Efficient Methods for Qualitative Spatial Reasoning

Jochen Renz and **Bernhard Nebel**[1]

Abstract. The theoretical properties of qualitative spatial reasoning in the RCC-8 framework have been analyzed extensively. However, no empirical investigation has been made yet. Our experiments show that the adaption of the algorithms used for qualitative temporal reasoning can solve large RCC-8 instances, even if they are in the *phase transition* region – provided that one uses the maximal tractable subset of RCC-8 that has been identified by us. In particular, we demonstrate that the orthogonal combination of heuristic methods is successful in solving almost all apparently hard instances in the phase transition region up to a certain size in reasonable time.

1 Introduction

Representing qualitative spatial information and reasoning with such information is an important subproblem in many applications, such as natural language understanding, document interpretation, and geographical information systems. The RCC-8 calculus [15] is well suited for representing topological relationships between spatial regions. Inference in the full calculus is, however, NP-hard [5, 16]. While this means that it is unlikely that very large instances can be solved in reasonable time, this result does not rule out the possibility that we can solve instances up to a certain size in reasonable time.

One result of our experiments is that almost all instances with up to 80 regions can be solved in a few seconds, and this holds even for the very hard instances in the *phase transition* [3] region when only "difficult relations" are used. Another interesting result of our experiments is that the *orthogonal combination* of heuristics [13] is effective even on instances in the phase transition region.

The algorithmic techniques we use are borrowed from similar work on qualitative temporal reasoning [13, 18, 7]. Additionally, we make use of a fragment of RCC-8, named $\widehat{\mathcal{H}}_8$, that permits polynomial-time inferences [16]. In the backtracking algorithm, which is used to solve the reasoning problem for full RCC-8, branching is done according to this tractable subset instead of decomposing every disjunctive relation into its base relations, which reduces the average branching factor from 4.0 to 1.4375. Although these theoretical savings cannot be observed in our experiments, using $\widehat{\mathcal{H}}_8$ instead of the base relations leads to significant performance improvements.

The rest of the paper is structured as follows. In Section 2, we give a brief sketch of the RCC-8 calculus and describe the algorithmic techniques for solving the reasoning problems in RCC-8. In Section 3 we describe our experimental setting and in Section 4 we evaluate the different path-consistency methods on randomly generated instances. In Section 5, we determine the phase transition region and in Section 6 we evaluate different heuristics for the backtracking algorithm on instances from the phase transition region. Finally, in Section 7, we evaluate the performance of combining different heuristics.[2]

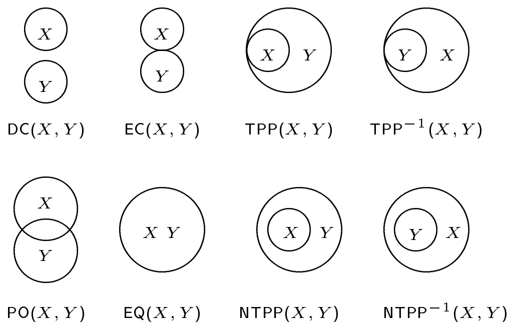

Figure 1. Two-dimensional examples for the eight base relations of RCC-8

2 The Region Connection Calculus RCC-8

RCC is a topological approach to qualitative spatial representation and reasoning where *spatial regions* are *regular subsets* of a topological space [15]. Regions themselves do not have to be internally connected, i.e., a region may consist of different disconnected parts. The domain of *spatial variables* (denoted as X, Y, Z) is the whole topological space.

RCC-8 uses a set of eight pairwise disjoint and mutually exhaustive relations, called *base relations*, denoted as DC, EC, PO, EQ, TPP, NTPP, TPP^{-1}, and NTPP^{-1}, with the meaning of *DisConnected, Externally Connected, Partial Overlap, EQual, Tangential Proper Part, Non-Tangential Proper Part*, and their converses. Examples for these relations are shown in Figure 1.

Sometimes it is not known which of the eight base relations holds between two regions, but it is possible to restrict to some of them. In order to represent this, unions of base relations can be used. Since base relations are pairwise disjoint, this results in 2^8 different relations. *Spatial formulas* are written as XRY, where R is a spatial relation. Apart from union ($\cup$), other

[1] Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Am Flughafen 17, D-79110 Freiburg, Germany

[2] The programs and a set of hard instances are available at http://www.informatik.uni-freiburg.de/~sppraum

© 1998 J.Renz and B.Nebel
ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

operations on our relations are defined, namely, converse ($\smile$), intersection ($\cap$), and composition ($\circ$). The formal definitions of these operations are:

$$\begin{array}{lcll}
\forall X,Y: & X(R\cup S)Y & \leftrightarrow & XRY \vee XSY, \\
\forall X,Y: & X(R\cap S)Y & \leftrightarrow & XRY \wedge XSY, \\
\forall X,Y: & XR^{\smile}Y & \leftrightarrow & YRX, \\
\forall X,Y: & X(R\circ S)Y & \leftrightarrow & \exists Z: (XRZ \wedge ZSY).
\end{array}$$

The composition of base relations can be computed from the semantics of the relations and is usually provided as a composition table [14, 2]. Based on this table, compositions of disjunctive relations can be easily computed. In the following $\widehat{\mathcal{S}}$ denotes the closure of a set of RCC-8 relations $\mathcal{S}$ under composition, intersection, and converse.

A set of spatial formulas Θ describing the topological relationship of n different regions can be represented by an $n\times n$ matrix M, where each entry M_{ij} represents the RCC-8 relation holding between region i and region j. Without loss of generality, $M_{ii} = \{\mathsf{EQ}\}$ and $M_{ji} = M_{ij}^{\smile}$ can be assumed. The fundamental reasoning problem (named RSAT) in this framework is deciding *consistency* of a set of spatial formulas Θ, i.e., whether there is a spatial configuration where the relations between the regions can be described by Θ. All other interesting reasoning problem can be reduced to it in polynomial time [4]. Unfortunately, RSAT is NP-complete [5, 16], i.e., it is unlikely that there is any polynomial algorithm for deciding consistency. However, it was shown in [12] that there are subsets $\mathcal{S}$ of RCC-8 for which the consistency problem (written RSAT($\mathcal{S}$)) can be decided in polynomial time. In particular the set of eight base relations $\mathcal{B}$ was shown to be tractable. From that it follows immediately that $\widehat{\mathcal{B}}$ consisting of 32 relations is also tractable. An even larger tractable subset containing all base relations is $\widehat{\mathcal{H}}_8$ [16], which contains 148 out of the 256 RCC-8 relations. This set was also shown to be maximal wrt. tractability, i.e., if any other RCC-8 relation is added, the consistency problem becomes NP-complete.

2.1 The Path-Consistency Algorithm

As in the area of qualitative temporal reasoning based on Allen's interval calculus [1], the *path-consistency algorithm* (PCA) [11, 9, 10] can be used to approximate consistency and to realize *forward-checking* [6] in a backtracking algorithm.

The algorithm checks the consistency of all triples of relations and eliminates relations that are impossible. This is done by iteratively performing the following operation

$$M_{ij} \leftarrow M_{ij} \cap M_{ik} \circ M_{kj}$$

for all triples of regions i, j, k until a fixed point $\overline{M}$ is reached. If $\overline{M}_{ij} = \emptyset$ for a pair i, j, then we know that M is inconsistent, otherwise $\overline{M}$ is called *path-consistent*. Computing $\overline{M}$ can be done in $O(n^3)$ time. This is achieved by using a *queue* of triples of regions for which the relations should be recomputed.

Path-consistency does not imply consistency. For instance, the following set of spatial constraints is path-consistent but not consistent:

$$\begin{tikzcd}
X \arrow[r, "\mathsf{TPP} \vee \mathsf{TPP}^{-1}"] \arrow[d, "\mathsf{DC} \vee \mathsf{TPP}"'] \arrow[dr, "\mathsf{EC} \vee \mathsf{TPP}"] & Y \arrow[d, "\mathsf{EC} \vee \mathsf{NTPP}"] \arrow[dl, "\mathsf{EC} \vee \mathsf{TPP}"] \\
Z \arrow[r, "\mathsf{EQ} \vee \mathsf{NTPP}"'] & W
\end{tikzcd}$$

If only relations in $\widehat{\mathcal{H}}_8$ are used, however, the path-consistency algorithm is sufficient for deciding consistency, i.e., PCA decides RSAT($\widehat{\mathcal{H}}_8$) [16].

In previous empirical investigations [18] of reasoning with Allen's interval relations [1], different methods for computing the composition of two relations were evaluated. In our setting, we simply use a composition table that specifies the compositions of all RCC-8 relations, which is a 256×256 table consuming approximately 128kB of main memory. This means that the composition of two arbitrary relations is done by a simple table lookup.

Van Beek and Manchak [18] also studied the effect of weighting the relations in the queue according to their restrictiveness and process the most restricting relation first. The reason for doing so is that this relation restricts the other relations on average most and therefore decreases the probability that they have to be processed again. Restrictiveness of a relation was approximated by summing up the restrictiveness of the involved base relations which is determined by counting the number of base relations contained in the corresponding entries in the basic composition table.

Van Beek and Manchak [18] found that this method is much more efficient than the usual path-consistency algorithm. Because of the small number of RCC-8 relations, we computed the exact restrictiveness by counting the base relations in the corresponding entries of the full composition table.

2.2 The Backtracking Algorithm

In order to solve arbitrary instances of RSAT, we have to search the corresponding search space using some sort of backtracking. In our experiments, we used a backtracking algorithm employed for solving qualitative temporal reasoning problems [13], which is based on the algorithm proposed by Ladkin and Reinefeld [7]. This algorithm uses PCA for forward checking and backtracks on decompositions of the specified relations into sub-relations that permit the PCA to decide the consistency problem. The set of sub-relations that is allowed in the decompositions is called *split-set*.

Choosing the right split-set influences the search noticeably as it influences the *average branching factor* of the search tree ($\mathcal{B}$: 4.0, $\widehat{\mathcal{B}}$: 2.50, $\widehat{\mathcal{H}}_8$: 1.4375), which is, of course, a worst case measure because the interleaved path-consistency computations reduce the branching factor considerably [8].

Apart from the choice of the split-set there are other heuristics which influence the efficiency of the search. In general it is the best search strategy to proceed with the most constraining relation and the least constraining choice. We investigated two different strategies for choosing the next relation to be processed.

static/dynamic: Constraints are processed according to a heuristic evaluation of their constrainedness which is determined *statically* before the backtracking starts or *dynamically* during the search.

local/global: The evaluation of the constrainedness is based on a *local* heuristic weight criterion or on a *global* heuristic criterion [18].

3 Test Instances and Measurement

There is no previous work on empirical evaluation of algorithms for reasoning with RCC-8 and no benchmark problems are known. Therefore we randomly generated our test instances with a given number of regions n, an average label-size l, and an average degree d of the constraint graph. Further, we used two different sets of relations for generating the instances, the set of all RCC-8 relations and the set of hard RCC-8 relations, i.e., those relations which are not contained in $\widehat{\mathcal{H}}_8$. Based on these sets of relations, we used two models to generate instances, denoted by $A(n, d, l)$ and $H(n, d, l)$. The former model uses all relations to generate instances, the latter only the relations in RCC-8 $- \widehat{\mathcal{H}}_8$. The instances are generated as follows:

1. A graph with n nodes and an average degree of d for each node is generated. This is accomplished by selecting $nd/2$ out of the $n(n-1)/2$ possible edges using a uniform distribution.
2. If there is no edge between the ith and jth node, we set $M_{ij} = M_{ji}$ to the universal relation.
3. Otherwise a non-universal constraint is selected according to the parameter l such that the average size of constraints for selected edges is l. This is accomplished by selecting one of the base relations with uniform distribution and out of the remaining 7 relations each one with probability $(l-1)/7$.[3]If this results in an allowed relation (i.e., RCC-8 or RCC-8 $-\widehat{\mathcal{H}}_8$), we assign this relation to the edge. Otherwise we repeat the process.

The reason for also generating instances using only relations not contained in $\widehat{\mathcal{H}}_8$ is that we assume that these instances are difficult to solve since every relation has to be split during the backtracking search, even if we use $\widehat{\mathcal{H}}_8$ as the split-set. We generated only instances of average label size $l = 4.0$.

The experiments were performed on a Sun Ultra 1 with 64MB of main memory.

4 Evaluation of the Path-Consistency Algorithm

Since the efficiency of the backtracking algorithm depends on the efficiency of the underlying path-consistency algorithm, we will first compare three different implementations of the path-consistency algorithm. In order to do so, we randomly generated instances with 50 to 1,000 regions. For each value of the average degree ranging from 8.0 stepping with 0.5 to 11.0 we generated 10 different instances. Figure 2 displays the average CPU time of the different methods for applying PCA to the generated instances. It can be seen that the effect of using a weighted queue is much higher for our problem than for the temporal problem (10× faster than using an ordinary queue without weights compared to only 2× faster [18]). Determining the weights of every relation using their exact restrictiveness does not have much advantage over approximating their restrictiveness using the approach by van Beek

[3] This method could result in the assignment of a universal constraint to a selected link, thereby changing the degree of the node. However, since the probability of getting the universal relation is very low, we ignore this in the following.

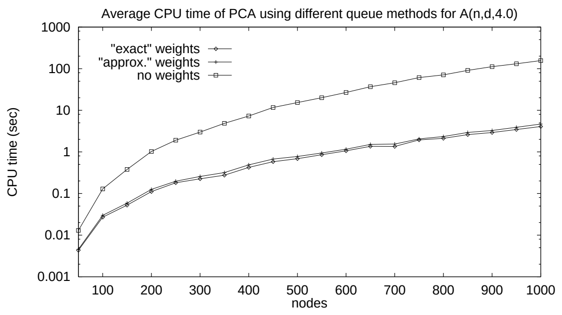

Figure 2. Comparing the performance of the path-consistency algorithm using different methods for weighting the queue (70 instances/data point, $d = 8.0 - 11.0$)

and Manchak [18], however. For our further experiments we always used the "exact weights" method because determining the restrictiveness amounts to just one table lookup.

5 The Phase-Transition Region

Similar to the empirical analysis of qualitative temporal reasoning problems [13], we wanted to determine the *phase transition* region [3] depending on the *average degree* of the nodes in the "constraint graph" in order to get an idea where the hard instances are. If all relations are allowed, the phase transition is around $d = 8 - 10$, depending on the instance size (see Figure 3).

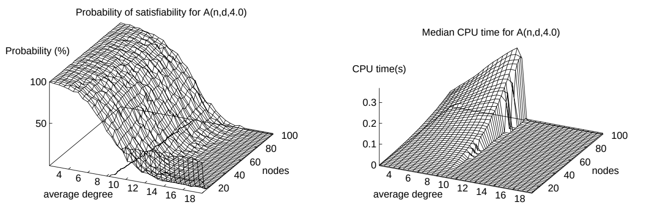

Figure 3. Probability of satisfiability and median CPU time for $A(n, d, 4.0)$ (500 instances/data point)

When using only "hard" relations, i.e., relations in RCC-8$-\widehat{\mathcal{H}}_8$, the phase transition appears at higher values for d (see Figure 4), but the median runtime appears to be similar to the former case.

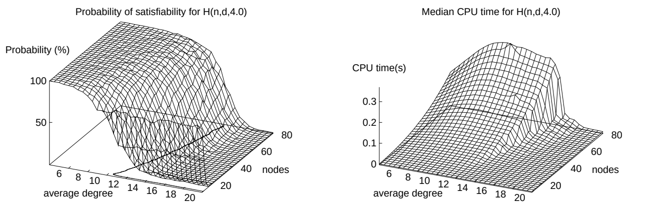

Figure 4. Probability of satisfiability and median CPU time for $H(n, d, 4.0)$ (500 instances/data point)

Nevertheless, these two sets of instances have quite different properties. This becomes obvious when counting the number of instances that are inconsistent, but path-consistent (see Figure 5). First of all, one notes that all such instances are close to the phase transition region. Secondly, there are many more inconsistent but path-consistent instances in the $H(n,d,4.0)$ case.

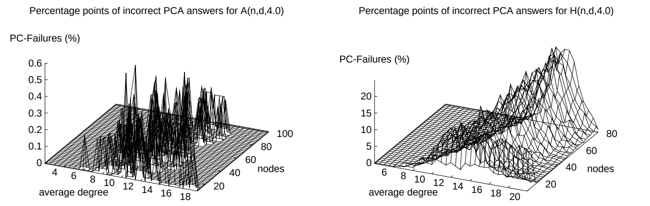

Figure 5. Percentage points of incorrect answers of the path-consistency algorithm for $A(n,d,4.0)$ and $H(n,d,4.0)$

Nevertheless, the path-consistency algorithm is still a very good approximation to consistency. As can be seen in Figure 5 this is particularly true for instances not in the phase transition region or when all relations are used.

6 Empirical Evaluation of the Heuristics

We ran all eight different heuristics (static/dynamic, local/global, hornsplit($\widehat{\mathcal{H}}_8$)/closebasesplit($\widehat{\mathcal{B}}$)) on the same randomly generated instances. For the hard relations we restricted ourselves to instances with 80 regions because larger ones appeared to be too difficult.

In first experiments we found that most of the instances were solved very fast with less than 1,000 visited nodes in the search space when using $\widehat{\mathcal{H}}_8$ for splitting. However, some problems turned out to be extremely hard, they could not be solved within our limit of 2 million visited nodes, which is about 1.5 hours of CPU time. Therefore we ran all our programs to a maximal number of 10,000 visited nodes and stored all instances that used more than 10,000 visited nodes for further experiments (see next section). The distribution of those hard instances is shown in Figure 6. It can be seen that almost all of them are in the phase transition region.

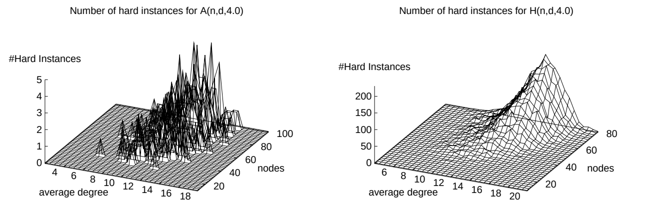

Figure 6. Number of instances using more than 10,000 visited nodes for any heuristic for $A(n,d,4.0)$ and $H(n,d,4.0)$

In Figure 7 we compare the 50% and 99% percentiles of the *dynamic/local* heuristic for the two split-sets on $A(n,d,4.0)$, where each data point is the average of the values for $d = 8$ to 10. We took the average of the different degrees in order to cover the whole phase transition which is about 8 for instances of size $n = 10$ and 10 for instances of size $n = 100$.

The CPU times of the other heuristics *static/local* and *static/global* were almost the same. Only the *dynamic/global* heuristic was about 3 times slower when using $\widehat{\mathcal{B}}$ and about 1.5 times slower when using $\widehat{\mathcal{H}}_8$. It can be seen that using $\widehat{\mathcal{H}}_8$ is double as fast as using $\widehat{\mathcal{B}}$ as the split-set. Even the percentile 99% CPU time of using $\widehat{\mathcal{H}}_8$ is faster than the median CPU time of using $\widehat{\mathcal{B}}$ in many cases.

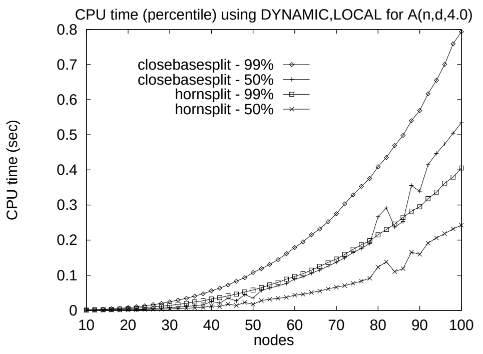

Figure 7. Percentile 50% and 99% CPU time of $\widehat{\mathcal{H}}_8$ and $\widehat{\mathcal{B}}$ for solving $A(n,d,4.0)$ ($d = 8.0 - 10.0$, 2,500 instances/data point)

Repeating the same experiments for $H(n,d,4.0)$, we noticed that there are very many hard instances for $n > 45$ (see Figure 6). For this reason, we show the results only up to a size of $n = 44$. Similarly to the $A(n,d,4.0)$ instances, one sees that using $\widehat{\mathcal{H}}_8$ is much faster than using $\widehat{\mathcal{B}}$ (Figure 8). Also *dynamic/global* was much slower than the other heuristics which have about the same CPU times.

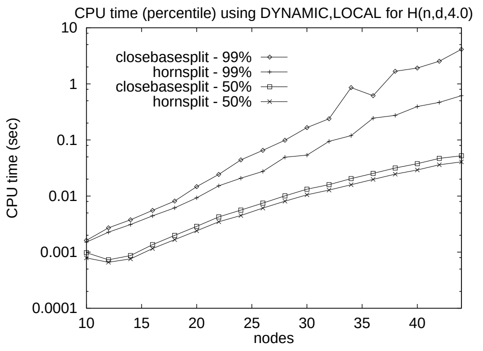

Figure 8. Percentile 50% and 99% CPU time of $\widehat{\mathcal{H}}_8$ and $\widehat{\mathcal{B}}$ for solving $H(n,d,4.0)$ ($d = 10.0 - 15.0$, 5,500 instances/data point)

It can be seen that although the median CPU times are about the same as for $A(n,d,4.0)$ the percentile 99% CPU times are much slower. As it was already shown in Figure 6 and Figure 5 this is a further evidence that there are very hard instances in the phase transition region of $H(n,d,4.0)$.

7 Evaluation of Orthogonally Combined Strategies for the Hard Instances

Nebel [13] observed that running different heuristics in parallel can solve more instances of a particular hard set of temporal reasoning instances proposed by van Beek and Manchak [18] in reasonable time than any single heuristic alone can solve.

We tried to evaluate the power of "orthogonally combining" our heuristics by running all of them on the set of hard instances (using more than 10,000 visited nodes) identified in the experiments described in the previous section. This led to a very surprising result for the $A(n,d,4.0)$ instances, namely, that all of the 384 hard instances except for a single one were solved by at least one of the heuristics using less than 10,000 visited nodes. When comparing the minimal response time of all the heuristics for all the hard instances, we found that only six of them used more than 300 visited nodes. This is in particular remarkable as all these instances are from the phase-transition region of an NP-hard problem, i.e., instances which are usually considered to be the most difficult ones. Further note that about 20% of the 384 instances were inconsistent. Interestingly, most of those inconsistent instances were solved faster than the consistent instances. At this point, it should be noted that combining heuristics orthogonally is very similar to randomized search techniques with restarts [17]. However, in contrast to randomized search, our method can also determine whether an instance is inconsistent.

We tried to solve the most difficult instance ($n = 64, d = 10.5$) with all of the eight heuristics spending 20,000,000 of nodes for each of it but did not succeed in solving it. In Table 1 we listed the percentage of hard instances that could be solved by the different heuristics, and the percentage of first response by each of them when running the heuristics in parallel.

	$A(n, d, 4.0)$		$H(n, d, 4.0)$	
Heuristics	Solved	1. Response	Solved	1. Response
$\widehat{\mathcal{H}}_8$/sta/loc	79.4%	17.6%	80.5%	13.9%
$\widehat{\mathcal{H}}_8$/sta/glo	90.0%	25.3%	86.4%	20.6%
$\widehat{\mathcal{H}}_8$/dyn/loc	81.8%	28.7%	90.7%	32.4%
$\widehat{\mathcal{H}}_8$/dyn/glo	78.4%	21.9%	72.1%	22.1%
$\widehat{\mathcal{B}}$/sta/loc	80.0%	1.6%	74.5%	2.8%
$\widehat{\mathcal{B}}$/sta/glo	84.4%	2.1%	70.8%	2.7%
$\widehat{\mathcal{B}}$/dyn/loc	81.3%	1.8%	60.6%	3.4%
$\widehat{\mathcal{B}}$/dyn/glo	64.8%	0.8%	–	–
combined	99.8%		99.3%	

Table 1. Percentage of solved hard instances for each heuristic

We did the same examination for the set of hard instances of the hard relations. However, we did not include the hard instances for the *dynamic/global/*$\widehat{\mathcal{B}}$ heuristic as it was very slow and generated so many hard instances which were easily solved by other heuristics that including them does not seem to be informative. The minimal response time for 92% of the 18,427 hard instances of $H(n, d, 4.0)$ was less than 300 visited nodes. 18.5% of the hard instances were inconsistent. Also the minimal response time for most of the inconsistent instances was faster than for the consistent instances.

It can be seen in Table 1 that although the number of solved instances was about the same for the $\widehat{\mathcal{H}}_8$ and the $\widehat{\mathcal{B}}$ heuristics, using $\widehat{\mathcal{H}}_8$ as the *split-set* is much more efficient in giving the fastest response. Nevertheless, the $\widehat{\mathcal{B}}$-heuristics were sometimes the fastest.

8 Discussion & Summary

The main motivation of our paper was to explore the space of methods for solving qualitative spatial reasoning instances and to get an idea of the critical instance size when reasoning becomes infeasible. It seems to be the case that up to 40 regions spatial reasoning in the RCC-8 framework can be done without any performance problems and that hard instances up to a size of 80 regions can almost always been solved in a few seconds, provided the orthogonal combination of heuristics as proposed by Nebel [13] is used.

One surprising result of our experiments was that the orthogonal combination of heuristics works quite well on the hard instances in the phase transition region. Additionally, we were surprised by the fact that this method is also quite useful in detecting inconsistent instances. This fact distinguishes our approach from randomized search methods with restarts (such as GSAT), which are only aimed at finding a "satisfying assignment" and which cannot prove that something is not satisfiable.

Finally, since we could show that almost any instance up to a certain size can be solved in a few seconds, RCC-8 seems to be also very suitable for efficient preprocessing even if RCC-8 is not expressive enough for a particular application and is only a sub-formalism of a more expressive formalism.

REFERENCES

[1] James F. Allen, 'Maintaining knowledge about temporal intervals', *CACM*, **26**(11), 832–843, (1983).

[2] Brandon Bennett, 'Spatial reasoning with propositional logic', in *Proc. KR-94*, pp. 51–62, (1994).

[3] Peter Cheeseman, Bob Kanefsky, and William M. Taylor, 'Where the *really* hard problems are', in *Proc. IJCAI-91*, pp. 331–337, (1991).

[4] Martin C. Golumbic and Ron Shamir, 'Complexity and algorithms for reasoning about time: A graph-theoretic approach', *JACM*, **40**(5), 1128–1133, (1993).

[5] Michelangelo Grigni, Dimitris Papadias, and Christos Papadimitriou, 'Topological inference', in *Proc. IJCAI-95*, pp. 901–906, (1995).

[6] R. M. Haralick and G. L. Elliot, 'Increasing tree search efficiency for constraint satisfaction problems', *Artificial Intelligence*, **14**, 263–313, (1980).

[7] Peter B. Ladkin and Alexander Reinefeld, 'Effective solution of qualitative interval constraint problems', *Artificial Intelligence*, **57**(1), 105–124, (1992).

[8] Peter Ladkin and Alexander Reinefeld. 'Fast algebraic methods for interval constraint problems', *Annals of Mathematics and Artificial Intelligence*, **19**, 383–411, (1997).

[9] Alan K. Mackworth, 'Consistency in networks of relations', *Artificial Intelligence*, **8**, 99–118, (1977).

[10] Alan K. Mackworth and Eugene C. Freuder, 'The complexity of some polynomial network consistency algorithms for constraint satisfaction problems', *Artificial Intelligence*, **25**, 65–73, (1985).

[11] Ugo Montanari, 'Networks of constraints: fundamental properties and applications to picture processing', *Information Science*, **7**, 95–132, (1974).

[12] Bernhard Nebel, 'Computational properties of qualitative spatial reasoning: First results', in *KI-95*, LNCS , pp. 233–244, (1995).

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

[14] David A. Randell, Anthony G. Cohn, and Zhan Cui, 'Computing transitivity tables: A challenge for automated theorem provers', in *Proc. CADE-92*, (1992).

[15] David A. Randell, Zhan Cui, and Anthony G. Cohn, 'A spatial logic based on regions and connection', in *Proc. KR-92*, pp. 165–176, (1992).

[16] Jochen Renz and Bernhard Nebel, 'On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the Region Connection Calculus', in *Proc. IJCAI-97*, (1997).

[17] Bart Selman, Hector J. Levesque, and David Mitchell, 'A new method for solving hard satisfiability problems', in *Proc. AAAI-92*, pp. 440–446, (1992).

[18] Peter van Beek and Dennis W. Manchak, 'The design and experimental analysis of algorithms for temporal reasoning', *JAIR*, **4**, 1–18, (1996).