Qualitative CSP, Finite CSP, and SAT: Comparing Methods for Qualitative Constraint-based Reasoning

Matthias Westphal and **Stefan Wölfl**
Department of Computer Science, University of Freiburg
Georges-Köhler-Allee, 79110 Freiburg, Germany
{westpham, woelfl}@informatik.uni-freiburg.de

Abstract

Qualitative Spatial and Temporal Reasoning (QSR) is concerned with constraint-based formalisms for representing, and reasoning with, spatial and temporal information over infinite domains. Within the QSR community it has been a widely accepted assumption that genuine qualitative reasoning methods outperform other reasoning methods that are applicable to encodings of qualitative CSP instances. Recently this assumption has been tackled by several authors, who proposed to encode qualitative CSP instances as finite CSP or SAT instances. In this paper we report on the results of a broad empirical study in which we compared the performance of several reasoners on instances from different qualitative formalisms. Our results show that for small-sized qualitative calculi (e.g., Allen's interval algebra and RCC-8) a state-of-the-art implementation of QSR methods currently gives the most efficient performance. However, on recently suggested large-size calculi, e.g., $\mathcal{OPRA}_4$, finite CSP encodings provide a considerable performance gain. These results confirm a conjecture by Bessière stating that support-based constraint propagation algorithms provide better performance for large-sized qualitative calculi.

1 Introduction

Qualitative reasoning is concerned with representation formalisms that are considered close to conceptual schemata used by humans for reasoning about their physical environment—in particular, about processes or events and about the spatial environment in which they are situated. The approach in qualitative reasoning is to develop relational schemas that abstract from concrete metrical data of entities (for example, time points, coordinate positions, distances) by subsuming similar (geo-) metric or topological configurations of entities into one qualitative representation. Given a fixed granularity, one identifies a set of relations to be used in disjunctive qualitative statements, which express indefinite and imprecise knowledge about the application domain. At the heart of qualitative reasoning is solving of constraint network over infinite domains by reasoning with so-called composition tables. Composition tables encode semantic, i.e., domain-specific information about possible (spatial or temporal) configurations between two entities if information is available about how these entities are related to some third entity. An adaption of the path consistency algorithm used in the finite CSP domain allows for successively refining the information between each pair of entities under consideration. For many qualitative formalisms, the path consistency method combined with backtracking search techniques provides an (NP-complete) decision procedure for determining whether a given qualitative description is satisfiable.

The need for genuine qualitative methods has been questioned by Brand [2004], who pointed out several advantages of a "(qualitative) relations as variables"-approach: the idea here is to solve problem instances in qualitative reasoning by encoding them as finite constraint satisfaction problems. Moreover, in a more recent empirical study [Pham *et al.*, 2006], it has been shown that in the case of temporal reasoning, SAT solvers applied to a suitable encoding of qualitative temporal reasoning instances can outperform qualitative reasoners such as the reasoner by Nebel [1997].

The aim of the paper is to analyze the benefits of genuine qualitative reasoning methods. For this, we will compare the reasoning performance of the Generic Qualitative Reasoner (GQR) [Gantner *et al.*, 2008], which implements the state-of-the-art techniques in QSR, against the reasoning performance of automated reasoning tools from the CSP and SAT domain, namely the CSP solver Mistral [Hebrard, 2008] and the SAT solver MiniSat [Eén and Sörensson, 2003]. This will enable us to compare established QSR algorithms to those used in the finite CSP and SAT domain.

The outline of this paper is as follows: in the next section we will sketch the usual representation and reasoning methods used in QSR. Section 3 deals with encoding schemes used for translating qualitative constraint networks into finite CSP and SAT instances. In section 4 we outline some theoretical considerations regarding the different representation formalisms, which will help to understand and compare the test results; these will then be reported on in section 5. The summary and outlook in section 6 completes the paper.

2 Formal Background

In what follows let D be a fixed non-empty set. A *relation system* $\mathcal{B}$ over D, is any non-empty set of relations de-

fined on D. Given a relation system $\mathcal{B}$, a *constraint network* over $\mathcal{B}$ is defined by a triple $N = \langle V, D, C\rangle$, where V is a finite set of variables, D is a non-empty set of values for the variables in V, and C is a finite set of constraints, i.e., pairs (s, R), where $s = (v_1, \ldots, v_{n_s})$ is a non-empty sequence of variables and R is an n_s-ary relation in $\mathcal{B}$. W. l. o. g., we assume that constraint networks contain for any set of variables $\{v_1, \ldots, v_n\}$ at most one constraint with scope $s = (v_1, \ldots, v_n)$. A *solution* of a constraint network $\langle V, D, C\rangle$ is a function $*^I \colon V \to D$ such that for each constraint (s, R) in C, $s^I := (v_1^I, \ldots, v_{n_s}^I) \in R$. A constraint network is said to be *satisfiable* if it has a solution.

While in the CSP field only problem instances with *finite* variable domains are considered, constraint networks in QSR are defined on infinite domains, which entails that typical reasoning techniques known from the CSP domain (such as arc consistency or path consistency) are no longer directly applicable. Hence, the idea in QSR is to restate constraint satisfaction problems on a symbolic level such that reasoning about specific variable assignments can be replaced by manipulation of symbol sets. To explain this further, we define:

Def. 1 ([Ligozat and Renz, 2004])**.** A *partition scheme* on a domain D is a finite set, $\mathcal{B}$, of binary relations on D that forms a partition of $D \times D$, contains the diagonal (or identity) relation $\{(x, x) : x \in D\}$, and is closed under converses (i.e., $B^{-1} := \{(y, x) : (x, y) \in B\} \in \mathcal{B}$ for $B \in \mathcal{B}$). The elements of $\mathcal{B}$ are referred to as *base relations* of the partition scheme.

Let $\mathcal{B}^*$ denote the relation system consisting of all possible unions of base relations from some fixed partition scheme $\mathcal{B}$ on D. A *(binary) qualitative constraint network* is a constraint network over $\mathcal{B}^*$ for some partition scheme $\mathcal{B}$.

Let $(S_B)_{B\in\mathcal{B}}$ be symbols to denote the relations in the partition scheme $\mathcal{B}$. We use a set notation to denote unions of base relations (elements of $\mathcal{B}^*$), that is, constraints in a qualitative constraint network can be written in the form

$$\langle (x, y), \{S_{B_1}, \ldots, S_{B_k}\}\rangle \tag{1}$$

for relations $B_1, \ldots, B_k \in \mathcal{B}$. Moreover, we can represent a constraint network $N = \langle V, D, C\rangle$ on the symbolic level by its *primary constraint graph* $G = \langle V, l\rangle$, where V is just the sets of variables of N and $l \colon V \times V \to 2^{\{\, S_B \,:\, B\in\mathcal{B}\,\}}$ is the partial function that assigns to each constraint scope (x, y) the symbol set $\{S_{B_1}, \ldots, S_{B_k}\}$, where $B_1 \cup \cdots \cup B_k$ is the constraint relation between x and y. To turn l into a total function, we assign to each variable pair (x, y), which is not the scope of a constraint, the set of all relation symbols (which denotes the universal binary relation on D).

Note that we do not require that $\mathcal{B}$ is closed under composition, i.e., we do not enforce that $B \circ B' \in \mathcal{B}^*$ for arbitrary $B, B' \in \mathcal{B}$. Instead, composition of concrete binary relations is replaced by a symbolic approximation (*weak composition*)

$$S_B \,;\, S_{B'} := \{\, S_{B''} \,:\, B'' \cap (B \circ B') \neq \emptyset \,\}.$$

For $R \in \mathcal{B}^*$ with $R = B_1 \cup \cdots \cup B_m$, we set $S_R := \{S_{B_1}, \ldots S_{B_m}\}$. Then we can define two operations on the set $2^{\{\, S_B \,:\, B\in\mathcal{B}\,\}}$, namely symbolic converse $S_R^{\smile} := \{S_{B^{-1}} : B \in R\}$ and symbolic composition $S_R \,;\, S_{R'} := \bigcup_{B\in R, B'\in R'} S_B \,;\, S_{B'}$. These two operations (restricted to base relations of a partition scheme) provide the important information that is fed into a qualitative reasoner in order to specify a qualitative formalism (also often referred to as *qualitative calculus*).

A constraint graph (V, l) is said to be *path-consistent* (or: *algebraically closed*) if (a) no label is empty and (b) $l(x, y) \subseteq l(x, z) \,;\, l(z, y)$ for each triple of variables x, y, z in V. Note that this notion is weaker than the notion of path consistency known from finite CSP that requires that each two-variable assignment consistent with the constraint network can be extended to a consistent three-variable assignment.

A constraint graph (V, l) is said to be a *refinement* of a constraint graph (V, l') if $l(x, y) \subseteq l'(x, y)$ for each pair of variables from V. Each constraint graph can be refined (in polynomial time) into a constraint graph, which is path-consistent or inconsistent (that is, it has empty labels). This can be achieved if we successively refine each of the labels $l(x, y)$ (for variables x and y) by applying the operation

$$l(x, y) \leftarrow l(x, y) \cap (l(x, z) \,;\, l(z, y)), \tag{2}$$

where z is any third variable occurring in the network. Implementations of this method are usually based on some variant of Mackworth's path consistency algorithm (see, e.g., [Bessière, 1996]), which uses queues to store those arcs or triangles in the network that need to be reprocessed due to previous refinements of the network.

Def. 2. Let $\mathcal{B}$ be a partition scheme. A subset $\mathcal{B}'$ of $\mathcal{B}^*$ is said to be a *tractable subclass* if the path consistency method applied to constraint networks over $\mathcal{B}'$ decides satisfiability.

If the path consistency method decides satisfiability of constraint networks with only base relations from a partition scheme $\mathcal{B}$, a constraint network over $\mathcal{B}^*$ is satisfiable if and only if it has a path-consistent refinement over $\mathcal{B}$. By using backtracking search methods, one can systematically try out different atomic refinements of a given constraint graph (a refinement is *atomic* if all labels are singleton sets) and check them for satisfiability. Moreover, by using tractable subclasses of a relation system, one can speed up the reasoning time: instead of splitting a constraint during backtracking into base relations, one can split it into relations belonging to a tractable subclass, which reduces the branching factor of the search tree considerably [Nebel, 1997].

3 Encodings of Qualitative CSP

In this section we discuss encodings of qualitative reasoning problems, first as an equivalent finite CSP problem and then as a SAT instance (i.e., as propositional CNF formula). Both encodings make essential use of the composition table of the qualitative formalism, that is, the function $; : \mathcal{B} \times \mathcal{B} \to 2^{\mathcal{B}}$ (here and in what follows we identify relations and their symbols). Certainly there is a variety of different, equivalent encodings such that we need to restrict our considerations to one encoding for both finite CSP and SAT. In particular, we do not consider a specific encoding for each calculus, but rather look at the general problem of reasoning with any (binary) qualitative constraint calculus, i.e., we are interested in algorithms for the general case and not in specific domain-tailored encodings.

3.1 Finite CSP Encoding

Qualitative constraint networks can be encoded as finite CSP instances [Renz and Nebel, 2001; Brand, 2004; Condotta *et al.*, 2006]. Given the constraint graph of a qualitative constraint problem $\langle V, l\rangle$ with $V = \{v_1, \ldots, v_n\}$, we obtain a finite constraint satisfaction problem by using a variant of the so-called *dual constraint problem*: Let X be a set of variables containing a variable x_{ij} for each pair of variables $v_i, v_j \in V$ with $i < j$. Then the dual problem has the form:

$$\langle X, \mathcal{B}, \{C_{ij} : 1 \le i < j \le n\} \cup \{C_{ijk} : 1 \le i < j < k \le n\}\rangle$$

where C_{ij} is a domain constraint restricting the values of x_{ij} to the elements of $l(v_i, v_j)$ and C_{ijk} is a binary constraint that enforces variables x_{ij}, x_{jk} ("qualitative edges") to assignments, where v_j takes the same value. However, on infinite domains this scheme does not work. Therefore, instead of these binary constraints, we use a set of ternary constraints

$$TC = \{\, ((x_{ij}, x_{ik}, x_{kj}), R_{;}) \,:\, 1 \le i < j < k \le n \,\}$$

where $R_{;} := \{(B'', B, B') \in \mathcal{B}^3 : B'' \in B \,;\, B'\}$. The resulting finite network has $\frac{n\cdot(n-1)}{2}$ variables and $\binom{n}{3} = \frac{(n-1)^3-(n-1)}{6}\, TC$ constraints.

A solution of this finite CSP instance corresponds to an atomic, path-consistent refinement of the qualitative constraint network, and vice versa [Condotta *et al.*, 2006]. This means that on the symbolic level consistent (not necessarily satisfiable) qualitative CSP instances correspond to consistent (satisfiable) finite CSP instances (note that in general a qualitative constraint network needs not be satisfiable even if it has an atomic and path-consistent refinement).

3.2 SAT Encoding

Various different encodings of CSP into SAT instances have been proposed in the literature (see e.g., [Walsh, 2000]). Here we will restrict consideration to the *1D support scheme* introduced by Pham et al. [2006] for qualitative temporal networks. They show that the 1D support scheme gives better results than direct and log encodings for Allen's interval calculus. It can be sketched as follows: Given the finite CSP encoding of a qualitative constraint network

$$\big\langle X, (l(v_i, v_j))_{x_{ij}\in X}, TC\big\rangle \tag{3}$$

(where now the unary constraints are written as variable dependent domains), we introduce propositional variables x_{ij}^B for each $x_{ij} \in X$ and $B \in l(v_i, v_j)$. To represent the disjunctive nature of the constraints, we add the clauses

$$x_{ij}^{B_1} \vee \cdots \vee x_{ij}^{B_k}$$

for each x_{ij} (where $l(v_i, v_j) = \{B_1, \ldots, B_k\}$) in order to assert that at least one value is assigned to x_{ij}, and

$$\neg(x_{ij}^{B_k} \wedge x_{ij}^{B_l}) \quad \Big(\equiv \neg x_{ij}^{B_k} \vee \neg x_{ij}^{B_l}\Big)$$

for each x_{ij} and distinct $B_k, B_l \in l(v_i, v_j)$ to have at most one value assigned. For the TC constraints we add clauses

$$(x_{ij}^{B_l} \wedge x_{jk}^{B_m}) \to (x_{ik}^{B_1} \vee \cdots \vee x_{ik}^{B_n}) \qquad \Big(\equiv \neg x_{ij}^{B_l} \vee \neg x_{jk}^{B_m} \vee x_{ik}^{B_1} \vee \cdots \vee x_{ik}^{B_n}\Big)$$

for all variables x_{ij}, x_{jk}, x_{ik} $(i < j < k)$ and relations $B_l \in l(v_i, v_j), B_m \in l(v_j, v_k)$, where $\{B_1, \ldots, B_n\} = (B_l;B_m)\cap l(v_i, v_k)$.

This scheme gives a propositional formula that is equivalent to the finite CSP instance. It should be mentioned that the generated formulas are quite large. For a qualitative CSP with n variables we obtain a propositional formula with $O(n^2 \cdot b)$ variables and $O(n^3 \cdot b^2)$ clauses. For the empirical tests presented later we preprocessed constraint networks using the path consistency algorithm (as suggested by Pham et al.) to reduce the domain sizes of variables in the finite CSP, before encoding it into the SAT formula. Still, some of our instances were as large as 600 MiB.

4 A Theoretical View

Before we present our empirical results, we discuss some theoretical background on the considered reasoning approaches.

4.1 CSP Solvers vs. Qualitative Constraint Solvers

As has been pointed out by Condotta et al. [2006], generalized arc consistency (GAC) is equivalent to path consistency in QSR, when applied to the finite CSP encoding of a qualitative CSP instance. Hence, any CSP solver that uses generalized arc consistency as constraint propagation achieves the same pruning as a typical qualitative solver. Since the efficiency of the constraint propagation is crucial for any constraint solver, we consider it useful to compare the path consistency algorithm from QSR to established GAC-variants. Note that, if we assume the same pruning behavior and neglect tractable subclasses, the explored search space is, in principle, the same.

The path consistency algorithm from QSR can be considered a coarse-grained arc consistency algorithm on the encoded qualitative CSP. Such coarse-grained arc consistency algorithms consist of a *main loop* that stores and iterates over a queue of unprocessed variable-constraint pairs and a *revise* function, which prunes those values from the domain of the variable that have no *support*, i.e., that cannot be part of a satisfying assignment of this constraint, given the possible values of the other involved variables.

Typically, a revise function performs a loop over the values of the domain and checks for a support. Contrary to this, the path consistency algorithm from QSR requires to compute the refinement function (2) which *in the end* results in a removal of unsupported values, but might perform unnecessary computations while calculating the ;-composition in (2). Bessière [1996] has shown that a support-based revise function is more efficient than a naïve implementation of the ;-function for Allen's interval calculus. Furthermore, Bessière conjectures that the difference in performance grows even more for partition schemes with considerably more base relations. But there are other means by which one can speed up the computation of compositions, without switching to a support-based concept (e.g., via precomputation of non-atomic composition results [Ladkin and Reinefeld, 1997]). In the following, we give their complexities in comparison to GAC variants: let e be the number of edges in the finite CSP, b be the number of base relations in the partition scheme, and d

be the maximal domain size in the finite CSP (as given in the form of equation (3)), i.e., $d \leq b$. As previously discussed, e corresponds to $\binom{n}{3} \in O(n^3)$, where n is the number of variables in the considered qualitative CSP instance. For the constraint propagation, we can give the following worst case complexities for the encoded finite CSP networks [Bessière, 2006] (note that the arity of the finite constraints is 3):

Constraint propagation	Complexity
GAC3	$O(e \cdot d^4)$
GAC2001	$O(e \cdot d^3)$
Path consistency (naïve)	$O(e \cdot b^4)$
Path consistency (precomputed)	$O(e \cdot b^2)$

The naïve computation of the ;-function requires $O(b^3)$ time, whereas the one based on precomputations only needs a table lookup at the expense of an additional exponential space requirement in the number of base relations. That is, if it is feasible to precompute the ;-function, the revision (2) is quite efficient and, in particular, less complex than GAC3 and GAC2001 in the worst case. Without precomputation, the worst case complexity is equal to that of GAC3, but worse than GAC2001 and in the average case (most likely) worse than GAC3 and its residual variant GAC3r.

4.2 SAT Solvers vs. Qualitative Constraint Solvers

For a reasonable comparison of SAT and QSR we would be interested in comparing qualitative constraint networks to clauses. Since presumably there is no general useful way to convert clauses to a constraint network with relations from a given qualitative formalism, a theoretical comparison becomes quite difficult, especially if one takes into account that most SAT solvers perform additional learning and restart techniques. For this reason, we only refer to the results concerning constraint propagation in the CSP and SAT domain (e.g., [Dimopoulos and Stergiou, 2006; Walsh, 2000]). However, it should not go unmentioned that the SAT encoding approach only encodes the instance-specific parts of a qualitative calculus (used base relations, composition table entries) and hence avoids storing irrelevant data, in contrast to qualitative reasoners which usually work on the whole formalism.

5 Empirical Analysis

5.1 Reasoning Systems

CSP. For a CSP solver, we used *Mistral* [Hebrard, 2008], a library for modeling and solving constraint problems. Bundled with a front-end for reading and modeling problems, it ranked fifth in the 2008 CSP competition. It was the highest ranking, available solver which is capable of handling ternary constraints and not portfolio-based. Mistral utilizes heuristic depth-first search (*domain over weighted-by-level degree*) with generalized arc consistency via GAC3r or GAC2001. The `mistral-prime` version of Mistral for the CSP solver competition 2008 was used for our tests.

SAT. We used the well-known SAT solver *MiniSat* [Eén and Sörensson, 2003] for our evaluation, since in our tests it was the only SAT solver that was able to cope with the size of very large problem instances. The most recent available version (at the time of writing) MiniSat 2.0 beta was used.

QSR. For a typical qualitative reasoner, we used the generic qualitative reasoner[1] (GQR) [Gantner *et al.*, 2008] version 1089, in which the heuristic for "qualitative edges" is based on dynamic weights as a boost [Boussemart *et al.*, 2004], combined with static (problem-independent) weights of relations [Renz and Nebel, 2001]. GQR chooses an edge with minimal proportion $\frac{staticweight}{dynamicweight}$. When a relation is reduced to the empty relation, dynamic weights are increased based on the depth of the search tree (exactly as in Mistral). The selection of values (or sub-relations) is based on static weights with 2-way branching. Further, last conflict-based reasoning is performed to reduce the amount of thrashing [Lecoutre *et al.*, 2006]. In the following GQR will refer to a version not exploiting any tractable subclasses, which allows us to directly compare the number of visited search space nodes to that of Mistral. GQR* will refer to a version taking advantage of tractable subclasses, if such are known. It assigns tractable subsets of relations as values and uses eligible constraints [Condotta *et al.*, 2007] to reduce the search space.

5.2 Test Setup

Since there are only few data from concrete applications publicly available that could be used as natural datasets for benchmark purposes, our empirical analysis is based on randomly generated problem instances in various qualitative formalisms. We used problem generators as described in [Nebel, 1997; Renz and Nebel, 2001]. Given a qualitative constraint calculus, a qualitative constraint network is calculated in dependency of parameters n, d, s as follows: A random graph is generated with n nodes and an average degree of d. This graph is then transformed into a qualitative CSP by assigning a non-empty relation from the qualitative partition scheme randomly to each edge. If an edge is not present in the graph, it is introduced and assigned the universal relation. In the so-called A-model the random relations are taken from a binomial distribution such that, on average, a relation consists of s base relations. For the H-model, each generated relation is checked against a given list of allowed relations, and consequently discarded if it is not included in the list until an allowed relation is found.

To reduce the number of experiments, we fixed the number of nodes beforehand, set the label-size parameter s to a reasonable value, and searched for the degree d where the runtime of GQR peaked. Further, since the SAT encoding scheme uses path-consistent qualitative constraint networks as input, we removed trivial instances that are already found to be unsatisfiable when path consistency is applied (these models are referred to as A'- and H'-model).

We considered the following calculi: (a) Region Connection Calculus RCC-8 (8 base relations) [Randell *et al.*, 1992], (b) Allen's Interval Calculus, Allen (13 b.r.) [Allen, 1983], (c) RCC-23 (23 b.r.) [Bennett, 1997], (d) $\mathcal{OPRA}_2$ calculus (72 b.r.), and (e) $\mathcal{OPRA}_4$ calculus (272 b.r.) [Moratz, 2004].

All experiments were conducted on an Intel Xeon CPU

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

Calculus	Model	GQR*	GQR	Mistral	MiniSat
RCC-8	$A'(100, 10.5, 4.0)$	500/500/500 $0.11s$/$0.11s$	500/500/500 $1.67s$/$1.67s$	500/500/500 $7.19s$/$6.93s$	491/500/500 $126.36s$/$112.24s$
RCC-8	$H'(100, 15.5, 4.0)$	496/500/500 $19.53s$/$2.2s$	485/497/498 $33.05s$/$3.27s$	309/377/426 $291.42s$/$96.56s$	499/500/500 $50.25s$/$23.86s$
RCC-8	$A'(125, 10.5, 4.0)$	500/500/500 $0.27s$/$0.26s$	500/500/500 $4.19s$/$4.18s$	500/500/500 $18.36s$/$18.83s$	5/59/254 $1195.75s$/$1240.32s$
RCC-8	$H'(125, 15.5, 4.0)$	406/452/480 $168.48s$/$28.55s$	345/399/428 $203.74s$/$56.59s$	92/146/186 $518.73s$/$302.75s$	199/458/499 $572.10s$/$513.52s$
Allen	$A'(100, 10.5, 6.5)$	482/494/500 $50.8s$/$3.66s$	417/456/469 $109.76s$/$9.59s$	210/282/311 $309.58s$/$157.12s$	383/471/495 $234.92s$/$132.55s$
RCC-23	$A'(100, 73.0, 9.0)$	- -	411/441/449 $77.70s$/$5.625s$	297/360/379 $215.06s$/$74.86s$	194/197/225 $217.92s$/$28.06s$
$\mathcal{OPRA}_2$	$A'(15, 14.0, 20.0)$	- -	31/75/128 $745.58s$/$708.77s$	86/161/232 $648.72s$/$471.95s$	57/161/252 $728.36s$/$605.90s$
$\mathcal{OPRA}_4$	$A'(10, 9.0, 26.0)$	- -	500/500/500 $50.75s$/$36.24s$	500/500/500 $11.03s$/$9.64s$	427/500/500 $146.96s$/$102.48s$

Table 1: Reasoners, solved instances and required time. An entry "1/10/100" indicates the number of solved instances in 5/15/30 min. Entries "$0.11s$/$0.11s$" refer to the average / median CPU time required to solve an instance. 500 instances were considered per model.

Calculus	Model	GQR*	GQR	Mistral	MiniSat
RCC-8	$A'(100, 10.5, 4.0)$	114.66/114 $0.11s$/$0.11s$	2260.23/2259 $1.67s$/$1.67s$	2328.90/2317 $7.19s$/$6.93s$	- $126.36s$/$112.24s$
RCC-8	$H'(100, 15.5, 4.0)$	1537.26/409 $6.25s$/$1.50s$	3360.67/778 $11.86s$/$2.16s$	1645.59/704 $291.42s$/$96.56s$	- $40.70s$/$18.32s$
Allen	$A'(100, 10.5, 6.5)$	802.27/249 $4.77s$/$1.27s$	5570.97/700 $28.04s$/$3.09s$	1049.93/567 $309.58s$/$157.12s$	- $107.54s$/$53.89s$
RCC-23	$A'(100, 73.0, 9.0)$	- -	1287.58/47 $34.52s$/$1.98s$	908.38/198 $139.21s$/$46.18s$	- $87.01s$/$24.55s$
$\mathcal{OPRA}_2$	$A'(15, 14.0, 20.0)$	- -	285130.47/262236 $726.55s$/$671.40s$	357949.40/213099 $334.99s$/$232.80s$	- $522.93s$/$434.23s$
$\mathcal{OPRA}_4$	$A'(10, 9.0, 26.0)$	- -	6107.09/4291 $50.75s$/$36.24s$	2961.12/2220 $11.03s$/$9.64s$	- $146.96s$/$102.48s$

Table 2: Selected results for average/median values of visited nodes and CPU time over instances solved by all reasoners.

with a CPU time and memory limit of 30 min and 4GiB, respectively, for each instance and each reasoner.

5.3 Empirical Results

Table 1 shows the number of solved instances for each calculus and further the required CPU time, while Table 2 provides an overview of the number of expanded search nodes.

For RCC-8, the A'-model provided only satisfiable instances, which is very probable if one aims for "hard" instances [Renz and Nebel, 2001]. Consequently, we also used the H'-model with $\mathcal{NP}_8$ as set of allowed relations. For $\mathcal{NP}_8$, the H'-model generates significantly harder instances for the path consistency algorithm [Renz and Nebel, 2001], that is, the pruning power of the path consistency algorithm is reduced simply by construction. This is reflected in the performance of GQR and Mistral in our test results. MiniSat, however, shows a significantly better performance on these instances. To test whether this was due to the fact that A'-instances were all satisfiable, we increased the node size (from 100 to 125) without altering the average degree, thus generating larger but more likely satisfiable instances in the case of H'. The tests show that MiniSat has a robust behavior for H', solving more instances than GQR*, but fails on most instances in the A'-model.

With Allen's interval calculus GQR performs significantly better than Mistral and solves more instances than MiniSat within the first five minutes. However, MiniSat solves more instances than GQR and is only surpassed by GQR*.

RCC-23 is the first calculus where no tractable subclasses are known and GQR does not use precomputed functions, i.e., GQR has to (re)compute the ;-function every time. Still, GQR outperforms Mistral, which in turn solved more instances than MiniSat. This, however, changes when we consider $\mathcal{OPRA}_2$. Here, GQR is slower than both MiniSat and Mistral. MiniSat shows the better performance when considering a 30 minute time window.

For $\mathcal{OPRA}_4$ we had to reduce the number of nodes, as otherwise the problems become far too hard. Mistral outperforms GQR even more than in the case of $\mathcal{OPRA}_2$. Interestingly, GQR outperforms MiniSat, contrary to what could be expected from the results of $\mathcal{OPRA}_2$. For $\mathcal{OPRA}_2$ and $\mathcal{OPRA}_4$ the support-based approach of Mistral seems to be more efficient than GQR's ;-function. It is furthermore evident that Mistral's heuristics are better in the average case.

6 Conclusions and Future Work

In this paper we have addressed the recently renewed interest in encoding qualitative reasoning problems as finite CSP or SAT instances. Our results show that a state-of-the-art implementation of QSR methods currently gives the most efficient performance for classical, small-sized qualitative calculi, such as e.g., Allen's interval algebra. From a theoretical point of view, path consistency combined with precomputation is better than a support-based approach like GAC2001 in the worst case. An implemented qualitative solver like GQR, without support-based revision, is still effective for up to at least 23 base relations, if compared to a general CSP solver. But our results also confirm Bessière's claim concerning the advantage of support-based revision for more recently discussed large-sized calculi, such as $\mathcal{OPRA}_4$. Our results indicate that it is useful for qualitative reasoners to borrow from recent achievements in finite CSP techniques, such as heuristics and constraint propagation algorithms. Since we evaluated complete reasoning systems, it would further be interesting to evaluate the components of the systems (e.g., constraint propagation) to improve both qualitative and finite CSP solvers.

The SAT approach provides robust results for specific hard instances, particularly where the path consistency algorithm is known to give bad estimates of satisfiability. In general, however, it is much more resource consuming and highly dependent on the used encoding scheme—at least the one considered in this paper seems unsuited for large, but simple problem instances.

Acknowledgments

This work was partially supported by Deutsche Forschungsgemeinschaft (Transregional Collaborative Research Center *SFB/TR 8 Spatial Cognition*, project R4-[LogoSpace]).

References

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[Bennett, 1997] Brandon Bennett. *Logical Representations for Automated Reasoning about Spatial Relationships*. PhD thesis, The University of Leeds, 1997.

[Bessière, 1996] Christian Bessière. A simple way to improve path consistency processing in interval algebra networks. In *AAAI/IAAI, Vol. 1*, pages 375–380, 1996.

[Bessière, 2006] Christian Bessière. Constraint propagation. Technical Report LIRMM 06020, CNRS/University of Montpellier, 2006.

[Boussemart *et al.*, 2004] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In *ECAI 2004*, pages 146–150. IOS Press, 2004.

[Brand, 2004] Sebastian Brand. Relation variables in qualitative spatial reasoning. In *KI 2004*, LNCS: 3238, pages 337–350. Springer, 2004.

[Condotta *et al.*, 2006] Jean-François Condotta, Dominique D'Almeida, Christophe Lecoutre, and Lakhdar Sais. From qualitative to discrete constraint networks. In *Workshop on Qualitative Constraint Calculi held with KI 2006*, pages 54–64, Bremen, Germany, 2006.

[Condotta *et al.*, 2007] Jean-François Condotta, Gérard Ligozat, and Mahmoud Saade. Eligible and frozen constraints for solving temporal qualitative constraint networks. In *CP 2007*, LNCS: 4741, pages 806–814. Springer, 2007.

[Dimopoulos and Stergiou, 2006] Yannis Dimopoulos and Kostas Stergiou. Propagation in CSP and SAT. In *CP 2006*, LNCS: 4204, pages 137–151. Springer, 2006.

[Eén and Sörensson, 2003] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *SAT 2003*, LNCS: 2919, pages 502–518. Springer, 2003.

[Gantner *et al.*, 2008] Zeno Gantner, Matthias Westphal, and Stefan Wölfl. GQR – A fast reasoner for binary qualitative constraint calculi. In *AAAI-08 Workshop on Spatial and Temporal Reasoning, Chicago, USA, AAAI*, 2008.

[Hebrard, 2008] Emmanuel Hebrard. Mistral, a constraint satisfaction library. In *Proceedings of the Third International CSP Solver Competition*, 2008.

[Ladkin and Reinefeld, 1997] Peter B. Ladkin and Alexander Reinefeld. Fast algebraic methods for interval constraint problems. *Ann. Math. Artif. Intell.*, 19(3-4):383–411, 1997.

[Lecoutre *et al.*, 2006] Christophe Lecoutre, Lakhdar Sais, Sébastien Tabary, and Vincent Vidal. Last conflict based reasoning. In *ECAI 2006*, pages 133–137. IOS Press, 2006.

[Ligozat and Renz, 2004] Gérard Ligozat and Jochen Renz. What is a qualitative calculus? A general framework. In *PRICAI 2004*, LNCS: 3157, pages 53–64. Springer, 2004.

[Moratz, 2004] Reinhard Moratz. Qualitative spatial reasoning about oriented points. Technical Report 003/10-2004, SFB/TR 8 Spatial Cognition, 2004.

[Nebel, 1997] Bernhard Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-horn class. *Constraints*, 1(3):175–190, 1997.

[Pham *et al.*, 2006] Duc Nghia Pham, John Thornton, and Abdul Sattar. Towards an efficient SAT encoding for temporal reasoning. In *CP 2006*, LNCS: 4204, pages 421–436. Springer, 2006.

[Randell *et al.*, 1992] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *KR 1992*, pages 165–176, 1992.

[Renz and Nebel, 2001] Jochen Renz and Bernhard Nebel. Efficient methods for qualitative spatial reasoning. *J. Artif. Intell. Res. (JAIR)*, 15:289–318, 2001.

[Walsh, 2000] Toby Walsh. SAT v CSP. In *CP 2000*, LNCS: 1894, pages 441–456. Springer, 2000.