Nogoods in Qualitative Constraint-based Reasoning

Matthias Westphal and Julien Hué

Department of Computer Science, University of Freiburg,
Georges-Köhler-Allee 52, 79110 Freiburg, Germany
{westpham,hue}@informatik.uni-freiburg.de

Abstract. The prevalent method of increasing reasoning efficiency in the domain of qualitative constraint-based spatial and temporal reasoning is to use domain splitting based on so-called tractable subclasses. In this paper we analyze the application of nogood learning with restarts in combination with domain splitting. Previous results on nogood recording in the constraint satisfaction field feature learnt nogoods as a global constraint that allows for enforcing generalized arc consistency. We present an extension of such a technique capable of handling domain splitting, evaluate its benefits for qualitative constraint-based reasoning, and compare it with alternative approaches.

1 Introduction

Qualitative Spatial and Temporal Reasoning (QSTR) is a knowledge representation discipline that deals with information about relations between objects defined on infinite domains, such as time and space. For example, two entities in space might "overlap" or one is a "part of" the other. A common reasoning task considered in QSTR is to solve constraint satisfaction problems over infinite domains with constraints from a fixed finite set of relations. With only a finite number of qualitative relations posing as constraints between entities, the idea is to employ inference techniques to tighten these constraints.

Constraint-based QSTR problems can be considered as entirely symbolic tasks where the qualitative relations are treated as symbols. Naturally, this leads to a constraint satisfaction problem on a finite domain where qualitative relations are possible values and constraint propagation enforces matching relation tuples. This type of QSTR has mostly benefited from the development of large tractable subclasses used by domain splitting branching rules [1, 2].

Recently encodings of these problems into Boolean SAT-formulas have attracted considerable interest. The obtained benchmarking results [3, 4] indicate that the constraint-based QSTR methods very often result in good runtime due to the use of fast, optimized constraint propagation algorithms and domain splitting. However, the results also show that the exploration of the search space is worse on very hard problems compared to SAT solvers on optimized encodings. This suggest that a blend of SAT/CSP and QSTR techniques should produce

better results. More specifically, associated with runtime distributions is the technique of restarting search with learning so-called nogoods – parts of the search space that do not contain a solution.

We pursue the questions of the impact of nogood learning and restarts on QSTR problems and how nogood learning can beneficially be used with specialized constraint propagation and domain splitting. There is a number of different approaches that can be taken – here, we focus on solutions that put nogoods on top of arbitrary propagation techniques. In particular, we mainly discuss a lightweight solution that does not perform a conflict analysis at each conflict which has the benefit of being easier to integrate and causes almost no time overhead on easy problem instances. We only briefly discuss our experience with conflict analysis at each conflict.

The outline of this paper is as follows. In the next section we give standard definitions for concepts from constraint satisfaction. In Section 3 we give some background on QSTR and in Section 4 we introduce techniques to combine nogoods with domain splitting. Section 5 outlines our implementation and evaluates the proposed techniques. Finally, Section 6 gives our conclusions.

2 Notation

We define several standard concepts of CSPs.

Definition 1. *A* **finite constraint satisfaction problem** *(finite CSP) is an ordered tuple $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$, where (i) $\mathcal{V}$ is a finite set of variables, (ii) $\mathcal{D}$ is a finite set of values (the domain), (iii) $\mathcal{C}$ is a finite set of constraints, where each constraint $\big((v_1, \ldots, v_n), R\big)$ consists of a relation R on $\mathcal{D}^n$ and a scope $v_1, \ldots, v_n \in \mathcal{V}$.*

Definition 2. *A* **solution** *φ of a finite CSP $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ is a function $\varphi \colon \mathcal{V} \to \mathcal{D}$ such that for each constraint $\big((v_1, \ldots, v_n), R\big) \in \mathcal{C}$ it holds $\big(\varphi(v_1), \ldots, \varphi(v_n)\big) \in R$.*

We consider depth-first search (DFS) with an inference algorithm ϕ. To this end, let $dom(v) \subseteq \mathcal{D}$ denote the set of remaining domain values of a variable $v \in \mathcal{V}$ at a search node. For backtracking search, we restrict ourselves to a 2-way branching scheme that employs a *domain splitting* branching rule [5] which restricts possible values of a domain rather than assigning a specific value.

Definition 3. *A* **decision** *on a variable $v \in \mathcal{V}$ during DFS on a finite CSP $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ is a unary constraint on v, written $v \leftarrow D = \{a_1, \ldots, a_n\} \subsetneq dom(v)$, that restricts the remaining values of v at succeeding search nodes.*

In the 2-way branching scheme, we first perform a *positive decision* $v \leftarrow D$ at a search node, and once we backtrack to this node, a *negative decision* $v \not\leftarrow D$ which is a shorthand for $v \leftarrow dom(v) \setminus D$. Each branch of the search tree can be seen as a *sequence of decisions* $\langle v_1 \leftarrow D_1, \ldots, v_n \leftarrow D_n \rangle$, where at each search node i we apply ϕ after enforcing $v_i \leftarrow D_i$. The remaining values of each variable $v \in \mathcal{V}$ are restricted by ϕ and we backtrack whenever $dom(v) = \emptyset$.

In order to combine nogoods with domain splitting, we require generalized nogoods which cover not only assignments of single values to variables, but also arbitrary sets of values.

Definition 4 ([6]). *A* **generalized nogood** *of a finite CSP* $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ *is a sequence of decisions* $\langle v_1 \leftarrow D_1, \ldots, v_n \leftarrow D_n \rangle$ *such that there is no solution for* $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \cup \{ (v_i, D_i) \mid 1 \leq i \leq n \} \rangle$.

3 Qualitative Constraint-Based Reasoning

Usually constraint satisfaction problems are assumed to be defined on a *finite* domain. Solutions (or a proof that none exists) are usually generated by explicitly assigning values to variables. In contrast, within constraint-based QSTR one considers constraints on infinite domains like time (e.g. the domain $\mathbb{Q}$) or space (e.g. the domain $\mathbb{Q}^2$). Hence, there is no basic default method like enumerating possible solutions to handle such problems. A key idea in QSTR is to consider as input constraint languages build on *finitely* many constraint relations. In this work, we consider input languages built on a *partition scheme* defined as follows.

Definition 5 ([7]). *A* **partition scheme** *on an infinite domain* $\mathcal{D}_\infty$ *is a finite set* $\mathcal{B}$ *of binary relations on* $\mathcal{D}_\infty$ *that forms a partition of* $\mathcal{D}_\infty \times \mathcal{D}_\infty$*, contains the identity relation* $\{ (x, x) \mid x \in \mathcal{D}_\infty \}$*, and is closed under converses* ($B^{-1} := \{ (y, x) \mid (x, y) \in B \} \in \mathcal{B}$ *for* $B \in \mathcal{B}$*).*

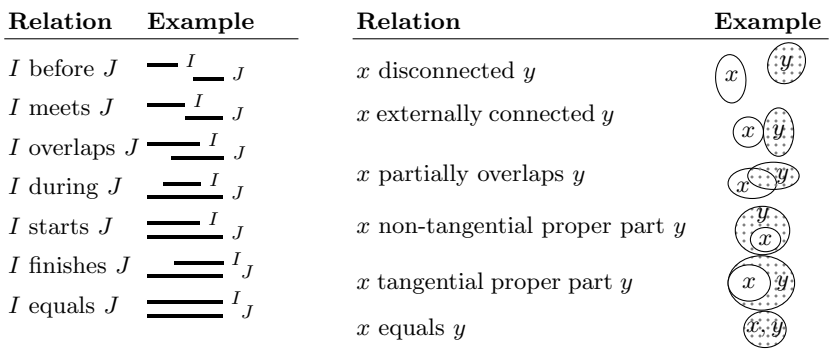

Fig. 1. Base relations (without converses); *left*: Allen's Interval Calculus, *right*: RCC-8.

Elements of $\mathcal{B}$ are called *base relations* of the partition scheme and exhaustively describe possible, distinct relations between entities. As an example consider the base relations of Allen's Interval Calculus [8] (AIC) for temporal reasoning and the Region Connection Calculus [9] with 8 base relations (RCC-8) for spatial reasoning in geographic information systems, depicted in Fig. 1.

To deal with indefinite knowledge we allow disjunctions of base relations to form relations between entities, e.g. if x_1 happened before or after x_2 we can write $(x_1 \,\text{before}\, x_2 \vee x_1 \,\text{after}\, x_2)$. We write $\mathcal{B}^*$ to denote the set of all possible disjunctions of base relations. This allows us to form logic statements about the relationship between multiple entities $x_1, \ldots, x_n \in \mathcal{D}_\infty$ by a formula: $\bigwedge_{1 \leq i < j \leq n} (\bigvee_{1 \leq l \leq k} x_i B_{ij}^l x_j), B_{ij}^l \in \mathcal{B}$. Such formulas are referred to as *qualitative constraint networks*.

The fundamental reasoning task for qualitative constraint networks is the *consistency problem*, i.e., deciding whether the input is consistent wrt. given inference rules. We here use the relation-algebraic approach that utilizes composition (denoted by $\circ$) on relations to establish local consistency on $\mathcal{D}_\infty$. The composition approach (often referred to as path consistency) is equivalent to a complete set of valid inference rules of the form

$$\forall x, y, z \in \mathcal{D}_\infty : (x\ R'\ y \wedge y\ R''\ z) \rightarrow \neg(x\ R\ z), \tag{1}$$

for $R, R', R'' \in \mathcal{B}^*$ and $R = \overline{(R' \circ R'')}$. In other words these rules remove those base relations from every triple that are not locally consistent. For example, we can conclude that $(x \prec y) \wedge (y \prec z) \wedge (x \succ z)$ is contradictory, since $(x \prec y \wedge y \prec z) \rightarrow \neg(x \succ z \vee x = z)$ is a valid rule. Thus, dealing with qualitative constraint networks can be cast as a *finite* constraint satisfaction problem. Here, $\mathcal{V} = \{\, x_{ij} \mid i < j \,\}, \mathcal{D} = \mathcal{B}^*, \mathcal{C} = \{\,\text{inference rules (1)}\,\}$, where x_{ij} refers to the relation between x_i and x_j. The latter can be used as an intensional constraint to avoid grounding the rules to tables for all triples. Enforcing these rules is equivalent to generalized arc consistency (GAC) (see, e.g. [3] for a discussion and related work). In general, the set of constraints $\mathcal{C}$ can be seen as a global constraint where the inference used is not necessarily built on rules in the form of (1), e.g. [10]. However, we stick to these rules in this work as it is a general approach to several qualitative formalisms. The nogood technique introduced herein is also applicable to specialized inference algorithms.

It is clear that valid rules in the form of (1) can be used to refute statements as every rule itself is a logically correct inference. The converse, however, is not necessarily true, as it depends on a "local-to-global" consistency property[1] of the used qualitative calculus (and $\mathcal{D}_\infty$). Whether there is such a set of rules that is *refutation complete*, depends on the used relations and $\mathcal{D}_\infty$. Problems of this type are in general undecidable, but both AIC and RCC-8 have good properties in this regard as we will briefly discuss next.

For both AIC and RCC-8, rules (1) are not refutation complete on the sets $\mathcal{B}^*$ (reasoning here is in fact NP-complete). However, the rules are refutation complete for $\mathcal{B}$. Moreover, these rules are refutation complete for the sets ORD-horn for AIC, and $\widehat{\mathcal{H}}_8$ for RCC-8. Both ORD-horn and $\widehat{\mathcal{H}}_8$ are strictly larger than the set of base relations and are maximal *tractable subclasses* (see [1, 2] for detailed discussion and proofs). The set ORD-horn covers 868 of all the 8192 relations in AIC, $\widehat{\mathcal{H}}_8$ covers 148 of all the 256 relations in RCC-8. Such tractable subsets motivate the following approach for solving instances that has been used in qualitative reasoners: (a) use domain splitting to refine relations such that they are included in a fixed tractable set, (b) maintain local consistency by using the inference rules on qualitative relations. Wrt. (a), it has been shown [11] that decisions should need only to take place once per variable on a search branch.

[1] Not to be confused with "global consistency" which is a stronger property.

4 Nogoods in Constraint-based QSTR

There exist different approaches to learning nogoods. We mainly consider the lightweight approach of Lecoutre et al. [12] where nogoods are only extracted from search once a solver restarts. Another approach is the work by Katsirelos and Bacchus [6] where nogoods are learnt from each conflict. We here analyze how the lightweight approach, originally only considering decisions as assignments of single values, can be extended to generalized nogoods and where this generalization worsens complexity bounds. For this we only briefly repeat discussion and arguments found in the work by Lecoutre et al. as our focus is on domain splitting. More details (without domain splitting) can be found in their paper. In the following, we assume a finite CSP $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ and use the following notation for complexity bounds: n as the number of variables in $\mathcal{V}$, d as the size of the domain $\mathcal{D}$, $\mathcal{N}$ as the set of learnt nogoods.

4.1 Extracting Nogoods from Search

The easiest way to learn nogoods during search is to use the current sequence of decisions whenever backtracking occurs. Hence, we limit ourselves to nogoods that are (sub-)sequences of decisions starting from the root node of the search tree. This makes extracting and using nogoods easier, but also means that nogoods derived in this way are useless for the current DFS, because the 2-way branching scheme already incorporates information from such failures. For this reason, we use nogoods in combination with restarts of the DFS.

It is sufficient to only consider the last branch of the search to derive nogoods, since due to the 2-way branching scheme all previous decision failures are accounted for. To extract the set of nogoods, we consider all prefixes of the corresponding sequence of decisions that end in a negative decision. For each such sequence $\langle v_1 \leftarrow D_1, \ldots, v_i \not\leftarrow D_i \rangle$, $\langle v_1 \leftarrow D_1, \ldots, v_i \leftarrow D_i \rangle$ was shown to be a nogood. All negative decisions can be stripped from each nogood since negative decisions were implied by the search. Additionally, we can try to minimize these nogoods by looking for a subset of its decisions where inference already finds a contradiction as in [12]. The number of nogoods derived from a search branch is unaffected by the generalization to domain splitting, unlike space complexity.

Proposition 1. *The space complexity of storing all nogoods that can be extracted from a search branch is $O(n^2 d)$ for singleton assignments [12] and $O(n^2 d^2)$ for domain splitting.*

Proof. We argue as in [12]: there are $O(nd)$ nogoods derived from the branch, each of them covering $O(n)$ positive decisions. Each decision (with domain splitting) requires $O(d)$ space and hence $O(n^2 d^2)$ space is required to store nogoods.

4.2 Using Nogoods for Inference

Following the approach by Lecoutre et al., we treat nogoods as additional constraints and take them into account when establishing GAC. Each nogood $\langle v_1 \leftarrow$

$D_1, \ldots, v_n \leftarrow D_n\rangle$ constitutes the constraint $dom(v_1) \not\subseteq D_1 \vee \cdots \vee dom(v_n) \not\subseteq D_n$. For propagation, we consider a lazy data structure built on watched literals.

Unfortunately, our extension to generalized nogoods causes the original approach of [12] to be not directly applicable. For singleton assignments it is sufficient to check if a variable equals a previous decision. For decisions based on domain splitting, we need to check subset relations. We stick to the idea of watched literals and extend the original idea by Lecoutre et al. as follows. We associate two watched literals with each nogood, but a decision $v \leftarrow D$ is associated with a watched literal (v, a) where $a \notin D$. As long as a (v, a) is part of the network, the restriction associated with the decision $v \leftarrow D$ has not happened in the network and the watched literal is valid. Further, we need to make sure that both watched literals of a nogood are on different variables, since two valid watched literals guarantee GAC and otherwise restrictions on domain values apply.

Algorithm 1 Propagation with watched literals for nogood constraints.

```
 1: function PROPAGATE(queue)
 2:     while queue ≠ ∅ do
 3:         v ← pick and remove variable from queue
 4:         for each a removed by REVISE or REMOVED on v do
 5:             if not REMOVED(v, a, queue) then
 6:                 return false
 7:         for every constraint C involving v do
 8:             for w ∈ scope(C) \ {v} do
 9:                 if REVISE(w, C) then
10:                     if dom(w) = ∅ then
11:                         return false
12:                     queue ← queue ∪ {w}
13:     return true
```

Algorithm 1 gives the constraint propagation with GAC for generalized nogoods. It is a regular propagation function with a REVISE function that handles the constraints $\mathcal{C}$ and additional lines 4-6 that take care of learnt nogoods. The function REMOVED will handle the learnt nogoods and we invoke it $O(nd)$ times in our scheme, as opposed to the original algorithm by Lecoutre et al. which only invokes it if a domain becomes singleton (which only happens $O(n)$ times).

Algorithm 2 details REMOVED and shows how watched literals are managed and GAC performed on nogoods. In order to achieve a low time complexity, we note that the order of decisions in a nogood is originally unimportant, such that we can arrange them in a way where decisions that cannot be watched anymore are ordered before the currently watched ones. The consequence is that during constraint propagation every decision in a nogood is analyzed only until it cannot be watched anymore. This requires a preprocessing step before each constraint propagation that moves the currently watched decisions to the front of the nogood, and requires modifying the for-loop over decisions in REMOVED, such that only decisions behind the currently watched ones are considered (see [12]).

Algorithm 2 Enforce generalized arc consistency on $\mathcal{N}$.

```
1: function REMOVED(v, a, queue)                    ▷ a was just removed from dom(v)
2:     for each nogood N that watches (v, a) do
3:         Let (v', a') be the other watched literal in N
4:         Let D' be the assigned set in v' ← D' ∈ N
5:         if dom(v') ∩ D_{v'} ≠ ∅ then                                  ▷ applicable
6:             changed ← false
7:             for each decision v'' ← D'' ∈ N, v'' ≠ v' do
8:                 if dom(v'') ⊈ D'' then
9:                     Let (v'', a''), such that a'' ∈ dom(v'') \ D''
10:                    Replace (v, a) with (v'', a'')
11:                    changed ← true
12:                    break
13:            if not changed then                                       ▷ enforce GAC
14:                dom(v') ← dom(v') \ D'
15:                if dom(v') = ∅ then return false
16:                queue ← queue ∪ {v'}
17:    return true
```

Proposition 2. *Enforcing GAC with the watched literal approach for generalized nogoods adds an additional cost of $O(nd^2|\mathcal{N}|)$ to the time complexity of existing constraint propagation.*

Proof. Working on a decision of a nogood incurs a cost of $O(d)$ (set theoretic operations). Every decision of a nogood can only be considered $O(d)$ times. We obtain $O(nd^2)$ for each nogood, i.e., the overall complexity $O(nd^2|\mathcal{N}|)$.

5 Implementation and evaluation of the Techniques

We have implemented the proposed lightweight nogood technique for domain splitting, the original technique by Lecoutre et al. [12], and further the extraction of nogoods via backchaining from conflicts by Katsirelos and Bacchus [6]. The proposed technique has been implemented in the qualitative constraint solver GQR [2] [13, 3] and thus we have optimized constraint propagators for the inference rules. GQR represents domains as bitsets and assigns a predefined weight to each base relation estimating its restrictiveness wrt. composition [14, 2]. These weights allow us to estimate the restrictiveness of remaining domain values by the sum of the elements' weights.

Further, we use 2-way branching and maintain GAC (cf. Section 3). The selection of variables is based on dom/wdeg [15], where domain size is replaced with the weight of the domain. Depending on the used branching strategy, value selection considers sets contained in a fixed predefined tractable subclass (cf. Section 3, domain splitting) or any included singleton value. We here choose a subset of the domain with maximum weight with cardinality used for tie-breaking.

[2] `http://sfbtr8.informatik.uni-freiburg.de/R4LogoSpace/Resources/GQR`

Propagation is handled by a coarse-grained scheme [16], as depicted in Algorithm 1. The queue used is a priority-queue that returns a variable where the weight of the domain is minimal [14].

For all nogood schemes, we perform unbounded learning, i.e., no extracted nogood is deleted or ignored. Restarts are based on a geometric restart policy based on the number of decision failures. The first DFS run is terminated after 10 failures and the limit for the next run is increased by a factor of 1.5.

We evaluate the discussed nogood approaches with the qualitative calculi AIC and RCC-8. Although we have implemented the generic approach for extracting nogoods from conflicts presented by Katsirelos and Bacchus [6], we do not detail here due to a lack of space and the observed running times significantly showing their generic method is unsuited to this framework.

We compare the following branching strategies with nogoods: (a) singleton assignments without restarts or nogoods (b) singleton assignments with the original nogood approach from [12], (c) domain splitting without restarts or nogoods, and (d) domain splitting with our presented nogood approach. For (c), (d) we use as tractable subclasses ORD-horn [1] for AIC and $\widehat{\mathcal{H}}_8$ [2] for RCC-8. We write $\mathcal{B}$ for (a), $\mathcal{B}+\mathcal{N}$ for (b), ORD-horn (or $\widehat{\mathcal{H}}_8$) for (c) and ORD-horn$+\mathcal{N}$ (or $\widehat{\mathcal{H}}_8+\mathcal{N}$) for (d). To at least briefly illustrate the behaviour of SAT solvers, we include results for the specialized encoding of the AIC from [4] called IA2SAT which is based on network decomposition. We here use the simplification version of MiniSAT 2.2.0 [17] as backend.

Unfortunately, there is no large set of benchmark instances from applications, such that we have to rely on randomly generated qualitative constraint networks as in [1–3]. For AIC, we derive random instances by fixing the number of considered entities, e, the average number of non-trivial qualitative relations an entity is involved in, c, and the average size of variables' initial domain, l. This is the so-called A-model [1], and we write $A(e, c, l)$ to denote the corresponding set of problems. We set l to be half the number of base relations to obtain a uniform distribution of relation labels. The set of pairs of entities with non-trivially relations and the used qualitative relations are chosen randomly, such that they average around c and l, respectively. In particular, c controls the tightness of the constraint problem and we use it to obtain problems from the phase transition. For RCC-8, we use the H-model with the set of $\mathcal{NP}_8$ [1, 2], which only differs from the A-model in requiring selected qualitative relations to be not included in $\widehat{\mathcal{H}}_8$. For each considered set we generated 1 000 problem instances. All experiments were conducted on an Intel Xeon CPU with 2.66 GHz, 4 GB memory, and a CPU time limit of 2 hours.

Tables 1-4 contain our obtained results where the best results are highlighted. As far as the runtime of the solver is concerned, we can conclude that both lightweight approaches have a positive impact. The addition of restarts and nogoods significantly lowers the average number of decisions and the runtime in every considered setup. We note here that changing the restarting strategy from the geometric scheme to the Luby sequence, changing the initial restart constant, or even applying minimization to the learnt nogoods as in [12] causes little change

in the results. The presented results for domain splitting with restarts based on the geometric scheme without minimization are the best we have observed.

From the results, we further conclude that $\mathcal{B}+\mathcal{N}$ does not achieve the efficiency of ORD-horn or $\widehat{\mathcal{H}}_8$ approaches. The gap between singleton assignments and domain splitting remains significant and domain splitting with restarts and lightweight nogoods outperforms all other variants. For 100 entity networks, we can see the nogood approaches to achieve a speedup of about 25-50% for medium to hard instances with very little overhead on easy instances (Tables 1,3). The nogood approaches have an even more significant impact on larger networks, where we can observe reductions of more than 50% (see Tables 2,4). Here the given constraints in networks are less dense and thus (without learnt nogoods) perhaps less restrictive.

Table 1. Times and decisions for AIC in each set $A(100, c, 6.5)$.

c	approach	solved	average	percentiles (25-, 50-, 75-, 90-)			
10.0	$\mathcal{B}$	997	34 044.83d	395d	1 140d	7 976d	46 478d
			78.67s	0.28s	1.96s	17.96s	100.43s
	$\mathcal{B}+\mathcal{N}$	999	19 601.61d	403d	754d	2 736d	14 754d
			39.09s	0.24s	0.82s	4.68s	24.94s
	ORD-horn	**1 000**	3 646.24d	**102d**	368d	1 395d	5 771d
			6.87s	**0.11s**	0.60s	2.66s	10.62s
	ORD-horn $+\mathcal{N}$	**1 000**	**2 111.70d**	129d	**336d**	**1 051d**	**3 014d**
			3.57s	0.12s	**0.38s**	**1.56s**	**4.96s**
10.5	$\mathcal{B}$	993	57 959.71d	113d	1 168d	13 451d	91 228d
			126.71s	0.15s	1.93s	27.12s	215.01s
	$\mathcal{B}+\mathcal{N}$	996	37 156.55d	86d	767d	5 249d	41 549d
			70.43 s	0.11s	0.98s	9.08s	82.20s
	ORD-horn	**1 000**	6 716.61d	**57d**	388d	2 230d	10 076d
			12.56s	**0.06s**	0.59s	4.18s	19.57s
	ORD-horn $+\mathcal{N}$	**1 000**	**4 169.03d**	60d	**335d**	**1 344d**	**5 688d**
			7.03s	0.07s	**0.42s**	**2.08s**	**9.34s**
	IA2SAT	**1 000**	15 401.87d	2 016d	9 694d	22 680d	36 388d
			56.59s	34.15s	46.96s	65.91s	99.47s
11.0	$\mathcal{B}$	995	35 040.04d	23d	507d	4 477d	38 619d
			76.42s	0.03s	0.71s	7.76s	78.06s
	$\mathcal{B}+\mathcal{N}$	996	19 680.94d	25d	401d	2 781d	20 910d
			36.28s	0.03s	0.45s	4.48s	34.75s
	ORD-horn	**1 000**	3 839.38d	**13d**	209d	1 144d	5 694d
			7.23s	**0.02s**	0.29s	1.94s	10.57s
	ORD-horn $+\mathcal{N}$	**1 000**	**2 882.40d**	18d	**198d**	**911d**	**4 354d**
			5.03s	0.03s	**0.25s**	**1.39s**	**7.31s**

Finally, Fig. 2 gives a per instance instance comparison for the hardest set of problems in AIC, $A(150, 10.5, 6.5)$. We can see for both nogood techniques the runtime on satisfiable instances becomes scattered (most likely due to restarts), while runtimes on unsatisfiable instances deviate less but show a positive trend (cf. Fig. 2). We also have to acknowledge that the proposed technique does not strongly reduce the heavy-tailed behavior (cf. Fig. 2).

Table 2. Times and decisions for AIC in the set $A(150, 10.5, 6.5)$.

c	approach	solved	average	percentiles (25-, 50-, 75-, 90-)			
10.5	ORD-horn	889	161 697.01d	3 702d	39 021d	358 550d	–
			746.97s	16.38s	182.29s	1 632.30s	–
	ORD-horn + $\mathcal{N}$	929	135 370.43d	**2 424d**	**17 822d**	173 075d	1 089 146d
			578.11s	**9.25s**	**74.74s**	**727.24s**	4 695.58s
	IA2SAT	**967**	**98 011.87d**	32 152d	81 126d	**146 257d**	**252 273d**
			1 105.24s	377.95s	728.56s	1 446.35s	**3 155.23s**

Table 3. Times and decisions for RCC-8 in each set $H(100, c, 4.0)$.

c	approach	solved	average	percentiles (25-, 50-, 75-, 90-)			
14.5	$\mathcal{B}$	1 000	5 941.69d	1 561d	1 922d	3 846d	9 903d
			7.92s	0.27s	0.98s	4.25s	14.34s
	$\mathcal{B} + \mathcal{N}$	1 000	3 442.47d	1 557d	1 824d	2 616d	5 740d
			3.26s	0.25s	0.54s	1.79s	6.73s
	$\widehat{\mathcal{H}}_8$	1 000	1 105.05d	**115d**	**303d**	951d	2 595d
			1.01s	**0.07s**	0.24s	0.83s	2.34s
	$\widehat{\mathcal{H}}_8 + \mathcal{N}$	1 000	**843.40d**	188d	364d	**727d**	**1 694d**
			0.63s	0.08s	**0.17s**	**0.48s**	**1.36s**
15.0	$\mathcal{B}$	1 000	8 790.21d	816d	1 989d	4 947d	16 149d
			12.87s	0.29s	1.47s	5.96s	24.56s
	$\mathcal{B} + \mathcal{N}$	1 000	7 182.89d	747d	1 854d	3 495d	8 805d
			9.19s	0.28s	0.93s	3.14s	10.48s
	$\widehat{\mathcal{H}}_8$	1 000	1 705.25d	**164d**	**478d**	1 500d	3 926d
			1.58s	0.11s	0.39s	1.33s	3.71s
	$\widehat{\mathcal{H}}_8 + \mathcal{N}$	1 000	**1 315.95d**	200d	511d	**1 200d**	**2 577d**
			1.08s	**0.10s**	**0.30s**	**0.88s**	**2.12s**
15.5	$\mathcal{B}$	1 000	6 537.62d	177d	1 412d	3 301d	10 673d
			9.58s	0.15s	0.68s	3.61s	14.90s
	$\mathcal{B} + \mathcal{N}$	1 000	4 339.76d	174d	1 225d	2 600d	6 541d
			5.60s	0.15s	0.57s	2.46s	7.93s
	$\widehat{\mathcal{H}}_8$	1 000	1 437.97d	**99d**	**315d**	1 031d	2 710d
			1.32s	**0.06s**	0.23s	0.92s	2.52s
	$\widehat{\mathcal{H}}_8 + \mathcal{N}$	1 000	**1 197.78d**	128d	352d	**954d**	**2 198d**
			1.01s	0.07s	**0.21s**	**0.72s**	**1.84s**

Table 4. Times and decisions for RCC-8 in the set $H(150, 16.0, 4.0)$.

c	approach	solved	average	percentiles (25-, 50-, 75-, 90-)			
16.0	$\widehat{\mathcal{H}}_8$	989	105 926.94d	2 395d	11 043d	65 433d	273 493d
			226.87s	4.76s	23.12s	137.88s	604.58s
	$\widehat{\mathcal{H}}_8 + \mathcal{N}$	**992**	**57 999.05d**	**1 241d**	**3 802d**	**20 172d**	**116 095d**
			120.45s	**1.81s**	**7.10s**	**41.68s**	**232.59s**

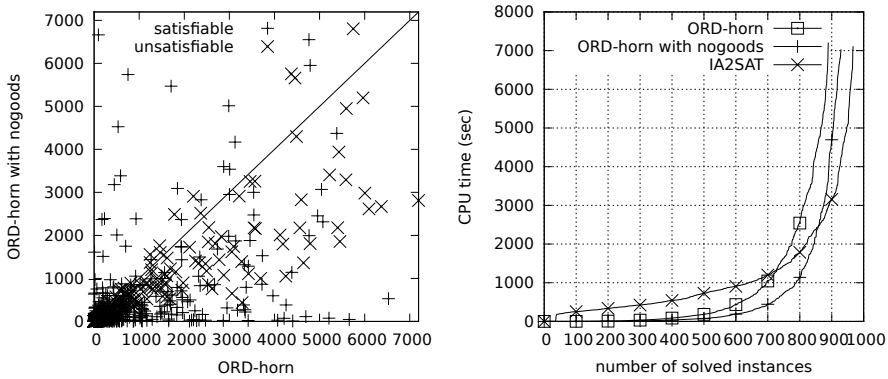

Fig. 2. Plotted data for AIC $A(150, 10.5, 6.5)$.

In summary, nogoods with restarts clearly improve the robustness and efficiency with little overhead on easy instances. The proposed technique is an improvement and clearly outperforms IA2SAT on instances with 100 entities, and for 150 entities with the exception of the hardest 10% of instances.

6 Conclusion

In this paper, we have discussed and analyzed an extension of the nogood recording and inference technique presented by Lecoutre et al. [12] to a branching scheme with domain splitting. The overhead caused by the extension to domain splitting is low polynomial and the method is still efficient.

We have further shown how nogood techniques are applicable in the field of qualitative constraint-based reasoning and help to improve the efficiency of constraint solving. The profound impact of nogood learning and restarts that we demonstrate also helps to understand empirical results of recently studied SAT encodings. Our results show that the approach is well suited to improve qualitative reasoning procedures, in particular because it can be used with any inference algorithm on qualitative relations.

With regard to learning nogoods from conflicts, we note that while the generic method of Katsirelos and Bacchus [6] has not performed well in our case, we have not tried alternative, specialized methods tailored towards the considered constraints. It was shown by Katsirelos and Bacchus that it is often desirable to construct such specialized methods for extracting nogoods. Moreover, it would be interesting to consider the impact of structural restrictions as considered by Boolean SAT encodings [4] in the context of extracting nogoods.

Another interesting point for qualitative reasoning is the question whether a general approach building on inference rules as used here is desirable or if a focus on specialized inference methods for particular formalisms is more beneficial.

Acknowledgements

This work is an improved version of earlier work that appeared as a poster [18]. We are grateful for helpful comments from our previous coauthors Stefan Wölfl

and Jason Li on work done for the poster. Further, we thank reviewers for suggestions and comments. This work was supported by DFG (Transregional Collaborative Research Center *SFB/TR 8 Spatial Cognition*, project R4-[LogoSpace]).

References

1. Nebel, B.: Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-horn class. Constraints **1**(3) (1997) 175–190
2. Renz, J., Nebel, B.: Efficient methods for qualitative spatial reasoning. Journal of Artificial Intelligence Research (JAIR) **15** (2001) 289–318
3. Westphal, M., Wölfl, S.: Qualitative CSP, finite CSP, and SAT: Comparing methods for qualitative constraint-based reasoning. In Boutilier, C., ed.: IJCAI 2009. (2009) 628–633
4. Li, J.J., Huang, J., Renz, J.: A divide-and-conquer approach for solving interval algebra networks. In Boutilier, C., ed.: IJCAI 2009. (2009) 572–577
5. van Beek, P.: Backtracking search algorithms. In Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Elsevier (2006)
6. Katsirelos, G., Bacchus, F.: Generalized nogoods in CSPs. In Veloso, M.M., Kambhampati, S., eds.: AAAI 2005, AAAI Press / The MIT Press (2005) 390–396
7. Ligozat, G., Renz, J.: What is a qualitative calculus? A general framework. In Zhang, C., Guesgen, H.W., Yeap, W.K., eds.: PRICAI 2004. Volume 3157 of LNCS., Springer (2004) 53–64
8. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM **26**(11) (1983) 832–843
9. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In Nebel, B., Rich, C., Swartout, W.R., eds.: KR 1992. (1992) 165–176
10. Bodirsky, M., Kára, J.: A fast algorithm and datalog inexpressibility for temporal reasoning. ACM Trans. Comput. Log. **11**(3) (2010)
11. Condotta, J.F., Ligozat, G., Saade, M.: Eligible and frozen constraints for solving temporal qualitative constraint networks. In Bessière, C., ed.: CP 2007. Volume 4741 of LNCS., Springer (2007) 806–814
12. Lecoutre, C., Saïs, L., Tabary, S., Vidal, V.: Recording and minimizing nogoods from restarts. Journal on Satisfiability, Boolean Modeling and Computation **1**(3-4) (2007) 147–167
13. Westphal, M., Wölfl, S., Gantner, Z.: GQR: A fast solver for binary qualitative constraint networks. In: Proceedings of the AAAI'09 Spring Symposium on Benchmarking of Qualitative Spatial and Temporal Reasoning Systems. (2009)
14. van Beek, P., Manchak, D.W.: The design and experimental analysis of algorithms for temporal reasoning. Journal of Artificial Intelligence Research **4** (1996) 1–18
15. Boussemart, F., Hemery, F., Lecoutre, C., Saïs, L.: Boosting systematic search by weighting constraints. In de Mántaras, R.L., Saitta, L., eds.: ECAI 2004, IOS Press (2004) 146–150
16. Bessière, C.: Constraint propagation. In Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Elsevier (2006)
17. Eén, N., Sörensson, N.: An extensible SAT-solver. In Giunchiglia, E., Tacchella, A., eds.: Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003, Selected Papers. Volume 2919 of LNCS., Springer (2003) 502–518
18. Westphal, M., Wölfl, S., Li, J.J.: Restarts and nogood recording in qualitative constraint-based reasoning. In Coelho, H., Studer, R., Wooldridge, M., eds.: ECAI 2010, IOS Press (2010) 1093–1094