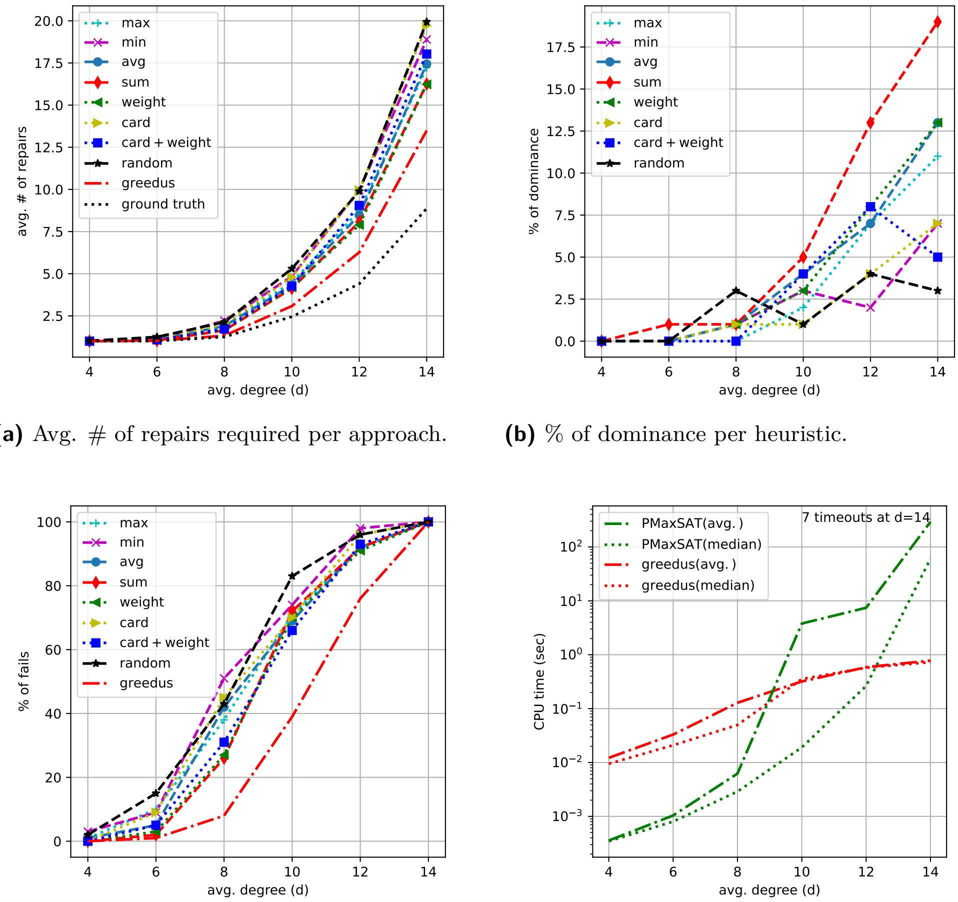
Dataset & Setup

We kept the dataset consistent with what has been used in previous works on the MAX-QCN problem for comparability, cf. [6, 7]. Specifically, we considered IA network instances generated by the standard $\mathsf{A}(n, d, l)$ model [21], used extensively in the literature. In short, $\mathsf{A}(n, d, l)$ creates network instances of size n, average constraint graph degree d, and an average number l of base relations per constraint. We set $n = 20$ and $l = 6.5$, and we considered 100 inconsistent network instances for *each* degree d between 4 and 14 with a 2-degree step; hence, 600 network instances in total. For this range of degrees d, the network instances of model $\mathsf{A}(n, d, l)$ lie within the *phase transition* region [19]. Again, the nature and size of the network instances is consistent with what has been used in the literature for the MAX-QCN problem in order to present results that are comparable and as complete as possible, cf. [6, 7] (see also the number of timeouts in Figure 4d for the dense instances). For the experiments we used an Intel® Core™ CPU i7-12700H @ 4.70GHz, 16 GB of RAM, and the Ubuntu Linux 22.04 LTS OS, and one CPU core per network. All coding/running was done in Python 3; however, we must note that the implementation of GREEDUS was sped up with PyPy,[1] which comes bundled with a just-in-time compiler, whereas the same is not possible for the implementation of the Partial MaxSAT encoding, because the RC2 MaxSAT solver in PySAT uses Glucose 3 [2] as the underlying SAT oracle, which is coded in C/C++.

Results & Remarks

All of the experimental results are concisely presented in Figure 4. In Figure 4a we evaluate how the different strategies that are implemented under the hood of GREEDUS behave

[1] `https://www.pypy.org/`



(a) Avg. # of repairs required per approach. **(b)** % of dominance per heuristic.

(c) % of fails to find optimal value per approach. **(d)** Runtime performance of main approaches.

Figure 4 Assessing the performance of an implementation of GREEDUS and of our Partial MaxSAT encoding, respectively, with Interval Algebra (IA) network instances of model $\mathsf{A}(n = 20, d, l = 6.5)$ [21]; a timeout occurs after $3\,600s$, and in that case the runtime up to that point is not taken into account (only 7 such timeouts occured, all for the Partial MaxSAT-based implementation at $d = 14$).

with respect to obtaining repairs in an inconsistent QCN if they are run standalone (see Section 3 for a description of these strategies), and how they define the respective behaviour of GREEDUS when taken all together; the ground truth here is the optimal value. The best performing strategies with respect to obtaining few repairs are sum and weight, and the worst performing one is random; however, as we will see in the sequel, no strategy goes to waste in this portfolio-style implementation. With respect to our last point, in Figure 4b we observe the percentage of times that a strategy *dominated* all others, where by "dominated" we mean that the strategy obtained a number of repairs that was strictly smaller than that of any other strategy. Somewhat surprisingly, the worst strategy when it comes to obtaining few repairs, viz., random, was still able to dominate all others at least a couple of times per avg. degree d. This means that, by removing random, we would obtain a slightly worse result for GREEDUS in Figure 4a, or, in other words, that random, albeit not the most helpful of all strategies, can still be considered indispensable. In Figure 4c we observe the percentage of times that an approach fails to find the optimal value. The performance of the strategies here mirrors that of Figure 4a (the two measures, of course, correlate), but what we get from Figure 4c is that GREEDUS can find the optimal value for the majority of instances up to an avg. degree d of 10. Even though the situation might seem dramatic for an avg. degree d of 12 and 14, the distance to the optimal value, as reported in Figure 4a, is quite small, and a failure still registers as a failure even when a value of $x + 1$ is reported instead of the optimal x. Finally, in Figure 4d we can see the implementation of GREEDUS scaling gracefully as the network instances become denser, whereas the performance of the implementation of the Partial MaxSAT encoding starts deteriorating drastically and even time-outs a few times when trying to solve the densest of instances.

▶ Remark 5. The time for generating the Partial MaxSAT encoding of a QCN was not taken into acount in our evaluation and, in particular, in Figure 4d. This is because the encoding is currently not generated in an optimal way and it would skew the results in favor of the implementation of GREEDUS. However, some computational effort would be required in any case to produce the encoding, so what we see in Figure 4d for the implementation of the Partial MaxSAT encoding is a lower bound (with respect to our experimental evaluation here). In addition, despite the fact that the implementation of GREEDUS was sped up with PyPy, some overhead still remains, since it is fully coded in the high-level language of Python, which has an inherent performance disadvantage against low-level languages like C/C++. Thus, what we see in Figure 4d for the implementation of GREEDUS is an upper bound. In fact, based on algorithm design alone, it should be feasible to have an implementation of GREEDUS that would either match or exceed the performance of the implementation of the Partial MaxSAT encoding in all cases. The *main takeaway* regarding runtime performance here is that GREEDUS scales much better with respect to the average constraint graph degree of the network instances, and this scaling behaviour is accurately depicted in Figure 4d.

6 Conclusion and Future Work

In this paper, we focused on the problem of resolving inconsistency in qualitative constraint networks (QCNs), which can be viewed as knowledge bases of intutive, human-like descriptions of spatio-temporal information like x $\{$*is north of* $\vee$ *is east of*$\}$ y. In particular, we presented two novel approaches for maximizing satisfiability in such networks: a greedy constraint-based and an optimal Partial MaxSAT-based one. The greedy technique adds the constraints of a given QCN to a new, initially empty network, one by one, filtering out the ones that fail the satisfiability check during the process; in doing so, it relies on many different strategies that create various orderings of the constraints to be processed, in a portfolio-style setting. The Partial MaxSAT encoding exploits to the fullest extent possible certain tractability properties associated with QCNs, viz., Horn theory-based *maximal tractable subsets* of relations [22], and is thus one of the most compact to date Partial MaxSAT encodings for the MAX-QCN problem, as evidenced also by the special case where all its clauses are assumed to be hard (the SAT case) [30]. We compared the two approaches against each other and provided some insight on the trade-off between obtaining repairs that are optimal and obtaining repairs in a manner that is fast. For future work, we would like to apply the techniques discussed here to other inconsistency-related reasoning tasks, such as the recently introduced one of decomposing QCNs into consistent components [24]. Further, we would like to explore more on the use of SAT/MaxSAT solvers, especially solvers based on local search, e.g., [4], as we think that they would better suit our needs; in our experience, inconsistencies in QCNs tend to form locally. Finally, we are looking into ways of devising an optimal method out of our greedy approach.

References

1. James F. Allen. Maintaining Knowledge about Temporal Intervals. *Commun. ACM*, 26:832–843, 1983.
2. Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving Glucose for Incremental SAT Solving with Assumptions: Application to MUS Extraction. In *SAT*, 2013.
3. Mehul Bhatt, Hans Guesgen, Stefan Wölfl, and Shyamanta Hazarika. Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions. *Spatial Cognition & Computation*, 11:1–14, 2011.
4. Shaowei Cai and Zhendong Lei. Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability. *Artif. Intell.*, 287:103354, 2020.
5. Shaowei Cai, Chuan Luo, John Thornton, and Kaile Su. Tailoring Local Search for Partial MaxSAT. In *AAAI*, 2014.
6. Jean-François Condotta, Ali Mensi, Issam Nouaouri, Michael Sioutis, and Lamjed Ben Said. A Practical Approach for Maximizing Satisfiability in Qualitative Spatial and Temporal Constraint Networks. In *ICTAI*, 2015.
7. Jean-François Condotta, Issam Nouaouri, and Michael Sioutis. A SAT Approach for Maximizing Satisfiability in Qualitative Spatial and Temporal Constraint Networks. In *KR*, 2016.
8. Frank Dylla, Jae Hee Lee, Till Mossakowski, Thomas Schneider, André van Delden, Jasper van de Ven, and Diedrich Wolter. A Survey of Qualitative Spatial and Temporal Calculi: Algebraic and Computational Properties. *ACM Comput. Surv.*, 50:7:1–7:39, 2017.
9. S.M. Hazarika. *Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions*. IGI Global, 2012.
10. Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, 2018.
11. Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an Efficient MaxSAT Solver. *J. Satisf. Boolean Model. Comput.*, 11:53–64, 2019.
12. David S. Johnson. Approximation Algorithms for Combinatorial Problems. *J. Comput. Syst. Sci.*, 9:256–278, 1974.
13. Jae Hee Lee, Michael Sioutis, Kyra Ahrens, Marjan Alirezaie, Matthias Kerzel, and Stefan Wermter. Neuro-Symbolic Spatio-Temporal Reasoning. *CoRR*, abs/2211.15566, 2022. URL: `https://doi.org/10.48550/arXiv.2211.15566`.
14. Joseph Y.-T. Leung, editor. *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.
15. Gérard Ligozat. *Qualitative Spatial and Temporal Reasoning*. ISTE. Wiley, 2013.
16. Shuichi Miyazaki, Kazuo Iwama, and Yahiko Kambayashi. Database Queries as Combinatorial Optimization Problems. In *CODAS*, 1996.
17. Robert Moskovitch and Yuval Shahar. Classification of multivariate time series via temporal abstraction and time intervals mining. *Knowl. Inf. Syst.*, 45:35–74, 2015.
18. Lenka Mudrová and Nick Hawes. Task scheduling for mobile robots using interval algebra. In *ICRA*, 2015.
19. Bernhard Nebel. Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class. *Constraints*, 1:175–190, 1997.
20. Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra. *J. ACM*, 42:43–66, 1995.
21. Jochen Renz and Bernhard Nebel. Efficient Methods for Qualitative Spatial Reasoning. *J. Artif. Intell. Res.*, 15:289–318, 2001.
22. Jochen Renz and Bernhard Nebel. Qualitative Spatial Reasoning Using Constraint Calculi. In *Handbook of Spatial Logics*, pages 161–215. Springer, 2007.
23. Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
24. Yakoub Salhi and Michael Sioutis. A Decomposition Framework for Inconsistency Handling in Qualitative Spatial and Temporal Reasoning. In *KR*, 2023.

25 Michael Sioutis and Diedrich Wolter. Dynamic Branching in Qualitative Constraint Networks via Counting Local Models. In *TIME*, 2020.

26 Michael Sioutis and Diedrich Wolter. Qualitative Spatial and Temporal Reasoning: Current Status and Future Challenges. In *IJCAI*, 2021.

27 Peter van Beek and Dennis W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *J. Artif. Intell. Res.*, 4:1–18, 1996.

28 Matthias Westphal. *Qualitative Constraint-based Reasoning: Methods and Applications*. PhD thesis, Albert-Ludwigs-Universität Freiburg, 2014.

29 Matthias Westphal and Julien Hué. A Concise Horn Theory for RCC8. In *ECAI*, 2014.

30 Matthias Westphal, Julien Hué, and Stefan Wölfl. On the Propagation Strength of SAT Encodings for Qualitative Temporal Reasoning. In *ICTAI*, 2013.

31 Zhi-Hua Zhou. Abductive learning: towards bridging machine learning and logical reasoning. *Sci. China Inf. Sci.*, 62:76101:1–76101:3, 2019.