A Divide-and-Conquer Approach for Solving Interval Algebra Networks

Jason Jingshi Li, Jinbo Huang, and **Jochen Renz**
Australian National University and National ICT Australia
Canberra, ACT 0200 Australia
{jason.li | jochen.renz}@anu.edu.au, jinbo.huang@nicta.com.au

Abstract

Deciding consistency of constraint networks is a fundamental problem in qualitative spatial and temporal reasoning. In this paper we introduce a divide-and-conquer method that recursively partitions a given problem into smaller sub-problems in deciding consistency. We identify a key theoretical property of a qualitative calculus that ensures the soundness and completeness of this method, and show that it is satisfied by the Interval Algebra (IA) and the Point Algebra (PA). We develop a new encoding scheme for IA networks based on a combination of our divide-and-conquer method with an existing encoding of IA networks into SAT. We empirically show that our new encoding scheme scales to much larger problems and exhibits a consistent and significant improvement in efficiency over state-of-the-art solvers on the most difficult instances.

1 Introduction

Temporal information such as "the financial crisis begun during the 2008 presidential campaign" does not specify exact dates or durations; only qualitative information about the relationships between different periods of time is given. Allen [1983] introduced the Interval Algebra (IA), a qualitative calculus to represent and reason about such relationships between temporal intervals. It distinguishes 13 *atomic* relations, i.e., between any two temporal intervals exactly one of these relations holds. Allen's approach was very successful and marked the start of a new research area, now called qualitative spatio-temporal reasoning (QSTR).

QSTR uses constraint calculi to reason about spatial and temporal information. The fundamental reasoning task in QSTR is to decide whether a given collection of information is consistent. This can be formulated as a *constraint satisfaction problem*. Fig. 1a depicts a simple example that is represented as a constraint network over IA: "Peter reads the newspaper during breakfast, and goes to work after breakfast." This information is clearly consistent; Fig. 1b depicts a solution where each variable has been instantiated to a concrete time interval. However, for large and complex networks, which may contain disjunctions of atomic relations such as

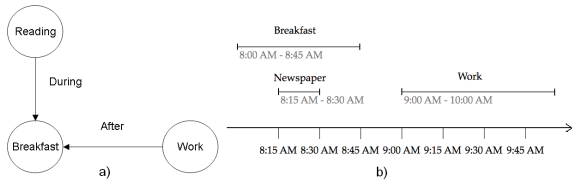

Figure 1: a) an IA network, and b) an instantiation of the network in the temporal domain.

"Peter reads his private email either before or after work," we will need temporal reasoning algorithms to decide consistency.

Deciding consistency in QSTR is NP-hard for many calculi including IA. If only atomic IA relations are permitted, however, the consistency problem becomes tractable. Specifically, it has been shown that the *path consistency* algorithm, which removes impossible relations from all triples of nodes in a constraint network, is sufficient for deciding consistency for atomic IA networks, i.e., networks where each edge is an atomic relation.

In the past twenty years there have been many attempts to make reasoning in IA and other spatial or temporal calculi more efficient. Most of these approaches use constraint satisfaction techniques where an instance is solved by backtracking over tractable sub-instances. More efficient solutions were obtained by studying different backtracking heuristics and most notably also by enlarging known *tractable subsets*—sets of relations for which the consistency problem is tractable [Nebel and Bürckert, 1995]. By combining different backtracking heuristics and the maximal tractable subset of IA, many instances, even hard ones in the phase-transition region, were solved reasonably fast [Nebel, 1997]. However, particularly large instances remain difficult to solve.

Pham *et al.* [2008] recently proposed an encoding of IA networks into Boolean formulas such that the formula is satisfiable if and only if a path-consistent atomic refinement of the network exists. The encoding enforces path consistency using a set of clauses for every triple of nodes in the network, and solution finding is delegated to an off-the-shelf SAT solver. This results in increased efficiency for some hard problem instances, but its memory requirements can be quite demanding as the encoding grows cubically with network size. Our ex-

periments showed that for large instances the SAT formulas can easily become too large even for modern SAT solvers.

In this paper we present a theoretical analysis that results in a divide-and-conquer approach to solving IA networks. The basic idea is that under certain conditions, which IA satisfies, one can decompose a network into two overlapping sub-networks, and solve the two sub-networks separately. This eliminates the need for examining triangles across the two sub-networks when enforcing path consistency. The same idea applies recursively to each of the sub-networks, leading to further savings. Combining this result with the approach of [Pham *et al.*, 2008] we are then able to produce significantly more compact SAT encodings and more efficient solutions.

In the following section we review the basic notions of qualitative spatial and temporal calculi and summarize existing constraint-based and SAT-based methods. In Section 3 we present a theoretical analysis of IA that leads to a divide-and-conquer SAT encoding scheme for IA networks. In Section 4 we empirically evaluate the resulting SAT encoding and show that it is superior to existing solvers on hard instances. Section 5 concludes the paper.

2 Preliminaries

An IA network describes possible scenarios of intervals along a directed line. The network is a complete graph with vertices representing the intervals, and directed edges representing the possible relationship between them. As in our earlier example of Fig. 1a, we often omit edges where the relationship between the two intervals is the universal relation (i.e., no constraint is specified). Also note that we only need to give one directed edge between a pair of vertices as the other one is implied.

More formally, a qualitative temporal (or spatial) calculus $\mathbf{A}$ represents relationships between temporal (or spatial) entities from a given *domain* $\mathcal{D}$, such as intervals, points, or extended regions. A calculus partitions $\mathcal{D} \times \mathcal{D}$ into a finite set of pairwise disjoint and jointly exhaustive *atomic relations* $\mathcal{B}$, i.e., between any two elements of $\mathcal{D}$ exactly one atomic relation holds. Temporal or spatial information can then be represented in the form of constraints xRy, where x and y are variables over $\mathcal{D}$ and $R \in 2^{\mathcal{B}}$ is a disjunction of atomic relations. Disjunctions are used to represent indefinite information. If the exact relationship between x and y is known, then R is an atomic relation.

IA [Allen, 1983] is the best known example of a qualitative calculus. Its domain is the set of intervals on a directed time line, the 13 atomic relations are {before, after, meets, met-by, overlaps, overlapped-by, equals, during, includes, starts, started-by, finishes, and finished-by}, leading to $2^{13} = 8192$ possible IA relations.

A set of binary constraints Θ of a qualitative calculus $\mathbf{A}$ can be represented as a binary constraint network $N = (V_N, \ell_N)$ over $\mathbf{A}$, where the variables of Θ become the vertices V_N of N and $\ell_N(i,j) = R_{ij}$ labels the edge between nodes i and j, where R_{ij} denotes the constraint between variables i and j. Note that R_{ij} is the universal relation in case no constraint is imposed between variables i and j.

A network N is a *sub-network* of another network M if $V_N \subseteq V_M$ and the labeling ℓ_N of the edges of N is exactly the same as the labeling ℓ_M of the same edges in M, i.e., $\ell_N(i,j) = \ell_M(i,j) \forall i,j \in V_N$. We write $N \subseteq M$ in such cases. N is a *refinement* of M if $V_N = V_M$ and $\ell_N(i,j) \subseteq \ell_M(i,j) \forall i,j \in V_N$. The intersection ($\cap$) of two networks N_1, N_2 is the maximal network that is a sub-network of both N_1 and N_2. When the two networks satisfy $V_{N_1} \cap V_{N_2} = V_{N_1 \cap N_2}$, we also define their union ($\cup$) to be the minimal[1] network of which both N_1 and N_2 are sub-networks. A network N is *atomic* if all the labels of N are atomic relations of $\mathbf{A}$, and is *path consistent* if $R_{ij} \subseteq R_{ik} \circ R_{kj} \forall i,j,k \in V_N$, where $\circ$ is the composition operator of $\mathbf{A}$ (the composition $R \circ S$ of two relations is defined as the relation $\{(a,c) | \exists b : (a,b) \in R, (b,c) \in S\}$).

2.1 Deciding consistency

The fundamental reasoning problem for a qualitative calculus is that of deciding *consistency* for a set of constraints Θ, i.e., deciding whether there is an instantiation of all variables in Θ with values from the domain $\mathcal{D}$ such that all constraints in Θ are satisfied. As domains of a qualitative calculus such as IA are usually infinite, standard constraint propagation methods do not directly apply. Instead, we can use operators on the relations in order to derive new information from the given constraints.

An example of such a technique is the well-known path consistency algorithm, which iteratively checks all triangles in the network to rule out impossible relations until a fixed point is reached. Continuing with our example, given that "reading" occurs during "breakfast" and "work" occurs after "breakfast," path consistency will detect that it must be the case that "reading" occurs before "work," after eliminating the other 12 atomic relations that are initially in the label. As no other relations on any of the edges can be eliminated, the network is now path consistent.

IA has the property that for any atomic network of size n, path consistency implies strong n-consistency [van Beek and Cohen, 1990]. This means that given a path-consistent atomic network and any subset of its variables (nodes) of size k, any consistent instantiation of $k-1$ variables in the set can be extended to a consistent instantiation of all k variables. This reduces consistency checking to a search for a path-consistent atomic network that is a refinement of the original network.

2.2 Constraint-based approach

The basic constraint-based approach for IA networks is hence a backtracking search where branching is performed on the atomic relations in the label of each edge. Path consistency is enforced immediately after each branch is created, and any inconsistency thus detected results in backtracking. The network is consistent if and only if a leaf of the search tree can be successfully reached where every edge has been assigned an atomic relation. Variable and value ordering heuristics for these algorithms have been studied to improve their efficiency [Ladkin and Reinefeld, 1992].

[1] Minimality here means that the network has the fewest vertices *and* the fewest edges possible, where we regard edges labeled with universal relations as invisible.

The major improvement in the efficiency of constraint-based solvers has been the use of large and preferably maximal tractable subsets of the full set of relations. Consistency for networks that contain only relations from such sets can be decided in polynomial time, often with the path consistency algorithm. The only maximal tractable subset (ORD-Horn) of IA that contains all atomic relations has been identified by Nebel and Bürckert [1995]. Nebel [1997] showed that the right combination of tractable subset and heuristics can lead to very fast solutions. This result led to an increased interest in identifying tractable subsets for other qualitative calculi [Renz and Li, 2008].

While Nebel's solver has been the standard IA solver for the past ten years, recent improvements have been proposed [Condotta *et al.*, 2007] and implemented in the latest version of the GQR solver [Gantner *et al.*, 2008], which is currently the fastest constraint-based IA solver.

2.3 SAT-based approach

Recently, Pham *et al.* [2008] proposed an encoding of IA networks into Boolean formulas. The encoding can be based on either the constraints between the intervals as given in the network (event-based), or the constraints between the end-points of those intervals (point-based). In either case, the Boolean formulas are constructed in such a way that each solution of the formula corresponds to a path-consistent atomic refinement of the network, and vice versa. Hence the formula is satisfiable if and only if the network is consistent.

They studied several alternative encoding methods and found that empirically the "point-based 1D support" encoding performs best. The "1-D support" encoding works by allocating a Boolean variable x_{ij}^v for every atomic relation v on the edge between two intervals (i, j). The variable is true if and only if the corresponding atomic relations holds between i and j in the final solution of a path-consistent atomic network. Two sets of at-least-one (ALO) and at-most-one (AMO) clauses are introduced to ensure that exactly one of these atomic relations holds in the final solution:

- ALO: $\bigvee_{v \in R_{ij}} x_{ij}^v$; AMO: $\bigwedge_{u,v \in R_{ij}} \neg x_{ij}^u \vee \neg x_{ij}^v$

To ensure the final network is path consistent, a set of support (SUP) clauses is introduced to encode that every triangle (i, j, k) of the network (regarded as a complete graph) be closed under composition and intersection:

- SUP: $\bigwedge_{u \in R_{ik}, v \in R_{kj}} \neg x_{ik}^u \vee \neg x_{kj}^v \vee x_{ij}^{w_1} \vee \ldots \vee x_{ij}^{w_m}$

where $\{w_1, \ldots, w_m\} = R_{ij} \cap (u \circ v)$.

In the point-based encoding, one includes the same set of clauses as above, but for the point algebra (PA) instead of IA, prescribing the relations between the endpoints of the intervals. The variable $x_{i^- j^-}^u$, for example, denotes the atomic relation u between the starting points of intervals i and j (substitute $+$ for $-$ for the ending points). To ensure soundness, this encoding requires an additional set of clauses to forbid spurious IA relations introduced by the intervals-to-points translation:

- $\bigwedge_{r \notin R_{ij}} \neg x_{i^- j^-}^u \vee \neg x_{i^- j^+}^u \vee \neg x_{i^+ j^-}^u \vee \neg x_{i^+ j^+}^u$

In their empirical study of IA networks of up to 100 nodes, Pham *et al.* [2008] have shown that the "point-based 1-D support" encoding used in conjunction with the MiniSat SAT solver outperforms Nebel's solver with its default settings on many hard instances.

3 A new divide-and-conquer approach

As mentioned earlier, a weakness of the SAT-based approach is the size of the encoding, which for large networks can outgrow available memory or can make SAT solving otherwise inefficient. In particular, many networks are not densely connected (the missing edges implicitly represent universal relations) and the encoding must nevertheless treat the network as a complete graph and generally include clauses for all its triangles, leading to a cubic complexity in all cases.

Our solution to this issue starts with a new theoretical result about how two consistent networks sharing a common sub-network can be *amalgamated* into a larger consistent network, provided a particular property holds. We then present a method that uses this result and recursively partitions a network into small sub-networks before they are individually encoded into Boolean formulas, thus avoiding the need to include many triangles across the sub-networks that have edges labeled by universal relations.

3.1 Amalgamation of networks

Previously, Li *et al.* [2008] analysed the problem of combining networks in QSTR without introducing inconsistencies. They introduced the Network Amalgamation Property (NAP), which guarantees that two networks over a calculus can be amalgamated into a larger network such that applying path consistency to the resulting network does not change any existing constraints. Here we consider a form of NAP where we restrict the two networks and their amalgams to be atomic networks. We call this the Atomic Network Amalgamation Property (aNAP).

Definition 1 (Atomic Network Amalgamation Property) *A qualitative calculus* $\mathbf{A}$ *has aNAP if for any path-consistent atomic networks* N_1 *and* N_2 *over* $\mathbf{A}$ *such that* $V_{N_1} \cap V_{N_2} = V_{N_1 \cap N_2}$, $N_1 \cup N_2$ *has a path-consistent atomic refinement.*

Fig. 2 illustrates this property. Consider the networks N_1 and N_2 over a calculus $\mathbf{A}$ having aNAP shown in Fig. 2a, with their intersection shown on the far left. Their union $N_1 \cup N_2$ is shown in Fig. 2b, where the dotted edge represents the universal relation and all other edges are those from N_1 and N_2. The dotted edge is the only edge with a non-atomic label and aNAP guarantees the existence of an atomic refinement of the edge that results in a path-consistent atomic network.

Next we establish the following sufficient condition for aNAP, and show that it is satisfied by both IA and PA.

Theorem 1 *A qualitative calculus has aNAP if path consistency implies strong* n*-consistency for all atomic networks of size* n *over the calculus.*

Proof. Given a qualitative calculus $\mathbf{A}$ satisfying the above condition and two path-consistent atomic networks N_1 and N_2 over $\mathbf{A}$ such that $V_{N_1} \cap V_{N_2} = V_{N_1 \cap N_2}$, let $N_0 = N_1 \cap N_2$

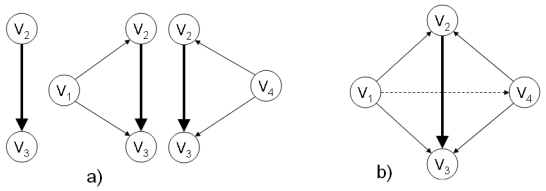

Figure 2: Amalgamation of atomic networks.

and $M = N_1 \cup N_2$. We wish to show that M is consistent, which will establish aNAP. Take any consistent instantiation α_1 of N_1, and consider its projection α_0 on V_{N_0}. Since N_2 is strongly $|V_{N_2}|$-consistent, α_0 can be extended to a complete consistent instantiation α_2 of N_2. The extension will not conflict with α_1 because none of the additional variables are in V_{N_1} (since $V_{N_1} \cap V_{N_2} = V_{N_0}$). Hence the combination of α_1 and α_2 is a consistent instantiation of M. ■

Corollary 1 *IA and PA have the aNAP property.*

Proof. Path consistent atomic networks of size n in IA and PA are guaranteed to be strongly n-consistent [van Beek and Cohen, 1990]. Hence both have the aNAP property. ■

3.2 Partitioning networks: reverse amalgamation

The above results allow us to amalgamate two networks into a larger network while preserving consistency. In this work we are interested in a divide-and-conquer method for networks, i.e., exactly the opposite of amalgamation: we want to take a large network and partition it into smaller networks such that their consistency can be used to decide the consistency of the larger network. The following theorem, directly derivable from the definition of aNAP, formalizes this idea.

Theorem 2 *Given a qualitative calculus* **A** *for which aNAP holds and for which path consistency implies consistency of atomic networks, a network M over* **A** *is consistent if there exist networks M_1, M_2 such that $M_1 \cup M_2 = M$, M_1 has a path-consistent atomic refinement N_1, M_2 has a path-consistent atomic refinement N_2, and $N_1 \cap N_2$ is a refinement of $M_1 \cap M_2$.*

Given that aNAP holds for a qualitative calculus, this theorem allows us to partition a network M into two overlapping sub-networks N_1 and N_2 that satisfy $V_{N_1} \cap V_{N_2} = V_{N_1 \cap N_2}$ and $N_1 \cup N_2 = M$. When deciding path consistency, we can then disregard all triangles with at least one edge (labeled with the universal relation) between $V_{N_1} \setminus V_{N_1 \cap N_2}$ and $V_{N_2} \setminus V_{N_1 \cap N_2}$. This applies recursively to each sub-network while we ensure that the intersection of the two sub-networks from any previous level stays together (to retain soundness).

We can stop the partition at any stage (or continue until no networks can be partitioned anymore) and then encode the remaining triangles using the existing SAT encoding of [Pham *et al.*, 2008]. Theorem 2 guarantees the soundness of the new encoding; completeness is also retained as the new encoding only disregards triangles in the network.

3.3 Partitioning networks: an example

In the following we give an example that demonstrates our partitioning method and how it leads to a smaller SAT encoding. Consider the network M of eight nodes $\{v_1 \dots v_8\}$ over either IA or PA in Fig. 3a, where edges labeled with universal relations are omitted. Recall that the basic SAT encoding must view the network as a complete graph and hence consider triangles over all edges including the invisible ones. Our goal is to show how some of those triangles can be ignored.

Let us make a cut through M as shown by the dotted line, breaking the network into the sub-network N_2 (Fig. 3c) on the right, and N_1 (Fig. 3b) on the left. Note that we have added to N_1 all the endpoints of the cut edges from the other side together with the incident edges; this is to ensure that $N_1 \cup N_2 = M$.

Theorem 2 implies that we need only encode N_1 and N_2 separately as long as we ensure that their respective solutions agree over their intersection. This condition is automatically ensured in the SAT encoding as long as we encode the intersection of N_1 and N_2 only once (while encoding the remainders of N_1 and N_2 separately). This immediately allows us to ignore a total of 21 triangles, namely those that include at least one (invisible) edge from $\{v_1, v_2, v_3\}$ to $\{v_7, v_8\}$.

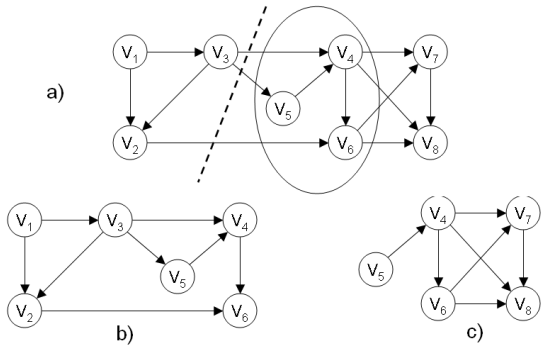

Figure 3: Partitioning a network.

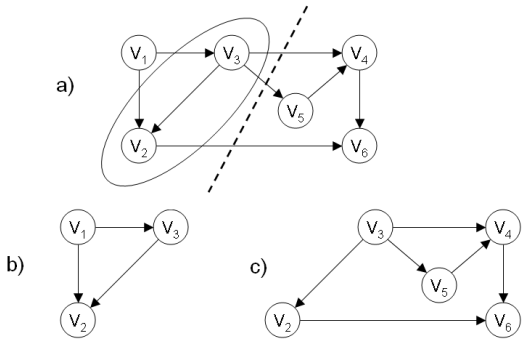

Figure 4: Recursively partitioning a sub-network.

We can apply the same idea recursively to each of the sub-networks. Let us proceed with N_1 for example. In recursion we need to take one additional measure, namely to ensure that the intersection of the two sub-networks previously created stays together. In this example, that is to say that we cannot split v_4, v_5, and v_6 when partitioning N_1. Fig. 4.b gives a feasible partitioning in this regard, allowing us to further ignore 3 triangles: (v_1, v_4, v_5), (v_1, v_4, v_6), and (v_1, v_5, v_6).

In the end, we avoid encoding 24 out of the 56 triangles compared with the original approach of [Pham *et al.*, 2008].

3.4 Partitioning heuristics

In general there are multiple ways to partition a given network. Our goal is to find "good" partitions, for example with a small overlap and a good balance between the two subnetworks, both of which tend to maximize the number of triangles that can be disregarded. Our implementation uses the software tool hMETIS [Karypis and Kumar, 1998]. hMETIS heuristically minimizes the number of edges being cut while producing a relatively balanced cut, which serves well to help achieve our goal.[2]

4 Empirical evaluation

We benchmarked the performance of our partitioned SAT encoding with the original SAT encoding from [Pham *et al.*, 2008], as well as two constraint-based solvers, the one from Nebel [1997] and the GQR-994 solver [Gantner *et al.*, 2008], with their best-performing heuristics enabled. In particular, Nebel's solver was run with options {static, global, queue} enabled, which had been shown to vastly improve the performance over default settings of the program in [Nebel, 1997]. MiniSat v2.070721 [Eén and Sörensson, 2003] was used to solve the CNF formulas generated by the SAT encodings. All our test instances were randomly generated networks around the phase-transition region with a given number of nodes and average degree, and an average label size of 6.5. Furthermore, we tested only networks that were not found inconsistent by path consistency. The tests were ran on 2.4 GHz processors with a 2 hour time limit and a memory limit of 2 GB.

4.1 Smaller networks

In the first part of the experiment we tested 27,000 IA networks with from 50–100 nodes and degrees from 8–12. We tested the event-based and point-based encodings of both the original and partition-based approaches. Out of all the tests only Nebel's solver failed on 9 instances; every other solver managed to solve all the instances.

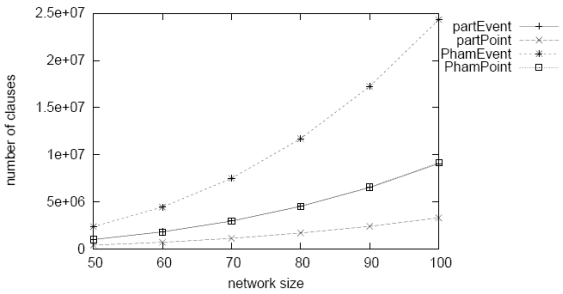

Figure 5: Average number of clauses for networks of size 50–100, degree 8–12; 4500 instance per datapoint.

The size of the Boolean formula generated in terms of number of clauses for each encoding is shown in Fig. 5. The data confirms that point-based encoding generates smaller CNF formulas than the event-based encoding for the same network. It also shows that the number of clauses generated from the our partitioned encoding is roughly half that from the encoding of [Pham *et al.*, 2008], and our event-based partitioned encoding produces about as many clauses as their point-based encoding.

[2]The parameters used in calling hMETIS are: $options[] = \{1, 10, 1, 1, 1, 0, 0, 1, 0\}$, and $ubFactor = 35$; in recursion we set $options[6] = 1$ and specify a set of nodes that needs to stay together.

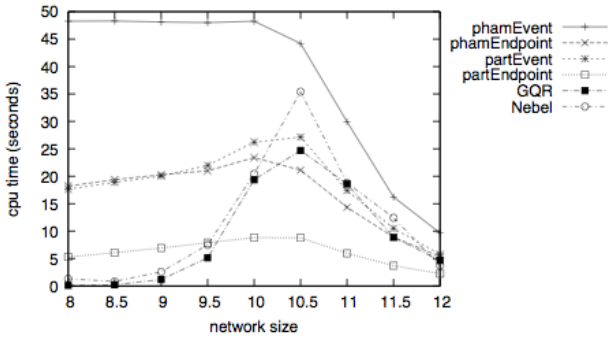

Figure 6: Average CPU time against network degree, for networks of size 50–100; 3000 instance per datapoint.

Fig. 6 compares the performance of all solvers. It confirms that the constraint-based approaches of Nebel and GQR solve easy instances very efficiently, whereas our point-based partitioned encoding dominates all other approaches in average CPU time on the harder instances.

4.2 Larger networks

In the second part of the experiment we generated 100 networks of size 110 to 200 with an average label size 6.5 for each degree from 8 to 12. In particular, we focused on the phase-transition region which was identified to be between degrees 10 and 11.5. We recorded the number of instances solved by each solver within the time and memory limit, and the time taken to solve them.

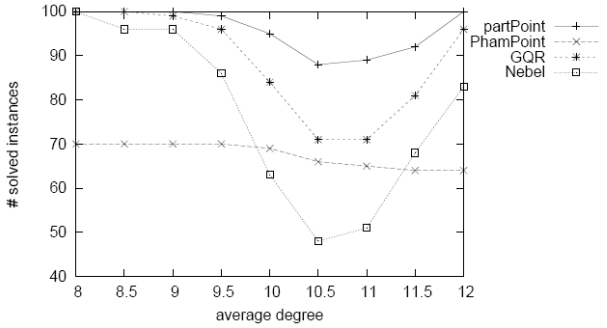

Figure 7: Number of solved instances against network degree.

Fig. 7 plots the number of instances solved against network degree, showing that the partitioned encoding dominates all other solvers for networks of any degree. The superior scalability of the partitioned encoding is further illustrated in Fig. 8, which focuses on the hard region and plots the number of instances solved against network size.

To further complete the picture, Fig. 9 shows the number of instances solved against CPU time. The constraint-

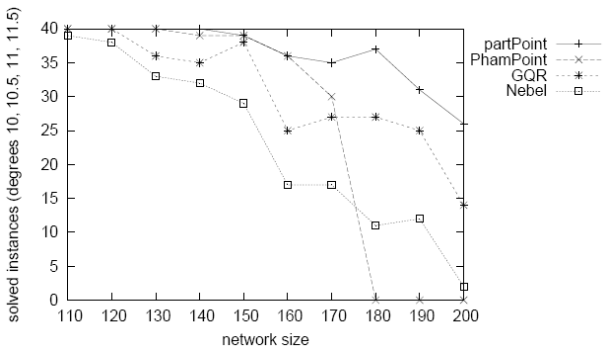

Figure 8: Number of solved instances against network size, for the phase-transition region.

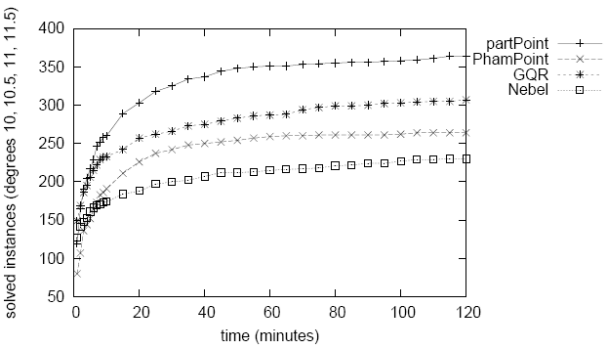

Figure 9: Number of solved instances against CPU time, for the phase-transition region.

based approaches of Nebel and GQR solved a large number of instances very quickly, but struggled to solve the harder instances even when more time was available. The point-based encoding of [Pham *et al.*, 2008] performed better than Nebel's solver, but was inferior to GQR. Once again our point-based partitioned encoding solved more instances than all the other approaches, and solved most instances under 1 hour.

5 Conclusion and future work

Based on the idea of divide-and-conquer, we proposed a new method for encoding constraint networks over IA into SAT. The method relies on a special property of IA that allows two atomic networks over IA to be consistently amalgamated over a shared sub-network. We created a program that uses this property to partition networks over IA to produce compact SAT encodings. The empirical evaluation shows that our partitioned encoding dominates existing approaches in efficiency and scalability on difficult IA instances.

In general, reduced formula size does not necessarily lead to faster SAT solving. While our results indicate that detecting and removing redundant clauses via partitioning is indeed beneficial, a second interesting observation can be made regarding the correlation of encoding size and solution time: As Pham *et al.* [2008] have previously observed and we again confirmed, the point-based encoding in fact leads to both smaller formulas and faster solutions than the event-based encoding whether or not partitioning is employed. This is apparently due to PA being a much smaller algebra—the smaller composition table leads to fewer clauses required per triangle for a given network. This indicates that SAT-based approaches may be made more efficient if a network can be translated into another one over a smaller calculus.

In this paper we demonstrated the usefulness of our method for solving IA networks. However, our method is applicable to any qualitative calculus for which path consistency implies consistency for atomic networks and for which aNAP holds. While the former property holds for many calculi, it is far less known where the latter holds. We presented a sufficient condition (Theorem 1) for aNAP, but it remains unclear whether it is also necessary. Another question is whether a purely syntactic and automated proof for aNAP exists, which would allow us to automatically identify calculi to which our partitioning method is applicable.

Acknowledgments

National ICT Australia is funded by the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

References

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

[van Beek and Cohen, 1990] P. van Beek and R. Cohen. Exact and approximate reasoning about temporal relations. *Comp. Intelligence*, 6:132–144, 1990.

[Condotta *et al.*, 2007] J.-F. Condotta, G.Ligozat, and M. Saade. Eligible and frozen constraints for solving temporal qualitative constraint networks. *CP'07*, 806–814, 2007.

[Eén and Sörensson, 2003] N. Eén and N. Sörensson. An extensible SAT-solver. *SAT'03*, 502–518, 2003.

[Gantner *et al.*, 2008] Z. Gantner, M. Westphal, and S. Wölfl. GQR—A fast reasoner for binary qualitative constraint calculi. *AAAI'08 WS on Spat. and Temp. Reasoning*, 2008.

[Karypis and Kumar, 1998] G. Karypis and V. Kumar. *hMeTiS: A Hypergraph Partitioning Package*, 1998. http://www.cs.umn.edu/~karypis.

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

[Li *et al.*, 2008] J.J. Li, T. Kowalski, J. Renz, and S. Li. Combining binary constraint networks in qualitative reasoning. *ECAI'08*, 515–519, 2008.

[Nebel and Bürckert, 1995] B. Nebel and H-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *J. ACM*, 42(1):43–66, 1995.

[Nebel, 1997] B. 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.*, 2008] D.N. Pham, J. Thornton, and A. Sattar. Modelling and solving temporal reasoning as propositional satisfiability. *Artif. Intell.*, 172(15):1752–1782, 2008.

[Renz and Li, 2008] J. Renz and J.J. Li. Automated complexity proofs for qualitative spatial and temporal calculi. *KR'08*, 715–723, 2008.

[SAT, 2008] SAT. The Annual SAT Competitions., 2008. http://www.satcompetition.org/.