Dynamic Branching in Qualitative Constraint-based Reasoning via Counting Local Models

Michael Sioutis[a,*], Diedrich Wolter[a]

[a] *Bamberg University, Faculty of Information Systems and Applied Computer Sciences, Bamberg, Germany*

Abstract

We introduce and evaluate *dynamic branching* strategies for solving Qualitative Constraint Networks (QCNs), which are networks that are mostly used to represent and reason about spatial and temporal information via the use of simple qualitative relations, e.g., a constraint can be "Task A is scheduled *after or during* Task C". In qualitative constraint-based reasoning, the state-of-the-art approach for tackling a given QCN consists in employing a backtracking algorithm, where the branching decisions during search are governed by the restrictiveness of the possible relations for a given constraint (e.g., *after* can be more restrictive than *during*). In the literature, that restrictiveness is defined a priori by means of static weights that are precomputed and associated with the relations of a given calculus, without any regard to the particulars of a given network instance of that calculus, such as its structure. In this paper, we address this limitation by proposing heuristics that dynamically associate a weight with a relation, based on the *count* of *local models* (or *local scenarios*) that the relation is involved with in a given QCN; these models are local in that they focus on triples of variables instead of the entire QCN. Therefore, our approach is adaptive and seeks to make branching decisions that preserve most of the solutions by determining what proportion of local solutions agree with that decision. Experimental results with a *random* and a *structured* dataset of QCNs of Interval Algebra show that it is possible to achieve up to 5 times better performance for structured instances, whilst maintaining non-negligible gains of around 20% for random ones. Finally, we show that these results can be further and notably improved via a selection protocol algorithm that aims to effectively and efficiently synthesize the involved heuristics (static or dynamic) into an overall better performing *meta-heuristic* in the phase transition.

Keywords: Qualitative constraints, spatial and temporal reasoning, counting local models, dynamic branching, adaptive algorithm

*Corresponding author

Email addresses: michail.sioutis@uni-bamberg.de (Michael Sioutis), diedrich.wolter@uni-bamberg.de (Diedrich Wolter)

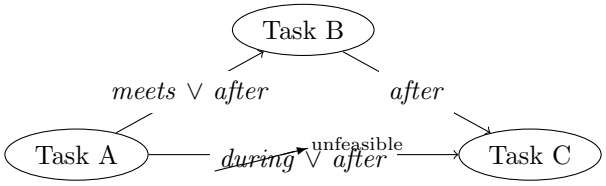


Figure 1: The static weighting scheme in the literature dictates that relation *during* is less restrictive than relation *after* in general for the IA calculus and, hence, *during* should be preferred over *after* in branching decisions [19, Figure 9], but in the above simplified QCN *during* cannot appear in any solution; such schemes are defined for other calculi as well [20]

1. Introduction

Qualitative Spatial and Temporal Reasoning (QSTR) is a major field of study in AI that deals with the fundamental cognitive concepts of space and time in a human-like manner, via simple qualitative constraint languages [1, 2]. Such languages consist of abstract, qualitative, expressions like *inside*, *before*, or *north of* to spatially or temporally relate two or more objects to one another, without involving any quantitative information. Thus, QSTR offers tools for efficiently automating common-sense spatio-temporal reasoning and, hence, further boosts research to a plethora of application areas and domains that deal with spatio-temporal information, such as cognitive robotics [3], deep learning [4], visual explanation [5] and sensemaking [6], semantic question-answering [7], qualitative simulation [8], modal logic [9, 10, 11, 12, 13], temporal diagnosis [14], and stream reasoning [15, 16].

Qualitative spatial or temporal information may be modeled as a *Qualitative Constraint Network* (QCN), which is a network where the vertices correspond to spatial or temporal entities, and the arcs are labeled with qualitative spatial or temporal relations respectively. For instance $x \leq y$ can be a temporal QCN over $\mathbb{Z}$. Given a QCN $\mathcal{N}$, the literature is particularly interested in its *satisfiability problem*, which is the problem of deciding if there exists a spatial or temporal interpretation of the variables of $\mathcal{N}$ that satisfies its constraints, viz, a *solution* of $\mathcal{N}$. For instance, $x = 0 \wedge y = 1$ is one of the infinitely many solutions of the aforementioned QCN, and $x < y$ is the corresponding *scenario* that concisely represents all the cases where x is assigned a lesser value than y. In general, for most widely-adopted qualitative calculi the satisfiability problem is NP-complete [17]. In the sequel, we will be using Interval Algebra (IA) [18] as an illustrative example of a qualitative calculus.

Motivation & Contribution. The state-of-the-art constraint-based approach for tackling a given QCN consists in employing a backtracking algorithm, where each branching decision during search is guided by the restrictiveness of the possible relations for a given constraint. Currently, that restrictiveness is defined a priori by means of entirely precomputed static weights that are associated with the relations of a given calculus. That static strategy has two major problems: it assumes a uniform use of relations in QCNs (in the sense that

weights are computed by equally considering all the relations of a calculus); and it does not exploit any structure that may exist in QCNs (a relation that is used to form more than one constraints in a given QCN, which is typically the case, may exhibit different levels of restrictiveness among those constraints). A simple example of how this scheme can be problematic is detailed in Figure 1. In this paper, we address this limitation by proposing a *dynamic branching* mechanism via heuristics that dynamically associate a weight with a relation during search, based on the *count* of *local models*, i.e., scenarios pertaining to triples of variables, that the relation is involved with in a given QCN. This makes our approach similar to a counting-based one for CSPs [21], as it too is adaptive and it too seeks to make branching decisions that preserve most of the solutions by determining what proportion of local solutions agree with that decision. Further inspiration was drawn from a recent work in [22], where it was observed that a scenario of a QCN may often be constructed collectively by relations that appear in many scenarions individually, i.e., a scenario of a QCN may often be constructed by selecting the most popular relation for each constraint. Through an evaluation with a random and a structured dataset of QCNs of IA, we show that we may achieve up to 5 times better performance for structured instances, and gains of about 20% for random ones. Finally, we also present a *selection protocol* algorithm that aims to effectively and efficiently synthesize the involved heuristics (static or dynamic) into a better performing *meta-heuristic*, and show that is possible to do better in the phase transition of each of our datasets with just a minimal amount of dynamic preprocessing. Specifically, we show that we can further ameliorate the already improved results by up to about 15% for structured instances and 5% for random ones.

The rest of the paper is organized as follows. In Section 2 we give some preliminaries on QSTR. Next, in Section 3 we propose our dynamic branching approach, discuss some dynamic heuristics that are used internally, present the related algorithms, and evaluate our framework with random and structured QCNs of IA, commenting on the outcome. Then, in Section 4 we present a selection protocol algorithm that aims to effectively and efficiently synthesize the involved heuristics into a better performing meta-heuristic, and evaluate it accordingly. Finally, in Section 5 we discuss related work, and in Section 6 we draw some conclusive remarks and give directions for future work.

2. Preliminaries

A binary qualitative spatial or temporal constraint language, is based on a finite set B of *jointly exhaustive and pairwise disjoint* relations, called the set of *base relations* [23], that is defined over an infinite domain D. The base relations of a particular qualitative constraint language can be used to represent the definite knowledge between any two of its entities with respect to the level of granularity provided by the domain D. The set B contains the identity relation Id, and is closed under the *converse* operation ($^{-1}$). Indefinite knowledge can be specified by a union of possible base relations, and is represented by the set

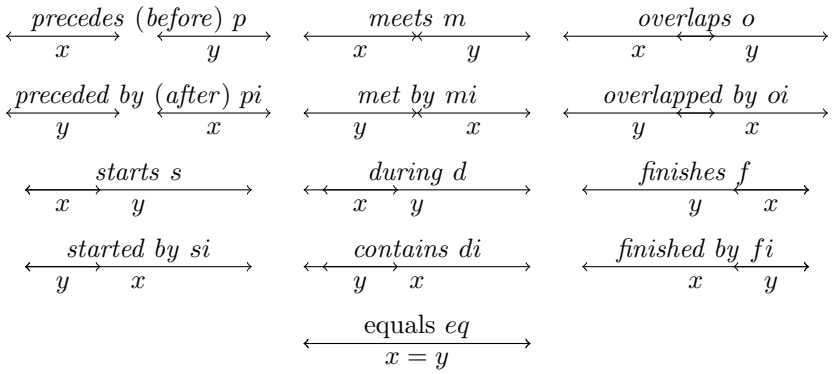


Figure 2: A representation of the 13 base relations b of IA, each one relating two potential intervals x and y as in $x\ b\ y$; here, bi denotes the converse of b (viz., b^{-1} formally)

containing them. Hence, 2^{B} represents the total set of relations. The set 2^{B} is equipped with the usual set-theoretic operations of union and intersection, the converse operation, and the *weak composition* operation denoted by the symbol $\diamond$ [23]. For all $r \in 2^{\mathsf{B}}$, we have that $r^{-1} = \bigcup\{b^{-1} \mid b \in r\}$. The weak composition ($\diamond$) of two base relations $b, b' \in \mathsf{B}$ is defined as the smallest (i.e., strongest) relation $r \in 2^{\mathsf{B}}$ that includes $b \circ b'$, or, formally, $b \diamond b' = \{b'' \in \mathsf{B} \mid b'' \cap (b \circ b') \neq \emptyset\}$, where $b \circ b' = \{(x, y) \in \mathsf{D} \times \mathsf{D} \mid \exists z \in \mathsf{D}$ such that $(x, z) \in b \wedge (z, y) \in b'\}$ is the (true) composition of b and b'. For all $r, r' \in 2^{\mathsf{B}}$, we have that $r \diamond r' = \bigcup\{b \diamond b' \mid b \in r, b' \in r'\}$.

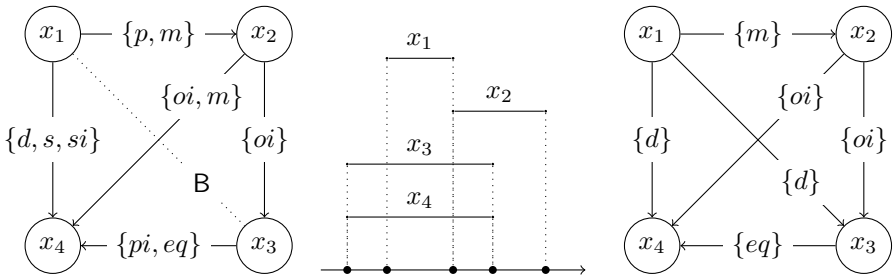
As an illustration, consider the well-known qualitative temporal constraint language of Interval Algebra (IA), introduced by Allen [18]. IA considers time intervals (as temporal entities) and the set of base relations $\mathsf{B} = \{eq, p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$ to encode knowledge about the temporal relations between intervals on the timeline, as depicted in Figure 2. Specifically, each base relation represents a particular ordering of the four endpoints of two intervals on the timeline, and eq is the identity relation Id.

Notably, most of the well-known and well-studied qualitative constraint languages, such as Interval Algebra [18] and $\mathsf{RCC8}$ [24], are in fact *relation algebras* [17].

The problem of representing and reasoning about qualitative spatial or temporal information can be modeled as a *qualitative constraint network*, defined as follows:

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

- $V = \{v_1, \ldots, v_n\}$ is a non-empty finite set of variables, each representing an entity of an infinite domain D;
- and C is a mapping $C : V \times V \to 2^{\mathsf{B}}$ such that $C(v, v) = \{\mathsf{Id}\}$ for all $v \in V$ and $C(v, v') = (C(v', v))^{-1}$ for all $v, v' \in V$, where $\bigcup \mathsf{B} = \mathsf{D} \times \mathsf{D}$.



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

Figure 3: Figurative examples of QCN terminology using IA

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

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

- a *solution* of $\mathcal{N}$ is a mapping $\sigma : V \to \mathsf{D}$ such that $\forall (u, v) \in V \times V$, $\exists b \in C(u, v)$ such that $(\sigma(u), \sigma(v)) \in b$ (see Figure 3b, as an example);
- $\mathcal{N}$ is *satisfiable* iff it admits a solution;
- a *sub*-QCN $\mathcal{N}'$ of $\mathcal{N}$, denoted by $\mathcal{N}' \subseteq \mathcal{N}$, is a QCN (V, C') such that $C'(u, v) \subseteq C(u, v)$ $\forall u, v \in V$; if in addition $\exists u, v \in V$ such that $C'(u, v) \subset C(u, v)$, then $\mathcal{N}' \subset \mathcal{N}$;
- $\mathcal{N}$ is *atomic* iff $\forall v, v' \in V$, $C(v, v')$ is a *singleton relation*, i.e., a relation $\{b\}$ with $b \in \mathsf{B}$;
- a *scenario* $\mathcal{S}$ of $\mathcal{N}$ is an atomic satisfiable sub-QCN of $\mathcal{N}$ (see Figure 3c);
- the *constraint graph* of $\mathcal{N}$ is the graph (V, E) where $\{u, v\} \in E$ iff $C(u, v) \neq \mathsf{B}$ and $u \neq v$;
- $\mathcal{N}$ is *trivially inconsistent*, denoted by $\emptyset \in \mathcal{N}$, iff $\exists v, v' \in V$ such that $C(v, v') = \emptyset$;
- $\mathcal{N}$ is the *empty* QCN on V, denoted by $\bot^V$, iff $C(u, v) = \emptyset$ for all $u, v \in V$.

Given a QCN $\mathcal{N} = (V, C)$ and $v, v' \in V$, we introduce the following operation that substitutes $C(v, v')$ with a relation $r \in 2^{\mathsf{B}}$ to produce a new, modified, QCN:

- $\mathcal{N}_{[v,v']/r}$ with $r \in 2^{\mathsf{B}}$ yields the QCN $\mathcal{N}' = (V, C')$, where $C'(v, v') = r$, $C'(v', v) = r^{-1}$, and $C'(u, u') = C(u, u')$ $\forall (u, u') \in V \times V \setminus \{(v, v'), (v', v)\}$.

We recall the definition of $\overset{\diamond}{}_G$-consistency [25] (cf [26]), which entails consistency for all triples of variables in a QCN that form triangles in an accompanying graph G, and is a basic and widely-used local consistency for reasoning with QCNs.

Definition 3. Given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, $\mathcal{N}$ is said to be $\overset{\diamond}{}_G$*-consistent* iff $\forall\{v_i, v_j\}, \{v_i, v_k\}, \{v_k, v_j\} \in E$ we have that $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$.

We note here that if G is complete, $\overset{\diamond}{}_G$-consistency becomes identical to $\diamond$-consistency [26], and, hence, $\diamond$-consistency is a special case of $\overset{\diamond}{}_G$-consistency. In the sequel, given a QCN $\mathcal{N} = (V, C)$ of some calculus and a graph $G = (V, E)$, we assume that $\overset{\diamond}{}_G(\mathcal{N})$ is computable. This assumption holds for most widely-adopted qualitative calculi [17].

3. A Dynamic Branching Approach

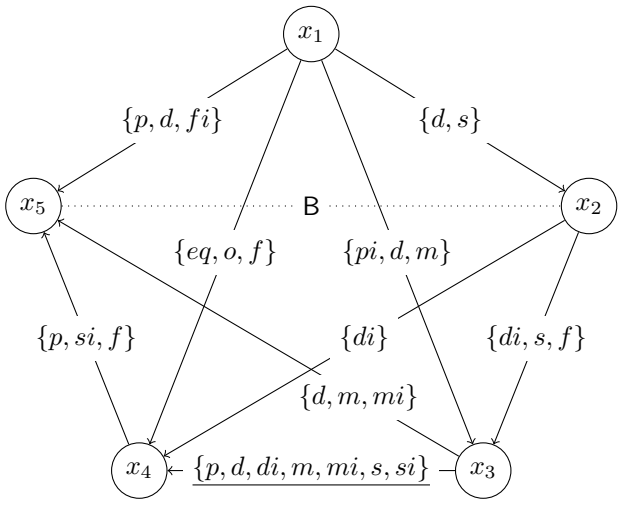
In qualitative constraint-based reasoning, the state-of-the-art approach to check the satisfiability of a given QCN $\mathcal{N}$, consists in splitting every relation r that forms a constraint between two variables in $\mathcal{N}$ into a subrelation $r' \subseteq r$ that belongs to a set of relations $\mathcal{A}$ over which the QCN becomes tractable [27]. In particular, for most widely-adopted qualitative calculi [17], such *split* sets are either known or readily available [28], and tractability is then achieved via the use of some local consistency in backtracking fashion; after every refinement of a relation r into a subrelation r', the local consistency is enforced to know whether the refinement is valid or backtracking should occur and another subrelation should be chosen at an earlier point [27, Section 2]. One of the most essential and widely-used such local consistencies is $\overset{\diamond}{}_G$-consistency, where G is either the complete graph on the variables of $\mathcal{N}$ [26] (the case of standard $\diamond$-consistency), or a *triangulation* (*chordal completion*) of the constraint graph of $\mathcal{N}$ [25].[1]

As an illustration, the subset $\mathcal{H}_{\mathsf{IA}}$ of the set of relations of Interval Algebra [30] is tractable for $\overset{\diamond}{}_G$-consistency, i.e., $\overset{\diamond}{}_G$-consistency is complete for deciding the satisfiability of any QCN defined over $\mathcal{H}_{\mathsf{IA}}$ with respect to a triangulation G of its constraint graph [25]. That subset contains exactly those relations that are transformed to propositional Horn formulas when using the propositional encoding of Interval Algebra [30]. To further facilitate the reader, let us consider the constraint $C(x_3, x_4)$ in the QCN of Interval Algebra in Figure 4. The relation $\{mi, di, si, p, m, d, s\}$ that is associated with that constraint does not appear in the subset $\mathcal{H}_{\mathsf{IA}}$ and hence tractability is not guaranteed in general, but it can be split into subrelations $\{mi\}, \{di, si\}, \{p, m\}, \{d, s\}$ with respect to $\mathcal{H}_{\mathsf{IA}}$; each of those subrelations belongs to $\mathcal{H}_{\mathsf{IA}}$.

Dynamic Selection of Subrelations via Counting Local Models

It is standard practice in the qualitative constraint-based reasoning community, and the constraint programming community in general, that, given a constraint of some QCN, a subrelation that is most likely to lead to a solution

[1] Please refer to [29] for the properties that are needed to exploit triangulations of QCNs in terms of tractability preservation.



b	$\#l$-models
p	6
d	6
di	7
m	4
mi	5
s	3
si	3

Figure 4: Given the above QCN $\mathcal{N} = (V, C)$ of IA, a partition of $C(x_3, x_4)$ with respect to the subset $\mathcal{H}_{\mathsf{IA}}$ [30] is $\{\{mi\}, \{di, si\}, \{p, m\}, \{d, s\}\}$; for this QCN, heuristic *dynamic_avg* would prioritize relation $\{p, m\}$, $\{di, si\}$, or $\{mi\}$ (subject to tie-breaks), heuristic *dynamic_sum* relation $\{p, m\}$ or $\{di, si\}$ (again subject to tie-breaks), and heuristics *static* [19], *dynamic_max*, and *dynamic_min* relations $\{d, s\}$, $\{di, si\}$, and $\{mi\}$ respectively

should be prioritized [19, 31]; in the context of finite-domain CSPs, this strategy is known as the *least-constraining value* heuristic [32].

Currently, the state of the art in qualitative constraint-based reasoning implements that selection strategy in a completely static manner. In particular, base relations of a calculus are assigned static weights a priori, and the overall weight that is associated with a subrelation corresponds to the sum of the weights of its base relations [19, 31]. In detail, a weight for a base relation is obtained by successively composing it with every possible relation and calculating the total number of resulting base relations from the compositions, which is then suitably scaled. Thus, the bigger the weight for a base relation is, the less restrictive that base relation is. For example, the weights of base relations d and s in Interval Algebra are 4 and 2 respectively and, consequently, the weight of relation $\{d, s\}$ is $4 + 2 = 6$ [19, Figure 9].

Two major problems with the aforementioned *static* strategy is that it assumes a uniform use of relations in QCNs (in the sense that weights are computed by equally considering all the relations of a calculus), and it does not exploit any structure that may be present in QCNs (a relation that is used to form more than one constraints in a given QCN, which is typically the case, may exhibit different levels of restrictiveness among those constraints).

In this paper, we propose the selection of subrelations to be *dynamic* and, in particular, based on the count of *local models* that the individual base relations of a subrelation are part of. Let $\mathcal{N}{\downarrow_{V'}}$, with $V' \subseteq V$, denote the QCN $\mathcal{N} = (V, C)$ restricted to V', we formally define the notion of local models as follows:

Definition 4 (local models)**.** Given a QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, and a constraint $C(v, v')$ with $\{v, v'\} \in E$, the *local models* of a base relation $b \in C(v, v')$ are the scenarios $\mathcal{S} = (V', C')$ of $\mathcal{N}{\downarrow_{V'}}$, with $V' = \{v, v', u\}$, such that $\{v, u\}, \{u, v'\} \in E$ and $C'(v, v') = \{b\}$.

Simply put, given a QCN (V, C), a graph $G = (V, E)$, and a constraint $C(v, v')$ with $\{v, v'\} \in E$, we count how many times a given base relation $b \in C(v, v')$ participates in the scenarios of each triangle in G that involves variables v and v', i.e., the local models from our perspective. For example, the base relation s in the constraint $C(x_3, x_4)$ in the QCN of Interval Algebra in Figure 4 participates in 3 local models, one for each of the 3 triangles involving variables $\{x_3, x_4, x_5\}$, $\{x_3, x_4, x_1\}$, and $\{x_3, x_4, x_2\}$ respectively. In that sense, our approach can be seen as being similar to a counting-based one for CSPs [21], which, as our own method, formalizes a framework that is adaptive and seeks to make branching decisions that preserve most of the solutions by determining what proportion of local solutions agree with that decision. We devise the following strategies for chosing a subrelation from a given set of subrelations:

- **dynamic_f**: for each subrelation r' find the f count of local models among each base relation $b \in r'$, where $\mathsf{f} \in \{\mathsf{max}, \mathsf{min}, \mathsf{avg}, \mathsf{sum}\}$, then choose the subrelation for which the highest such count was obtained.

In the context of counting local models, dynamic_max, dynamic_min, dynamic_avg, and dynamic_sum prioritize the subrelation with the best *most*, *least*, *on average*, and *in aggregate* supportive base relation respectively. At this point, let us revisit the QCN of Interval Algebra in Figure 4, where the relation $\{mi, di, si, p, m, d, s\}$ that is associated with the constraint $C(x_3, x_4)$ is split into subrelations $\{mi\}, \{di, si\}, \{p, m\}, \{d, s\}$ with respect to subset $\mathcal{H}_{\mathsf{IA}}$. By viewing the table that lists the count of local models for each base relation in $C(x_3, x_4)$ on the right-hand side of the figure, the reader can verify that each strategy correctly prioritizes its subrelation of choice according to its objective; as a reminder, the weights associated with the static strategy detailed earlier are provided in [19, Figure 9].

Tackling QCN*s via Incorporating Dynamic Branching*

For reference, a variation of the state-of-the-art backtracking algorithm for solving a QCN is provided in Algorithm 1, the main diffence to the one appearing in the literature [27, Section 2] being the use of dynamic selection of subrelations, in line 8, instead of selection based on static weights. Another difference is the use of a graph as a parameter, but, over the past few years, this has become a standard way of generalizing the original algorithm to exploit certain properties of a calculus that relate to graphs, see [33] and references therein.

The dynamic strategies that we described earlier are formally presented in Algorithms 2 and 3. In particular, in lines 2–4 of Algorithm 2 we calculate the count of local models for each base relation of each subrelation that pertains to a given constraint. This calculation is performed via a call to Algorithm 3. After

Algorithm 1: Refinement($\mathcal{N}$, G, $\mathcal{A}$, f)

in : A QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, a subset $\mathcal{A} \subseteq 2^{\mathsf{B}}$, and a function $\mathsf{f} \in \{\mathsf{max}, \mathsf{min}, \mathsf{avg}, \mathsf{sum}\}$.
out : A refinement of $\mathcal{N}$ with respect to G over $\mathcal{A}$, or $\bot^V$.

```
1  begin
2      N ← ◇_G(N);
3      if ∅ ∈ N then
4          return ⊥^V;
5      if ∀{v, v'} ∈ E, C(v, v') ∈ A then
6          return N;
7      (v, v') ← {v, v'} ∈ E such that C(v, v') ∉ A;
8      foreach r ∈ dynamicSelection(N, G, A, (v, v'), f) do
9          result ← Refinement(N_{[v,v']/r}, G, A, f);
10         if result ≠ ⊥^V then
11             return result;
12     return ⊥^V;
```

Algorithm 2: dynamicSelection($\mathcal{N}$, G, $\mathcal{A}$, (v, v'), f)

in : A QCN $\mathcal{N} = (V, C)$, a graph $G = (V, E)$, a subset $\mathcal{A} \subseteq 2^{\mathsf{B}}$, a pair of variables (v, v'), and a function $\mathsf{f} \in \{\mathsf{max}, \mathsf{min}, \mathsf{avg}, \mathsf{sum}\}$.
out : A relation $r \in \mathcal{A}$.

```
1  begin
2      counter ← hashTable();
3      foreach r ∈ {r_1, ..., r_n ∈ A | {r_1, ..., r_n} is a partition of C(v, v')} do
4          counter[r] ← f{localModels(b, N, G, (v, v')) | b ∈ r};
5      while counter is not empty do
6          r ← counter key paired with the maximum value;
7          remove r from counter;
8          yield r;
```

obtaining the count of models for each such base relation, we implement the chosen strategy by applying the respective function among $\{\mathsf{max}, \mathsf{min}, \mathsf{avg}, \mathsf{sum}\}$ on the results. Now, each subrelation is associated with a number, a dynamic weight, and this information is stored in a hash map (line 4). Then, in lines 5–8 the subrelation with the highest dynamic weight is prioritized each time there is a need for a new subrelation to be tried out in an assignment.

Complexity Analysis. Given the fact that for a calculus the number of its base relations, i.e., $|\mathsf{B}|$, can be viewed as a constant, Algorithm 2 calculates the count of local models for a base relation in a given constraint in linear time in the maximum degree of the graph G that is used as a parameter; each subsequent prioritization of a subrelation based on those calculated counts (lines 5–8) takes

Algorithm 3: localModels$(b, \mathcal{N}, G, (v, v'))$

```
in  : A base relation b, a QCN 𝒩 = (V, C), a graph G = (V, E), and a
      pair of variables (v, v').
out : An integer.
1 begin
2     count ← 0;
3     foreach u ∈ N_G(v) ∩ N_G(v') do
4         foreach (b', b'') ∈ C(v, u) × C(u, v') do
5             if b ∈ b' ◇ b'' then
6                 count ← count + 1;
7     return count;
```

constant time. In particular, given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, Algorithm 2 runs in $\Theta(\Delta(G))$ time (where Δ denotes maximum degree). In practice, there was no noticable slowdown for the dataset that we consider in this paper (see following subsection), which is not surprising, as the search space for solving a QCN is $O(|\mathsf{B}|^{|E|})$ in general.

Evaluation of Heuristics

In this section we evaluate the proposed dynamic branching heuristics, as well as the state-of-the-art static branching strategy that appears in the literature, with respect to the fundamental reasoning problem of *satisfiability checking* of QCNs. Specifically, we explore the *efficiency* of the involved heuristics in determining the satisfiability of a given network instance when used in the standard backtracking algorithm (see Algorithm 1), and investigate their *fitness score* too, which is the difference *"% of times a heuristic* f *is the best choice" − "% of times a heuristic* f *is the worst choice"*; clearly, *fitness score* $\in [-100\%, 100\%]$. Finally, we also present results for two virtual portfolios of reasoners that always make the *best* and *worst* choice of a heuristic respectively for a given network instance. Specifically, these portfolios are virtual in the sense that they do not actually exist beforehand, but are established only after the evaluation is fully completed by looking at which heuristic performed best or worst for each instance and considering that heuristic as the choice of the virtual portfolio *best* or *worst* respectively. Therefore, the virtual portofolios *best* and *worst* allow us to set upper and lower bounds respectively of the expected performance of the various heuristics.

Technical specifications. The evaluation was carried out on a computer with an Intel Core i7-8565U processor, 16 GB of RAM, and the Ubuntu 18.04.4 LTS x86_64 OS. All algorithms were coded in Python and run using the PyPy intepreter under version 5.10.0, which implements Python 2.7.13. Only one CPU core was used per run.

Table 1: Evaluation with random IA networks generated using model $\mathsf{H}(\mathsf{n}=40,\mathsf{d})$ [34]; $\frac{\text{median}|\text{average}|\text{maximum \# of visited nodes}}{\text{median}|\text{average}|\text{maximum CPU time}}$

d	*best*	*static*	*dynamic*_max	*dynamic*_min	*dynamic*_avg	*dynamic*_sum	*worst*
9	$\frac{62.0\|61.8\|95.0}{0.0s\|0.0s\|10.8s}$	$\frac{78.0\|79.4\|413.0}{0.0s\|\mathbf{0.0s}\|11.0s}$	$\frac{107.0\|112.0\|258.0}{0.0s\|0.0s\|10.8s}$	$\frac{66.0\|\mathbf{65.9}\|144.0}{0.0s\|0.0s\|10.9s}$	$\frac{78.0\|79.5\|350.0}{0.0s\|0.0s\|10.8s}$	$\frac{94.0\|95.6\|520.0}{0.0s\|0.0s\|10.9s}$	$\frac{114.0\|120.4\|520.0}{0.0s\|0.0s\|11.0s}$
10	$\frac{52.0\|52.6\|168.0}{0.0s\|0.0s\|10.9s}$	$\frac{71.0\|125.8\|8.9k}{0.0s\|0.1s\|10.9s}$	$\frac{85.0\|101.3\|2.0k}{0.0s\|\mathbf{0.0s}\|10.9s}$	$\frac{60.0\|\mathbf{89.9}\|2.9k}{0.0s\|0.0s\|11.0s}$	$\frac{67.0\|104.4\|8.7k}{0.0s\|0.1s\|10.9s}$	$\frac{81.0\|129.6\|6.6k}{0.0s\|0.1s\|10.9s}$	$\frac{108.0\|227.7\|8.9k}{0.0s\|0.1s\|11.0s}$
11	$\frac{59.0\|679.3\|388.0k}{0.0s\|0.5s\|4.4m}$	$\frac{174.0\|2.5k\|500.3k}{0.1s\|1.7s\|5.3m}$	$\frac{143.5\|2.1k\|388.0k}{0.1s\|1.4s\|4.4m}$	$\frac{141.0\|\mathbf{1.7k}\|546.8k}{0.1s\|\mathbf{1.2s}\|6.2m}$	$\frac{133.0\|2.1k\|619.7k}{0.1s\|1.4s\|6.9m}$	$\frac{181.0\|2.6k\|474.2k}{0.1s\|1.9s\|5.4m}$	$\frac{682.0\|5.0k\|619.7k}{0.4s\|3.4s\|6.9m}$
12	$\frac{931.5\|10.3k\|953.5k}{0.7s\|7.4s\|10.7m}$	$\frac{4.4k\|24.6k\|958.6k}{3.1s\|16.8s\|10.7m}$	$\frac{3.5k\|21.0k\|1.0m}{2.6s\|15.1s\|11.6m}$	$\frac{3.1k\|18.7k\|1.0m}{2.3s\|13.5s\|12.0m}$	$\frac{2.9k\|\mathbf{18.6k}\|996.1k}{2.2s\|\mathbf{13.3s}\|11.4m}$	$\frac{4.7k\|24.6k\|953.5k}{3.5s\|17.4s\|10.8m}$	$\frac{10.0k\|34.9k\|1.0m}{7.4s\|24.5s\|12.0m}$
13	$\frac{2.0k\|7.6k\|214.9k}{1.6s\|5.9s\|2.7m}$	$\frac{3.4k\|11.9k\|277.3k}{2.6s\|8.7s\|3.1m}$	$\frac{3.2k\|11.2k\|302.8k}{3.0s\|9.8s\|4.1m}$	$\frac{2.7k\|\mathbf{10.8k}\|245.6k}{2.1s\|\mathbf{8.2s}\|3.0m}$	$\frac{2.9k\|10.9k\|272.8k}{2.3s\|8.4s\|3.2m}$	$\frac{3.4k\|11.7k\|283.8k}{2.7s\|9.1s\|3.5m}$	$\frac{4.6k\|15.7k\|302.8k}{3.7s\|12.3s\|4.1m}$
14	$\frac{460.0\|1.3k\|58.1k}{0.4s\|1.1s\|43.9s}$	$\frac{720.0\|1.9k\|64.9k}{0.5s\|\mathbf{1.4s}\|47.5s}$	$\frac{700.5\|1.9k\|77.5k}{0.5s\|1.5s\|58.0s}$	$\frac{584.5\|\mathbf{1.6k}\|64.9k}{0.6s\|1.6s\|1.0m}$	$\frac{619.0\|1.7k\|63.8k}{0.6s\|1.6s\|55.8s}$	$\frac{708.5\|2.0k\|106.0k}{0.6s\|1.6s\|1.3m}$	$\frac{933.0\|2.5k\|106.0k}{0.8s\|2.1s\|1.3m}$
15	$\frac{113.0\|220.7\|2.6k}{0.1s\|0.2s\|11.2s}$	$\frac{179.0\|365.2\|5.4k}{0.1s\|0.3s\|11.2s}$	$\frac{168.5\|350.2\|4.4k}{0.1s\|0.3s\|11.3s}$	$\frac{157.0\|\mathbf{285.7}\|2.8k}{0.1s\|\mathbf{0.2s}\|11.4s}$	$\frac{167.0\|317.9\|4.8k}{0.1s\|0.3s\|11.3s}$	$\frac{176.0\|360.2\|5.9k}{0.1s\|0.3s\|11.3s}$	$\frac{251.5\|463.7\|5.9k}{0.2s\|0.4s\|11.4s}$

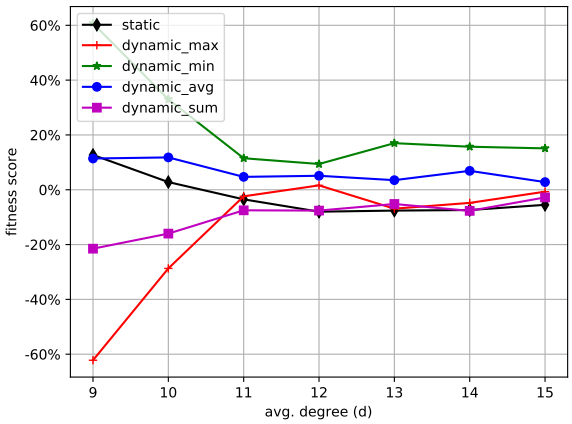


Figure 5: Fitness score of each strategy for the instances of Table 1; *"% of times a heuristic* f *is the best choice" − "% of times a heuristic* f *is the worst choice"*

Dataset. We generated 7 000 *random* instances of Interval Algebra using model $\mathsf{H}(\mathsf{n} = 40, \mathsf{d})$ [34], 1 000 of $n = 40$ variables for each constraint graph degree value $\mathsf{d} \in \{9, 10, 11, 12, 13, 14, 15\}$ specifically, and 4 000 *structured* instances of Interval Algebra using model $\mathsf{BA}(\mathsf{n} = 80, \mathsf{m}, \mathsf{3CNF})$ [35], 1 000 of $n = 80$ variables for each constraint graph *preferential attachment* [36] value $\mathsf{m} \in \{4, 5, 6, 7\}$ specifically. In particular, model $\mathsf{H}(\mathsf{n}, \mathsf{d})$ [34] considers constraint networks of a regular graph structure, and was chosen for its suitability and consistency with respect to other traditional works in the literature, and model $\mathsf{BA}(\mathsf{n}, \mathsf{m}, \mathsf{3CNF})$ [35] is a more recent model that considers constraint networks exhibiting a scale-free structure, and was chosen exactly for its ability to induce structure in network instances and consequently allow us to have a more varied dataset. In both of the aforementioned generation models, constraints were picked from the set of relations expressible in 3CNF when transformed into first-order formulae [34], in order to obtain instances that maximally challenge search. Finally, regarding the graphs that were given as input to our algorithms, the *maximum cardinality search* algorithm [37] was used to obtain triangulations of the constraint graphs of our QCNs. The choice of such chordal graphs was not only reasonable but also crucial given their important theoretical and practical implications in qualitative constraint-based spatial and temporal reasoning, as reviewed in [33]; in short, the use of those graphs itself was inspired by [38, 29, 39] among other works, where preliminary results pertainining to tree decompositions were established.

Results. Regarding the generation model $\mathsf{H}(\mathsf{n} = 40, \mathsf{d})$, the main results are presented in Table 1. The dynamic strategies of *dynamic*_min and *dynamic*_avg are up to 20% faster on average than the *static* one in the phase transition of

Table 2: Evaluation with structured IA networks generated using model BA($n = 80, m, 3CNF$) [35]; $\frac{\text{median|average|maximum \# of visited nodes}}{\text{median|average|maximum CPU time}}$

m	*best*	*static*	*dynamic*_max	*dynamic*_min	*dynamic*_avg	*dynamic*_sum	*worst*
4	$\frac{144.0\|144.1\|177.0}{0.0s\|0.0s\|10.7s}$	$\frac{166.0\|167.1\|257.0}{0.0s\|\mathbf{0.0s}\|10.8s}$	$\frac{213.0\|216.0\|332.0}{0.0s\|0.0s\|10.8s}$	$\frac{146.0\|\mathbf{146.3}\|320.0}{0.0s\|0.0s\|10.8s}$	$\frac{178.0\|177.0\|231.0}{0.0s\|0.0s\|10.8s}$	$\frac{198.0\|198.9\|321.0}{0.0s\|0.0s\|10.7s}$	$\frac{219.0\|222.3\|332.0}{0.0s\|0.0s\|10.8s}$
5	$\frac{113.0\|114.5\|550.0}{0.0s\|0.0s\|10.8s}$	$\frac{128.0\|154.4\|1.0k}{0.0s\|0.1s\|10.8s}$	$\frac{155.0\|173.3\|1.6k}{0.0s\|0.1s\|11.0s}$	$\frac{124.0\|\mathbf{139.9}\|2.9k}{0.0s\|\mathbf{0.1s}\|10.9s}$	$\frac{141.0\|159.8\|1.2k}{0.0s\|0.1s\|10.8s}$	$\frac{151.0\|177.8\|1.2k}{0.0s\|0.1s\|10.9s}$	$\frac{178.0\|226.3\|2.9k}{0.1s\|0.1s\|11.0s}$
6	$\frac{121.0\|452.3\|72.5k}{0.1s\|0.6s\|1.7m}$	$\frac{235.0\|4.7k\|1.3m}{0.3s\|5.4s\|23.7m}$	$\frac{201.0\|7.4k\|3.8m}{0.2s\|8.1s\|1.1h}$	$\frac{172.0\|1.4k\|460.4k}{0.2s\|1.8s\|8.8m}$	$\frac{182.5\|\mathbf{933.4}\|139.6k}{0.2s\|\mathbf{1.2s}\|3.1m}$	$\frac{223.0\|4.0k\|2.0m}{0.3s\|4.6s\|34.5m}$	$\frac{337.5\|10.6k\|3.8m}{0.4s\|11.8s\|1.1h}$
7	$\frac{34.0\|82.4\|3.4k}{0.0s\|0.1s\|11.1s}$	$\frac{51.0\|178.9\|10.4k}{0.1s\|0.2s\|15.2s}$	$\frac{50.0\|168.7\|17.9k}{0.1s\|0.2s\|27.7s}$	$\frac{47.0\|\mathbf{110.8}\|3.4k}{0.1s\|\mathbf{0.2s}\|11.3s}$	$\frac{49.0\|129.4\|5.0k}{0.1s\|0.2s\|11.2s}$	$\frac{51.0\|206.6\|22.6k}{0.1s\|0.3s\|35.2s}$	$\frac{74.5\|252.2\|22.6k}{0.1s\|0.4s\|35.2s}$

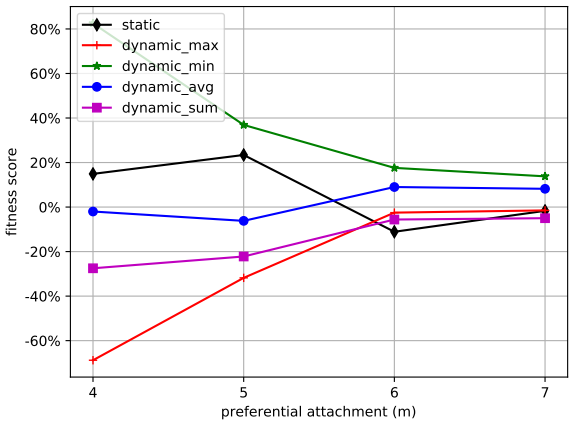


Figure 6: Fitness factor of each strategy for the instances of Table 2; *"% of times a heuristic* f *is the best choice" − "% of times a heuristic* f *is the worst choice"*

the tested instances.[2] Specifically, the phase transition covers mostly the case where $\mathsf{d} = 12$, and a little less the case where $\mathsf{d} = 13$. With respect to the rest of the dynamic heuristics, viz., *dynamic*_max and *dynamic*_sum, the results suggest that *dynamic*_max outperforms *static* by a small margin on average in the phase transition, whilst *dynamic*_sum almost mimics the performance of *static*, if not arguably being a little worse than *static* overall. This last finding informs us that relying too much on the number of base relations of a relation (viz., the cardinality of a relation) is a bad choice in general, i.e., it is better to focus on few base relations individually, where each one appears in many local models (quality), than on many base relations aggregately, where each one appears in few local models (quantity). The aforementioned results are depicted from a different perspective and complemented in Figure 5, where the fitness score for each heuristic is detailed. The superiority of heuristics *dynamic*_min and *dynamic*_avg among all strategies becomes even more so clear, and the marginal performance gains of *dynamic*_max, and *dynamic*_sum at times with respect to *static* in the phase transition are well-captured by their fitness scores too. Finally, at this point, it is interesting to observe the performance of the virtual portfolios of reasoners *best* and *worst*; as a reminder these always make the *best* and *worst* choice of a heuristic respectively for a given network instance. The performance of portfolio *best* in particular allows us to be optimistic about future research in dynamic strategies, since it shows that there is still a lot of room for improvement. Specifically, research could be carried out both in

[2]Even though the improvement for this particular dataset may not seem that drastic, bear in mind that the instances of *this* dataset have little to no structure as their constraint graphs are regular graphs.

terms of defining new dynamic strategies and in terms of devising selection protocols that choose among already existing strategies. In fact, a first effort to obtain a selection protocol that chooses among the better performing strategies is presented in the next section.

Regarding the generation model $\mathsf{BA}(\mathsf{n} = 80, \mathsf{m}, \mathsf{3CNF})$, the results are presented in Table 2 and Figure 6. Here, *dynamic*_min and *dynamic*_avg are up to 3 and 5 times faster on average respectively than the *static* one in the phase transition of the tested instances, which appears for $\mathsf{m} = 6$. The rest of the results are qualitatively similar to the previous dataset.

4. A Selection Protocol for Heuristics

It was shown in the previous section that heuristics *dynamic*_min and *dynamic*_avg are the better performing ones for the dataset that was used here. In particular, *dynamic*_min was found to perform better than any other heuristic for the majority of the instances, whereas *dynamic*_avg appeared to be more advantageous when it came to some very difficult instances of the phase transition, albeit only marginally so. In this section, given such a dataset of instances, we deal with the following two questions:

- Can we parsimoniously obtain a valid ranking of the involved heuristics, i.e., one that does not involve going through the entire dataset and solving all QCNs (as we did in Section 3)?
- Can we have a selection protocol that will both effectively and efficiently synthesize the involved heuristics into a better performing meta-heuristic?

Regarding the first question, we will show that this can be done with a minimal amount (i.e., negligible in terms of computational time) of dynamic preprocessing of a given set of QCNs, and regarding the second question, we will demonstrate that the aforementioned preprocessing phase can be further utilized to obtain a meta-heuristic that outperforms all other heuristics discussed so far in the phase transition region. However, we must note that the performance of our meta-heuristic to be presented in this section leaves a lot to be desired when compared with the virtual portofolio *best*, but we think that it is nevertheless encouraging and that it sets the basis on how future research could proceed (we make a remark on that in Section 5).

A Selection Protocol Algorithm

In Algorithm 4 we present our selection protocol algorithm, called Oracle, which given a QCN decides which (static or dynamic) heuristic should be used to tackle the QCN; as a reminder, these heuristics are *dynamic*_min, *dynamic*_avg, *dynamic*_sum, *dynamic*_max, and *static*. In a sense, Algorithm 4 tries to mimic the behaviour of Algorithm 1 by sequentially trying to assign subrelations to constraints with the purpose of reaching a locally consistent refinement of the QCN over a given subset $\mathcal{A}$ of relations (with respect to a given graph G).

Algorithm 4: Oracle($\mathcal{N}$, G, $\mathcal{A}$, $k = 7$)

```
   in  : A QCN 𝒩 = (V, C), a graph G = (V, E), a subset 𝒜 ⊆ 2^B, and a
         positive integer k (defaulted to 7).
   out : A function f ∈ {max, min, avg, sum}, or null.
 1 begin
 2     if ∀{v, v'} ∈ E, C(v, v') ∈ 𝒜 then
 3         return null;
 4     metrics ← hashTable();
 5     foreach g ∈ {max, min, avg, sum, null} do
 6         metrics[g] ← 0;
 7     foreach g ∈ {max, min, avg, sum, null} do
 8         foreach _ ∈ {0, 1, ..., k − 1} do
 9             ℳ = (V', C') ← 𝒩;
10             counter ← 0;
11             do
12                 (v, v') ← random {v, v'} ∈ E such that C(v, v') ∉ 𝒜;
13                 counter ← counter + 1;
14                 if g ≠ null then
15                     r ← dynamicSelection(ℳ, G, 𝒜, (v, v'), g);
16                 else
17                     r ← staticSelection({r_1, ..., r_n ∈ 𝒜 | {r_1, ..., r_n} is a
                         partition of C(v, v')}) see, e.g., [19, Figure 9]
18                 ℳ' ← ◇_G(ℳ_[v,v']/r);
19                 if ∅ ∈ ℳ' then
20                     break;
21                 ℳ ← ℳ';
22             while ∃{v, v'} ∈ E | C(v, v') ∉ 𝒜;
23             assignedConstraints ← {C'(v, v') | {v, v'} ∈ E ∧ C'(v, v') ∈ 𝒜};
24             metrics[g] ← metrics[g] + |assignedConstraints| / counter;
25     return metrics key paired with the maximum value;
```

However, the main difference with Algorithm 1 is that no backtracking occurs and the aforementioned sequential treatment stops once failure in encountered (i.e., an assignment of the empty relation takes place). After each assignment of a subrelation to a constraint, or, in other words, a refinement of a relation r (associated with a constraint) into a subrelation r', $\overset{\diamond}{}_G$-consistency is enforced to know whether the refinement is valid or it leads to failure. This process is performed k random times for each heuristic, i.e., each time a random ordering of constraints is processed, and metrics are kept while doing so in order to find the better performing heuristic in the end. In what follows we describe the algorithm in more detail. In lines 2–3 of the algorithm null is returned in case all of the relations that are associated with the constraints of the given QCN already belong to the provided subset $\mathcal{A}$; in this case, none of the heuristics will be used,

as no refinement will ever take place (see again lines 5–6 in Algorithm 1). As a convention, in our algorithm null suggests that one can stick to the original *static* strategy. In lines 4–6 a hash map is created to keep track of a performance metric for each heuristic. As it is seen later on in lines 23–24, the metric of choice is the number of constraints that are assigned a subrelation from $\mathcal{A}$ per step, which is a direct estimate of how fast the QCN is being refined over $\mathcal{A}$, or, in more technical terms, of how few nodes are visited as the QCN is being refined over $\mathcal{A}$. To avoid confusion, in each step a single constraint is assigned a subrelation from $\mathcal{A}$ (line 18), but the subsequent application of $\diamond_G$-consistency may (and will almost always) lead to other constraints being refined and potentially assigned with subrelations from $\mathcal{A}$; in that sense, the choice of the proper subrelation for that initial and subsequent single assignments, which pertains to the choice of the heuristic that will be used to begin with, is important. Finally, the whole sequential treatment that we described in the beginning is implemented within lines 11–22 for each of the heuristics involved and for k random times for each heuristic, and the best performing heuristic is returned in line 25. Specifically, within lines 11–22 the constraints of the given QCN are sequentially assigned a subrelation from subset $\mathcal{A}$ until failure occurs due to the respective sequential application of $\diamond_G$-consistency. It is worth noting that if $k = 1$ and the same ordering of constraints is used as in Algorithm 1 without any failure occuring for any of the heuristics, the proposed heuristic returned in line 25 is no longer an estimate but an exact decision of a heuristic that will perform best for Algorithm 1 (this would also essentially mean that Algorithm 1 would refine a QCN over $\mathcal{A}$ in a backtrack-free manner).

Complexity Analysis. Due to the incremental functionality of the state-of-the-art algorithm for enforcing $\diamond_G$-consistency [25] (see also Theorem 1 in [40], Section 3] and the surrounding text), Algorithm Oracle runs in $O(k \cdot \alpha)$ time, where $\alpha = O(\Delta(G)|E||\mathsf{B}|)$ is the runtime of the state-of-the-art algorithm for enforcing $\diamond_G$-consistency [25]. Since k should be treated as a constant (it defaults to 7, and in our experimentation we did not observe any meaningful gain in increasing the number of random orderings of constraints to be processed for each heuristic), the total runtime is $O(\Delta(G)|E||\mathsf{B}|)$. In practice, the overall computational impact was found to be negligible, which is not all surprising since the application of $\diamond_G$-consistency on QCN instances requires a minimal amount of time when compared to the main backtracking module that searches for a solution among an exponential number of candidates (see Algorithm 1).

Evaluation of Selection Protocol Algorithm

After having introduced our selection protocol algorithm, viz., Algorithm Oracle, and having analyzed its computational properties, it is important to evaluate whether it achieves the objectives specified earlier in this section, namely, whether (*i*) it can be used to parsimoniously obtain a valid ranking of the involved heuristics, i.e., one that does not involve going through the entire dataset and solving all QCNs, and whether (*ii*) it implements a selection protocol that

Table 3: Heuristic selection by Algorithm Oracle in the phase transition of the different datasets used here, where $\%s$ denote that a heuristic is selected some percentage (%) amount of times; in parenthesis, the on average heuristic selection for an entire dataset, not just its phase transition

heuristic	H($n = 40, d \in \{12, 13\}$) [34]	BA($n = 80, m = 6$, 3CNF) [35]
static	0.4% (0.6%)	2.9% (2.6%)
*dynamic*_max	2.0% (3.2%)	9.5% (7.6%)
*dynamic*_min	57.9% (56.7%)	52.3% (57.3%)
*dynamic*_avg	39.6% (39.0%)	32.7% (30.2%)
*dynamic*_sum	0.1% (0.5%)	2.6% (2.3%)

will both effectively and efficiently synthesize the involved heuristics into a better performing meta-heuristic. We tackle these open questions in what follows; we used the same technical specifications for the evaluation as in Section 3.

Ranking of Heuristics. In Table 3 we present results regarding the heuristic selection of the algorithm for each of the datasets used in this paper. For conciseness, we focus on the phase transition of these datasets, but we also present the on average heuristic selection in parentheses; in general, there was only little variation accross instances of different density for each dataset, which is in accordance with the design of the algorithm (it should not affect the ranking of the heuristics whether an instance of a specific type/model is sparse or dense). The results are in line with the performance of the different heuristics as they were evaluated in the previous section. In particular, in the evaluation of Section 3, the dynamic strategies *dynamic*_min and *dynamic*_avg were found to clearly outperform all the rest of the heuristics, forming a league of their own. Regarding these two heuristics specifically, *dynamic*_min was shown to have a slight edge over *dynamic*_avg overall, with the exception of some very hard instances in the phase transition where *dynamic*_avg perfomed better. With respect to the rest of the heuristics, *dynamic*_max outperformed *static* by a small margin, and *dynamic*_sum almost matched the performance of *static*, yet it could be argued that it performed a little worse than *static* overall. At this point, we can argue that we can parsimoniously obtain a valid ranking of the involved heuristics, with just a minimal amount of dynamic preprocessing. Indeed, so far we have

Table 4: Evaluation with random IA networks generated using model $\mathsf{H}(\mathsf{n} = 40, \mathsf{d})$ [34], where a **green** font denotes best performance among all heuristics (when compared with Table 1); median|average|maximum # of visited nodes / median|average|maximum CPU time

d	*best*	*random*	*oracle*	*worst*
9	62.0|61.8|95.0 / 0.0s|0.0s|10.8s	85.0|86.8|210.0 / 0.0s|**0.0s**|10.9s	70.0|**70.7**|146.0 / 0.0s|0.0s|10.9s	114.0|120.4|520.0 / 0.0s|0.0s|11.0s
10	52.0|52.6|168.0 / 0.0s|0.0s|10.9s	77.0|108.1|3.9k / 0.0s|**0.1s**|10.9s	64.0|**95.5**|3.2k / 0.0s|0.1s|11.1s	108.0|227.7|8.9k / 0.0s|0.1s|11.0s
11	59.0|679.3|388.0k / 0.0s|0.5s|4.4m	256.5|2.2k|490.1k / 0.1s|1.5s|5.4m	137.0|**1.7k**|546.8k / 0.1s|**1.2s**|6.3m	682.0|5.0k|619.7k / 0.4s|3.4s|6.9m
12	931.5|10.3k|953.5k / 0.7s|7.4s|10.7m	4.7k|21.7k|996.7k / 3.6s|15.4s|11.4m	3.0k|**17.7k**|996.1k / 2.3s|**13.0s**|11.6m	10.0k|34.9k|1.0m / 7.4s|24.5s|12.0m
13	2.0k|7.6k|214.9k / 1.6s|5.9s|2.7m	3.4k|11.3k|267.0k / 2.7s|8.8s|3.3m	2.7k|**10.8k**|245.6k / 2.1s|**8.1s**|3.0m	4.6k|15.7k|302.8k / 3.7s|12.3s|4.1m
14	460.0|1.3k|58.1k / 0.4s|1.1s|43.9s	679.5|1.8k|62.9k / 0.6s|**1.5s**|52.3s	615.0|**1.7k**|64.9k / 0.6s|1.6s|1.0m	933.0|2.5k|106.0k / 0.8s|2.1s|1.3m
15	113.0|220.7|2.6k / 0.1s|0.2s|11.2s	172.1|338.0|5.5k / 0.1s|**0.3s**|11.3s	160.0|**303.4**|4.8k / 0.2s|0.3s|11.3s	251.5|463.7|5.9k / 0.2s|0.4s|11.4s

shown that our algorithm is able to parsimoniously establish a ranking of the heuristics that matches their performance in practice (with respect to the evaluation that took place in Section 3, as also mentioned earlier), and that it can do so with minimal effort (review again the complexity analysis part). The exact computational impact was found to be negligible and is included in the results that are presented in what follows, viz., Tables 4 and 5. In fact, as the findings in Table 3 reveal, we could make an informed decision about which heuristic would respectively perform best by preprocessing around 10 instances of a given dataset of a specific type/model (either $\mathsf{H}(\mathsf{n}, \mathsf{d})$ [34] or $\mathsf{BA}(\mathsf{n}, \mathsf{m}, \mathsf{3CNF})$ [35] in this paper); this corresponds to 1% of the total instances used here.

Performance of Meta-heuristic. A first efficient yet basic meta-heuristic can be synthesized by randomly choosing among the available heuristics with

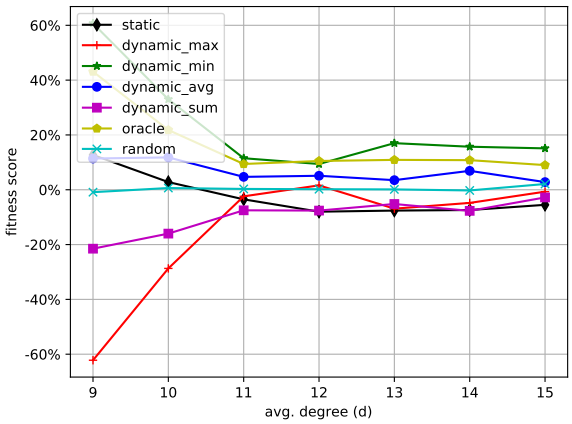


Figure 7: Fitness score of each strategy for the instances of Tables 1 and 4; *"% of times a heuristic* f *is the best choice" − "% of times a heuristic* f *is the worst choice"*

equal probability; we call this meta-heuristic *random*. Although the random aspect of this meta-heuristic might suggest poor performance, we note that we randomly choose among heuristics that were shown to perform equal or better overall than the state of the art. A more informed and efficient meta-heuristic can be synthesized by simply trusting the output of Algorithm 4, viz., Oracle, for every given QCN that it receives as input, and then using that output, viz., the suggested heuristic, to perform backtracking search utilizing it (see Algorithm 1); we call this meta-heuristic *oracle* (to match the name of Algorithm 4). We show here that this simple strategy is also quite effective, in the sense that it allows one to synthesize the available heuristics (static or dynamic) into an overall better performing meta-heuristic in the phase transition. Specifically, regarding the generation model $\mathsf{H}(\mathsf{n} = 40, \mathsf{d})$, the main results are presented in Table 4 and complemented in Figure 7; for the meta-heuristic *random* we present the on average performance with respect to 7 random runs. The meta-heuristic *oracle* not only outperforms the meta-heuristic *random*, which was expected, but also all the invidual heuristics in the phase transition (please compare with Table 1), which covers mostly the case where $\mathsf{d} = 12$, and a little less the case where $\mathsf{d} = 13$. In particular, we can further ameliorate the already improved results by up to about 5% (with respect to the number of visited nodes) for the random instances here. Perhaps surprisingly, our proposed meta-heuristic *oracle* perfoms better in the phase transition, where it is hard to do so as any wrong choice could cost more, than outside of it, where the selection of the better performing heuristic should be more straightforward as not a lot of backtracking occurs. We can explain this as follows. By reviewing the logs, we observed that the more difficult the instance to be solved is, the bigger the difference between the metrics of the various heuristics (line 24 in Algorithm Oracle) will

Table 5: Evaluation with structured IA networks generated using model BA(n = 80, m, 3CNF) [35]; $\frac{\text{median|average|maximum \# of visited nodes}}{\text{median|average|maximum CPU time}}$

m	*best*	*random*	*oracle*	*worst*
4	$\frac{144.0\|144.1\|177.0}{0.0s\|0.0s\|10.7s}$	$\frac{180.6\|181.1\|254.9}{0.0s\|\mathbf{0.0s}\|10.8s}$	$\frac{151.0\|\mathbf{153.3}\|320.0}{0.0s\|0.1s\|11.0s}$	$\frac{219.0\|222.3\|332.0}{0.0s\|0.0s\|10.8s}$
5	$\frac{113.0\|114.5\|550.0}{0.0s\|0.0s\|10.8s}$	$\frac{145.1\|161.0\|967.9}{0.0s\|\mathbf{0.1s}\|10.9s}$	$\frac{131.0\|\mathbf{145.3}\|1.5k}{0.1s\|0.1s\|11.2s}$	$\frac{178.0\|226.3\|2.9k}{0.1s\|0.1s\|11.0s}$
6	$\frac{121.0\|452.3\|72.5k}{0.1s\|0.6s\|1.7m}$	$\frac{230.0\|4.3k\|1.2m}{0.3s\|4.8s\|21.0m}$	$\frac{175.0\|\mathbf{778.4}\|139.6k}{0.3s\|\mathbf{1.1s}\|3.2m}$	$\frac{337.5\|10.6k\|3.8m}{0.4s\|11.8s\|1.1h}$
7	$\frac{34.0\|82.4\|3.4k}{0.0s\|0.1s\|11.1s}$	$\frac{53.3\|151.8\|6.3k}{0.1s\|\mathbf{0.2s}\|11.3s}$	$\frac{48.0\|\mathbf{126.0}\|5.0k}{0.1s\|0.2s\|11.3s}$	$\frac{74.5\|252.2\|22.6k}{0.1s\|0.4s\|35.2s}$

be; thus, the better performing heuristic is more likely to stand out among the inferior ones. Regarding the generation model BA(n = 80, m, 3CNF), the results are presented in Table 5 and Figure 8. In this case, we can further ameliorate the already improved results by up to about 15% (with respect to the number of visited nodes) for the structured instances here (please also compare with Table 2). Again, the improvement involves instances in the phase transition, which appears for m = 6. The rest of the results are qualitatively similar to the previous dataset. Finally, since the runtime distribution is heavy-tailed for both datasets, the interested reader may want to look into the 0.5^{th} percentile of most difficult instances pertaining to Table 4 (and Table 1 by extention) and Table 5 (and Table 2 by extention) for each strategy discussed so far in this work, depicted in Figures A.9 and A.10 respectively in Appendix A.

5. Related Work

With regard to dynamic branching heuristics in the context of QCNs, to the best of our knowledge the work presented here is the first to move away from the static weighting scheme dictated by the literature; see again Figure 1 and the approach in [19] with regard to that aspect. However, in [19] the authors also propose a different approach that is based on knowledge of the structure of scenarios for some given set of QCNs. In particular, they propose to extract scenarios from a small number of QCN instances of a given dataset of some type/model, without the use of any heuristic whatsoever, and then examine these scenarios and determine which base relations are most likely to appear in

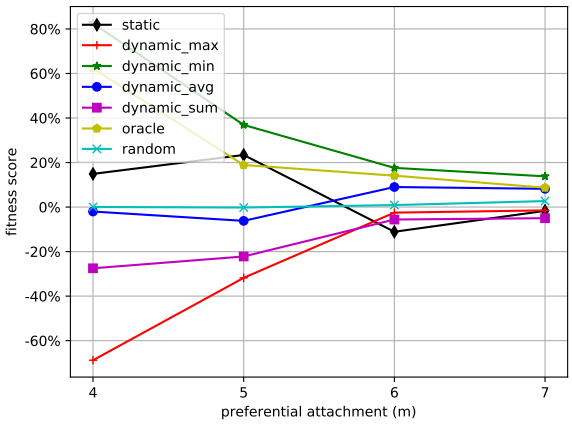


Figure 8: Fitness factor of each strategy for the instances of Tables 2 and 5; *"% of times a heuristic* f *is the best choice" − "% of times a heuristic* f *is the worst choice"*

a scenario and which base relations are least likely to do so. Once this information is obtained, it is used to revise the static weighting scheme accordingly; for example, even though in [19, Figure 9] base relation *equals* appears more restrictive than *overlaps* according to the static weights presented there, *equals* could be considered the least restrictive base relation in QCNs where the majority of entities are equal to one another (because this base relation would appear most in scenarios). Our approach is largely different in that we do not a priori extract scenarios from QCNs, nor to be a priori fix weights for base relations based on the extracted scenarios, but we dynamically, i.e., at runtime, establish weights for the different involved base relations based on local information. A major drawback with the former approach is that randomly trying to extract scenarios from even the smallest of QCNs instances (i.e., without the use of any heuristics at any step of the search process, as suggested in [19]) could still be a non-practical extra effort. In our own experience, a QCN of Interval Algebra of few tens of variables would already pose a significant challenge for that type of random search; we remind the reader that given a QCN $\mathcal{N} = (V, C)$ and a graph $G = (V, E)$, the search space for solving it (extracting a scenario) is $O(|\mathsf{B}|^{|E|})$ in general. Instead, our approach is similar to a counting-based one for CSPs [21], as, much like the work in [21], it is adaptive and it seeks to make informed branching decisions that preserve most of the solutions by determining what proportion of local solutions agree with that decision.

With respect to heuristic selection in the context of QCNs, we are not aware of any published work that contributes towards this direction. However, there is quite some related work in other domains such as planning, traditional constraint programming, Boolean satisfiability (SAT), and Answer Set Programming (ASP) that can be used for inspiration for future work; we detail as follows.

In [41], the authors empirically examine several ways of exploiting and combining the information of multiple heuristics in a *satisficing*[3] best-first search algorithm in the context of planning, and compare their performance in terms of coverage, plan quality, speed, and search guidance. The observant reader will note that our approach here roughly compares with the *sum* approach in [41], in that we select the heuristic with the better sum of assigned constraints per step, averaged over k tries; see again Algorithm 4 and lines 24–25 there in particular. However, in the work of [41] more sophisticated methods are exploited, such as the *alternation* method [42], which avoids aggregating the estimates of each individual heuristic and instead tries to make equal use of all heuristics, essentially alternating between different heuristics. This method was shown to perform better than the *sum* method, and we think that it is worth exploring it for heuristic selection in QCNs. Regarding related work pertaining to traditional constraint programming, SAT, and ASP, we would like to leverage a hierarchical-like approach for heuristic selection (cf. [43]). Such approaches typically make a series of decisions based on various statistical features. Currently, in our case, the count that associates a given constraint with the number of local configurations/models that the constraint participates in could be considered as a statistical feature. However, more features of this type could be exploited for a given constraint, such as the cardinality of its assigned relation, the frequency of repetition of that relation in adjacent constraints, the density of the neighbourhood of the constraint (see [44] for the exact definition of neighbourhood for a QCN), its splitting factor, i.e., whether its assigned relation belongs to a tractable set of relations or it should be split into two or more subrelations, its proximity to a Horn formula, and its potential search space *coverage* (e.g., relation $\{>\}$ provides more coverage than $\{=\}$ in qualitative point-based reasoning, as the latter restricts two points to being equal and, hence, focuses on a single point), among others. Certain dynamic features may also be explored and recorded, by running some propagators for a limited amount of time on a given instance. Such features naturally relate to standard CSP and SAT features as described in [43] and standard ASP features as presented in [45], and exploiting them for QCNs can be an important direction for future work.

6. Conclusion and Future Work

We introduced and evaluated *dynamic branching* strategies for solving QCNs via backtracking search, based on the *count* of *local models* (or *local scenarios*) that a possible relation for a given constraint is involved with in a considered QCN. Thus, we addressed a limitation in the state of the art in qualitative constraint-based reasoning, where the selection of a possible relation for a given contraint is dictated a priori by precomputed static weights, without any regard to the particulars of a given network instance of that calculus, such as its structure. Our approach is adaptive and seeks to make branching decisions that

[3]This term is a a portmanteau of *satisfy* and *suffice*.

preserve most of the solutions by determining what proportion of local solutions agree with that decision. An evaluation with a *random* and a *structured* dataset of QCNs of Interval Algebra showed that up to 5 times better performance may be achieved for structured instances, whilst non-negligible gains of around 20% are maintained for random ones. We conjecture that the more structure is present in a QCN, the better our dynamic approach will perform compared to the static one. Finally, we showed that these results can be further and notably improved via a selection protocol algorithm that aims to effectively and efficiently synthesize the involved heuristics (static or dynamic) into an overall better performing *meta-heuristic* in the phase transition.

As this is a new approach, there are many directions for future work. In particular, we aim to improve on the selection protocol that we presented here and is used to choose among different strategies, as the performance of the virtual portfolio *best* in our evaluation, which always makes the *best* choice of a heuristic for a given network instance, revealed that there is still a lot of room for improvement. Regarding this research direction, the works that we referenced in Section 5 will be a source of inspiration for us. Further, more sophisticated dynamic heuristics could be developed by going beyond triples of variables, which currently form the local models, and engaging larger parts of an instance. Finally, we would like to pair this approach with ongoing research on singleton consistencies [46, 47], and implement adaptive reasoners where the level of consistency checking during search would be adjusted according to the count of local models pertaining to a given constraint.

Acknowledgments

Parts of this work appear in [48], which is indexed in the proceedings of the 27$^{\text{th}}$ International Symposium on Temporal Representation and Reasoning (TIME 2020). The major additional contributions to that work that we presented here are (*i*) the definition and implementation of a selection protocol algorithm that effectively and efficiently synthesizes the heuristics (static or dynamic) discussed in [48] into an overall better performing *meta-heuristic* in the phase transition (see Section 4), and (*ii*) the connection of the work in [48] with the literature in a way that establishes novel directions for future work (see Section 5). We would like to thank the reviewers involved for their helpful comments, which enabled us to better our manuscript. Finally, we would like to acknowledge support from the BMBF project *Dependable Intelligent Systems*.

[1] G. Ligozat, Qualitative Spatial and Temporal Reasoning, ISTE, Wiley, 2013.

[2] F. Dylla, J. H. Lee, T. Mossakowski, T. Schneider, A. van Delden, J. van de Ven, D. Wolter, A Survey of Qualitative Spatial and Temporal Calculi: Algebraic and Computational Properties, ACM Comput. Surv. 50 (2017) 7:1–7:39.

[3] F. Dylla, J. O. Wallgrün, Qualitative Spatial Reasoning with Conceptual Neighborhoods for Agent Control, J. Intell. Robotic Syst. 48 (2007) 55–78.

[4] N. Krishnaswamy, S. Friedman, J. Pustejovsky, Combining Deep Learning and Qualitative Spatial Reasoning to Learn Complex Structures from Sparse Examples with Noise, in: AAAI, 2019.

[5] J. Suchan, M. Bhatt, P. A. Walega, C. P. L. Schultz, Visual Explanation by High-Level Abduction: On Answer-Set Programming Driven Reasoning About Moving Objects, in: AAAI, 2018.

[6] J. Suchan, M. Bhatt, S. Varadarajan, Out of Sight But Not Out of Mind: An Answer Set Programming Based Online Abduction Framework for Visual Sensemaking in Autonomous Driving, in: IJCAI, 2019.

[7] J. Suchan, M. Bhatt, Semantic Question-Answering with Video and Eye-Tracking Data: AI Foundations for Human Visual Perception Driven Cognitive Film Studies, in: IJCAI, 2016.

[8] Z. Cui, A. G. Cohn, D. A. Randell, Qualitative Simulation Based on a Logical Formalism of Space and Time, in: AAAI, 1992.

[9] E. Muñoz-Velasco, M. Pelegrín-García, P. Sala, G. Sciavicco, I. E. Stan, On coarser interval temporal logics, Artif. Intell. 266 (2019) 1–26.

[10] D. Bresolin, D. D. Monica, A. Montanari, P. Sala, G. Sciavicco, Decidability and complexity of the fragments of the modal logic of Allen's relations over the rationals, Inf. Comput. 266 (2019) 97–125.

[11] A. Morales, I. Navarrete, G. Sciavicco, A new modal logic for reasoning about space: spatial propositional neighborhood logic, Ann. Math. Artif. Intell. 51 (2007) 1–25.

[12] R. Kontchakov, A. Kurucz, F. Wolter, M. Zakharyaschev, Spatial Logic + Temporal Logic = ?, in: Handbook of Spatial Logics, 2007, pp. 497–564.

[13] D. Gabelaia, R. Kontchakov, Á. Kurucz, F. Wolter, M. Zakharyaschev, Combining Spatial and Temporal Logics: Expressiveness vs. Complexity, J. Artif. Intell. Res. (2005) 167–243.

[14] J. Gamper, W. Nejdl, Abstract temporal diagnosis in medical domains, Artificial Intelligence in Medicine 10 (1997) 209–234.

[15] D. de Leng, F. Heintz, Qualitative Spatio-Temporal Stream Reasoning with Unobservable Intertemporal Spatial Relations Using Landmarks, in: AAAI, 2016.

[16] F. Heintz, D. de Leng, Spatio-Temporal Stream Reasoning with Incomplete Spatial Information, in: ECAI, 2014.

[17] F. Dylla, T. Mossakowski, T. Schneider, D. Wolter, Algebraic Properties of Qualitative Spatio-temporal Calculi, in: COSIT, 2013.

[18] J. F. Allen, Maintaining Knowledge about Temporal Intervals, Commun. ACM 26 (1983) 832–843.

[19] P. van Beek, D. W. Manchak, The design and experimental analysis of algorithms for temporal reasoning, J. Artif. Intell. Res. 4 (1996) 1–18.

[20] Z. Gantner, M. Westphal, S. Wölfl, GQR-A Fast Reasoner for Binary Qualitative Constraint Calculi, in: AAAI Workshop on Spatial & Temporal Reasoning, 2008.

[21] G. Pesant, C. Quimper, A. Zanarini, Counting-Based Search: Branching Heuristics for Constraint Satisfaction Problems, J. Artif. Intell. Res. 43 (2012) 173–210.

[22] M. Sioutis, Z. Long, T. Janhunen, On Robustness in Qualitative Constraint Networks, in: IJCAI, 2020, to appear.

[23] G. Ligozat, J. Renz, What Is a Qualitative Calculus? A General Framework, in: PRICAI, 2004.

[24] D. A. Randell, Z. Cui, A. Cohn, A Spatial Logic Based on Regions and Connection, in: KR, 1992.

[25] A. Chmeiss, J.-F. Condotta, Consistency of Triangulated Temporal Qualitative Constraint Networks, in: ICTAI, 2011.

[26] J. Renz, G. Ligozat, Weak Composition for Qualitative Spatial and Temporal Reasoning, in: CP, 2005.

[27] J. Renz, B. Nebel, Qualitative Spatial Reasoning Using Constraint Calculi, in: Handbook of Spatial Logics, 2007, pp. 161–215.

[28] J. Renz, Qualitative Spatial and Temporal Reasoning: Efficient Algorithms for Everyone, in: IJCAI, 2007.

[29] J. Huang, Compactness and its implications for qualitative spatial and temporal reasoning, in: KR, 2012.

[30] B. Nebel, H. Bürckert, Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra, J. ACM 42 (1995) 43–66.

[31] J. Renz, B. Nebel, Efficient Methods for Qualitative Spatial Reasoning, J. Artif. Intell. Res. 15 (2001) 289–318.

[32] R. Dechter, Constraint processing, Elsevier Morgan Kaufmann, 2003.

[33] M. Sioutis, Y. Salhi, J. Condotta, Studying the use and effect of graph decomposition in qualitative spatial and temporal reasoning, Knowl. Eng. Rev. 32.

[34] B. Nebel, Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class, Constraints 1 (1997) 175–190.

[35] M. Sioutis, J. Condotta, M. Koubarakis, An Efficient Approach for Tackling Large Real World Qualitative Spatial Networks, Int. J. Artif. Intell. Tools 25 (2016) 1–33.

[36] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science (New York, N.Y.) 286 (1999) 509–512.

[37] R. E. Tarjan, M. Yannakakis, Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs, SIAM J. Comput. 13 (1984) 566–579.

[38] J. Huang, J. J. Li, J. Renz, Decomposition and tractability in qualitative spatial and temporal reasoning, Artif. Intell. 195 (2013) 140–164.

[39] J. J. Li, J. Huang, J. Renz, A divide-and-conquer approach for solving interval algebra networks, in: IJCAI, 2009.

[40] A. Gerevini, Incremental qualitative temporal reasoning: Algorithms for the Point Algebra and the ORD-Horn class, Artif. Intell. 166 (2005) 37–80.

[41] G. Röger, M. Helmert, The More, the Merrier: Combining Heuristic Estimators for Satisficing Planning, in: ICAPS, 2010.

[42] M. Helmert, The Fast Downward Planning System, J. Artif. Intell. Res. 26.

[43] B. Hurley, L. Kotthoff, Y. Malitsky, B. O'Sullivan, Proteus: A Hierarchical Portfolio of Solvers and Transformations, in: CPAIOR, 2014.

[44] M. Sioutis, A. Paparrizou, T. Janhunen, On the Utility of Neighbourhood Singleton-Style Consistencies for Qualitative Constraint-Based Spatial and Temporal Reasoning, in: TIME, 2019.

[45] H. H. Hoos, M. Lindauer, T. Schaub, claspfolio 2: Advances in Algorithm Selection for Answer Set Programming, Theory Pract. Log. Program. 14 (2014) 569–585.

[46] M. Sioutis, A. Paparrizou, J. Condotta, Collective singleton-based consistency for qualitative constraint networks: Theory and practice, Theor. Comput. Sci. 797 (2019) 17–41.

[47] M. Sioutis, Just-In-Time Constraint-Based Inference for Qualitative Spatial and Temporal Reasoning, Künstliche Intell. 34 (2020) 259–270.

[48] M. Sioutis, D. Wolter, Dynamic Branching in Qualitative Constraint Networks via Counting Local Models, in: TIME, 2020.

Appendix A. Evaluation Figures

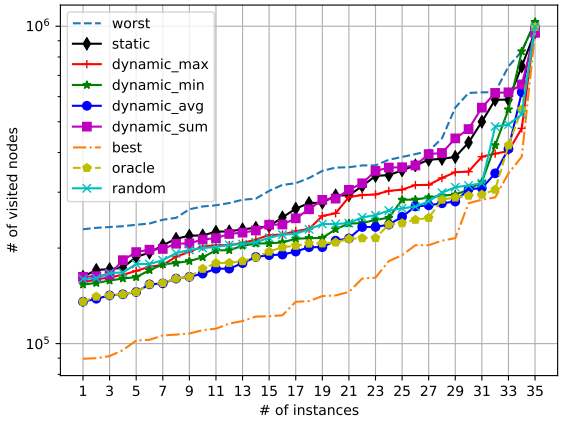


Figure A.9: Insight into the 0.5^{th} percentile of most difficult instances of Table 4 (and Table 1 by extention) for each strategy

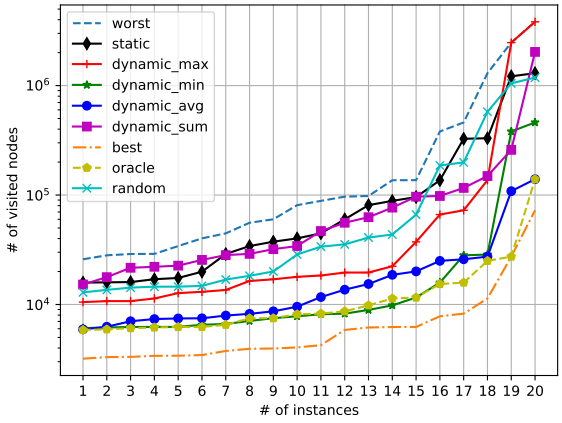


Figure A.10: Insight into the 0.5^{th} percentile of most difficult instances of Table 5 (and Table 2 by extention) for each strategy