Dynamic Branching in Qualitative Constraint Networks via Counting Local Models

Michael Sioutis
Faculty of Information Systems and Applied Computer Sciences, University of Bamberg, Germany
https://msioutis.gitlab.io/
michail.sioutis@uni-bamberg.de

Diedrich Wolter
Faculty of Information Systems and Applied Computer Sciences, University of Bamberg, Germany
https://www.uni-bamberg.de/en/sme/team/diedrich-wolter/
diedrich.wolter@uni-bamberg.de

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 to tackle 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.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Computing methodologies → Temporal reasoning; Computing methodologies → Spatial and physical reasoning

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


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 [18, 8]. 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 [10], deep learning [17], visual explanation [37]

1 Corresponding author.
and sensemaking [36], semantic question-answering [35], qualitative simulation [5], modal logic [21, 3, 20, 16, 11], temporal diagnosis [12], and stream reasoning [6, 14].

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, $x = 0 \land y = 1$ 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 $\mathsf{NP}$-complete [9]. In the sequel, we will be using Interval Algebra (IA) [1] 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 (as 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 [24], 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 [32], where it was observed that a scenario of a QCN may often be constructed collectively by relations that appear in many scenarios individually, i.e., a scenario of a QCN may often be constructed by selecting the most popular relation for each constraint. Finally, 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.

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 approach, discuss some dynamic heuristics.
M. Sioutis and D. Wolter

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* [19], 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 \( \text{Id} \), 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 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 \( \circ \) [19]. For all \( r \in 2^B \), we have that \( r^{-1} = \bigcup\{b^{-1} \mid b \in r\} \).

The weak composition (\( \circ \)) of two base relations \( b, b' \in B \) is defined as the smallest (i.e., strongest) relation \( r \in 2^B \) that includes \( b \circ b' \), or, formally, \( b \circ b' = \{b'' \in B \mid b'' \cap (b \circ b') \neq \emptyset\} \), where \( b \circ b' = \{(x, y) \in D \times D \mid \exists z \in D \text{ such that } (x, z) \in b \land (z, y) \in b'\} \) is the (true) composition of \( b \) and \( b' \). For all \( r, r' \in 2^B \), we have that \( r \circ r' = \bigcup\{b \circ b' \mid b \in r, b' \in r'\} \).

As an illustration, consider the well-known qualitative temporal constraint language of Interval Algebra (IA), introduced by Allen [1]. IA considers time intervals (as temporal entities) and the set of base relations \( B = \{\text{eq}, p, \overline{p}, m, \overline{m}, o, \overline{o}, s, \overline{s}, d, \overline{d}, f, \overline{f}\} \) 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

\[
\begin{align*}
\text{precedes} & \quad p \quad \overline{p} \\
\text{meets} & \quad m \quad \overline{m} \\
\text{overlaps} & \quad o \quad \overline{o} \\
\text{starts} & \quad s \quad \overline{s} \\
\text{during} & \quad d \quad \overline{d} \\
\text{finishes} & \quad f \quad \overline{f} \\
\text{equals} & \quad \text{eq}
\end{align*}
\]

\ch{Figure 2} The base relations of IA; \( \cdot \) denotes the converse of \( \cdot \).
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 \([1]\) and RCC8 \([25]\), are in fact relation algebras \([9]\).

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 \(D\);
- \(C\) is a mapping \(C : V \times V \rightarrow 2^B\) such that \(C(v, v) = \{Id\}\) for all \(v \in V\) and \(C(v, v') = (C(v', v))^{-1}\) for all \(v, v' \in V\), where \(\bigcup B = D \times D\).

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 \(N = (V, C)\) be a QCN, then:

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

Given a QCN \(N = (V, C)\) and \(v, v' \in V\), we introduce the following operation that substitutes \(C(v, v')\) with a relation \(r \in 2^B\) to produce a new, modified, QCN: \(N_{[v, v']/r}\) with \(r \in 2^B\) yields the QCN \(N' = (V, C')\), 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 \(\hat{G}\)-consistency \([4]\) (cf \([27]\)), 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 1.** Given a QCN \(N = (V, C)\) and a graph \(G = (V, E)\), \(N\) is said to be \(\hat{G}\)-consistent iff \(\forall \{v_i, v_j\}, \{v_i, v_k\}, \{v_k, v_j\} \in E\) we have that \(C(v_i, v_j) \subseteq C(v_i, v_k) \circ C(v_k, v_j)\).
We note here that if $G$ is complete, $\Diamond$-consistency becomes identical to $\sigma$-consistency [27], and hence, $\sigma$-consistency is a special case of $\Diamond$-consistency. In the sequel, given a QCN $\mathcal{N} = (V, C)$ of some calculus and a graph $G = (V, E)$, we assume that $\Diamond(G)$ is computable. This assumption holds for most widely-adopted qualitative calculi [9].

3 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 [29]. In particular, for most widely-adopted qualitative calculi [9], such split sets are either known or readily available [26], 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 [29, Section 2]. One of the most essential and widely-used such local consistencies is $\Diamond$-consistency, where $G$ is either the complete graph on the variables of $\mathcal{N}$ [27], or a triangulation (chordal completion) of the constraint graph of $\mathcal{N}$ [4].

As an illustration, the subset $H_{IA}$ of the set of relations of Interval Algebra [23] is tractable for $\Diamond$-consistency, i.e., $\Diamond$-consistency is complete for deciding the satisfiability of any QCN defined over $H_{IA}$ with respect to a triangulation $G$ of its constraint graph [4]. That subset contains exactly those relations that are transformed to propositional Horn formulas when using the propositional encoding of Interval Algebra [23]. 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 $H_{IA}$ 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 $H_{IA}$; each of those subrelations belongs to $H_{IA}$.

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 should be prioritized [39, 28]; in the context of finite-domain CSPs, this strategy is known as the least-constraining value heuristic [7].

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 [39, 28]. In detail, a weight for a base relation is obtained by successively composing it with every possible relation and calculating the sum of the cardinalities of the results, 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 $4 + 2 = 6$ [39, Figure 9].

Two major problems with the aforementioned static strategy is that it assumes a uniform use of relations in QCNs (since 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 $\mathcal{N}|_{V'}$, 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 3 (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 $S = (V', C')$ of $\mathcal{N}|_{V'}$, 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. In that sense, our approach can be seen as being similar to a counting-based one for CSPs [24], 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 choosing 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 $f \in \{\text{max}, \text{min}, \text{avg}, \text{sum}\}$, 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 *aggregate* supportive base relation respectively. At this point, let us revisit the QCN of Interval Algebra in Figure 4, where the relation $\{mi, di, p, m, d, s\}$ is associated with the constraint $C(x_3, x_4)$ is split into subrelations $\{mi\}, \{di, si\}, \{p, m\}, \{d, s\}$ with respect to

<table>
<thead>
<tr>
<th>$b$</th>
<th>$#l$-models</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>6</td>
</tr>
<tr>
<td>$d$</td>
<td>6</td>
</tr>
<tr>
<td>$di$</td>
<td>7</td>
</tr>
<tr>
<td>$m$</td>
<td>4</td>
</tr>
<tr>
<td>$mi$</td>
<td>5</td>
</tr>
<tr>
<td>$s$</td>
<td>3</td>
</tr>
<tr>
<td>$si$</td>
<td>3</td>
</tr>
</tbody>
</table>

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 $\mathcal{H}_{IA}$ [23] is $\{\{mi\}, \{di, si\}, \{p, m\}, \{d, s\}\}$; for this QCN, heuristics **dynamic_avg** and **dynamic_sum** would prioritize relation $\{p, m\}$, and heuristics **static** [39], **dynamic_max**, and **dynamic_min** would prioritize relations $\{d, s\}, \{di, si\}, \text{and} \{mi\}$ respectively.
Algorithm 1 Refinement($\mathcal{N}$, $G$, $A$, $f$)

\begin{algorithm}
\begin{algorithmic}
\State $\mathcal{N} \leftarrow \mathcal{QCN}_G(\mathcal{N})$;
\If {$\emptyset \in \mathcal{QCN}_G(\mathcal{N})$} \State \Return $\bot^V$; \EndIf
\If {$\forall \{v,v'\} \in E, C(v,v') \in A$} \State \Return $\mathcal{N}$; \EndIf
\State $(v,v') \leftarrow \{v,v'\} \in E$ such that $C(v,v') \not\in A$;
\ForEach {$r \in \text{dynamicSelection}(\mathcal{N}, G, A, (v,v'), f)$} \State \text{result} $\leftarrow \text{Refinement}(\mathcal{N}[v,v']/r, G, A, f)$; \EndFor
\If {result $\neq \bot^V$} \State \Return result; \EndIf
\State \Return $\bot^V$;
\end{algorithmic}
\end{algorithm}

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

\begin{algorithm}
\begin{algorithmic}
\State $\mathcal{N} \leftarrow \mathcal{QCN}_G(\mathcal{N})$;
\State \text{counter} $\leftarrow \text{hashTable}()$;
\ForEach {$r \in \{r_1,r_2,\ldots,r_n \in A \mid \{r_1,r_2,\ldots,r_n\} \text{ is a partition of } C(v,v')\}$} \State \text{counter}[r] $\leftarrow f(\{\text{localModels}(b, \mathcal{N}, G, (v,v')) \mid b \in r\}$; \EndFor
\While {$\text{counter}$ is not empty} \State $r \leftarrow \text{counter}$ key paired with the maximum value; \State remove $r$ from $\text{counter}$; \State \Return $r$;
\end{algorithmic}
\end{algorithm}

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

\begin{algorithm}
\begin{algorithmic}
\State $\mathcal{N} \leftarrow \mathcal{QCN}_G(\mathcal{N})$;
\ForEach {$u \in N_G((v,v') \cap N_G(v') \cup N_G(v))$} \State \text{foreach} $(b',b'') \in C(u,v) \times C(u,v')$ \Do \If {$(b',b'') \in b' \circ b''$} \State \text{count} $\leftarrow \text{count} + 1$; \EndIf
\EndFor
\State \Return \text{count};
\end{algorithmic}
\end{algorithm}

subset $\mathcal{H}_A$. 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 [39, Figure 9].
Dynamic Branching in Qualitative Constraint Networks via Counting Local Models

For reference, a variation of the state-of-the-art backtracking algorithm for solving a QCN is provided in Algorithm 1, the main difference to the one appearing in the literature [29, 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 [34] 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 obtaining the count of models for each such base relation, we implement the chosen strategy by applying the respective function among \{max, min, avg, sum\} on the results. Now, each subrelation is associated with a number, a dynamic weight, and 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., |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 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. In practice, there was no noticeable slowdown for the dataset that we consider in this paper (see Section 4), which is not surprising, as the search space for solving a QCN is \(O(|B|^{|E|})\) in general.

4 Evaluation

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.

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 interpreter under version 5.10.0, which implements Python 2.7.13. Only one CPU core was used per run.

Dataset. We generated 7000 random instances of Interval Algebra using model \(H(n = 40, d)\) [22], 1000 for each constraint graph degree value \(d \in \{9, 10, 11, 12, 13, 14, 15\}\) specifically, and 4000 structured instances of Interval Algebra using model BA\((n = 80, m, 3CNF)\) [31], 1000 for each constraint graph preferential attachment [2] value \(m \in \{4, 5, 6, 7\}\) specifically. In both of the aforementioned generation models, constraints were picked from the set of relations expressible in 3CNF when transformed into first-order formulae [22], in order to increase the branching factor in the search tree as much as possible. Finally, regarding the graphs that were given as input to our algorithms, the maximum cardinality search algorithm [38] was used to obtain triangulations of the constraint graphs of our QCNs.
<table>
<thead>
<tr>
<th>d</th>
<th>best</th>
<th>static</th>
<th>dynamic_max</th>
<th>dynamic_min</th>
<th>dynamic_avg</th>
<th>dynamic_sum</th>
<th>worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>62.0±61.8</td>
<td>78.0±79.4</td>
<td>107.0±112.0</td>
<td>66.0±65.9</td>
<td>78.0±79.5</td>
<td>94.0±95.6</td>
<td>114.0±120.4</td>
</tr>
<tr>
<td></td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
</tr>
<tr>
<td>10</td>
<td>52.0±52.6</td>
<td>71.0±125.6</td>
<td>85.0±101.1</td>
<td>60.0±89.9</td>
<td>67.0±104.4</td>
<td>81.0±129.6</td>
<td>108.0±227.7</td>
</tr>
<tr>
<td></td>
<td>0.0±0.0</td>
<td>0.0±0.1</td>
<td>0.0±0.0</td>
<td>0.0±0.0</td>
<td>0.0±0.1</td>
<td>0.0±0.1</td>
<td>0.0±0.1</td>
</tr>
<tr>
<td>11</td>
<td>59.0±79.3</td>
<td>174.0±254.1</td>
<td>143.0±114.3</td>
<td>141.0±1.7</td>
<td>133.0±12.1</td>
<td>181.0±268.4</td>
<td>682.0±508.6</td>
</tr>
<tr>
<td></td>
<td>0.0±0.8</td>
<td>0.1±0.4</td>
<td>0.1±0.1</td>
<td>0.1±0.7</td>
<td>0.1±0.7</td>
<td>0.1±0.7</td>
<td>0.1±0.7</td>
</tr>
<tr>
<td>12</td>
<td>93.1±103.4</td>
<td>4.4±4.6</td>
<td>3.5±14.8</td>
<td>2.9±10.6</td>
<td>4.7±24.6</td>
<td>10.0±14.8</td>
<td>13.3±12.0</td>
</tr>
<tr>
<td></td>
<td>0.7±7.4</td>
<td>3.1±10.8</td>
<td>2.3±13.5</td>
<td>2.2±13.3</td>
<td>3.5±17.4</td>
<td>4.0±17.8</td>
<td>4.0±17.8</td>
</tr>
<tr>
<td>13</td>
<td>2.0±7.6</td>
<td>3.4±11.9</td>
<td>3.2±11.2</td>
<td>2.7±10.8</td>
<td>4.4±11.7</td>
<td>4.6±15.7</td>
<td>30.2±18.4</td>
</tr>
<tr>
<td></td>
<td>1.0±5.9</td>
<td>2.6±3.3</td>
<td>2.3±3.2</td>
<td>2.1±3.3</td>
<td>2.9±3.2</td>
<td>3.3±2.3</td>
<td>4.1±2.3</td>
</tr>
<tr>
<td>14</td>
<td>460.0±1.4</td>
<td>7.2±0.1</td>
<td>7.0±0.5</td>
<td>7.1±0.5</td>
<td>7.9±0.5</td>
<td>9.8±0.5</td>
<td>34.0±0.5</td>
</tr>
<tr>
<td></td>
<td>0.4±1.1</td>
<td>0.5±1.6</td>
<td>0.5±1.6</td>
<td>0.5±1.6</td>
<td>0.6±1.6</td>
<td>0.6±1.6</td>
<td>0.6±1.6</td>
</tr>
<tr>
<td>15</td>
<td>113.0±20.7</td>
<td>1.7±0.2</td>
<td>1.6±0.1</td>
<td>1.5±0.1</td>
<td>1.7±0.1</td>
<td>1.7±0.1</td>
<td>251.5±46.3</td>
</tr>
<tr>
<td></td>
<td>0.1±0.2</td>
<td>0.1±0.3</td>
<td>0.1±0.3</td>
<td>0.1±0.3</td>
<td>0.1±0.3</td>
<td>0.1±0.3</td>
<td>0.2±0.4</td>
</tr>
</tbody>
</table>
Results. Regarding the generation model $H(n = 40, d)$, the main results are presented in Table 1. The dynamic strategies of $\text{dynamic}_\text{min}$ and $\text{dynamic}_\text{avg}$ are up to $20\%$ faster on average than the $\text{static}$ one in the phase transition of the tested instances. Specifically, the phase transition covers mostly the case where $d = 12$, and a little less the case where $d = 13$. With respect to the rest of the dynamic heuristics, viz., $\text{dynamic}_\text{max}$ and $\text{dynamic}_\text{sum}$, the results suggest that $\text{dynamic}_\text{max}$ outperforms $\text{static}$ by a small margin on average in the phase transition, whilst $\text{dynamic}_\text{sum}$ almost mimics the performance of $\text{static}$, if not arguably being a little worse than $\text{static}$ 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 aggregated, 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 $\text{dynamic}_\text{min}$ and $\text{dynamic}_\text{avg}$ among all strategies becomes even more so clear, and the marginal performance gains of $\text{dynamic}_\text{max}$, and $\text{dynamic}_\text{sum}$ at times with respect to $\text{static}$ 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 $\text{best}$ and $\text{worst}$; as a reminder these always make the $\text{best}$ and $\text{worst}$ choice of a heuristic respectively for a given network instance. The performance of portfolio $\text{best}$ 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 terms of defining new dynamic strategies and in terms of devising selection protocols that choose among already existing strategies.

Regarding the generation model $\text{BA}(n = 80, m, 3\text{CNF})$, the results are presented in Table 2 and Figure 6. Here, $\text{dynamic}_\text{min}$ and $\text{dynamic}_\text{avg}$ are up to 3 and 5 times faster on

---

\(^3\)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.
Table 2 Evaluation with structured IA networks that were generated using model BA(n = 80, m, 3CNF) [31]:

<table>
<thead>
<tr>
<th>m</th>
<th>best</th>
<th>static</th>
<th>dynamic_max</th>
<th>dynamic_min</th>
<th>dynamic_avg</th>
<th>dynamic_sum</th>
<th>worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>144.0/144.1</td>
<td>177.0</td>
<td>213.0/216.0/332.0</td>
<td>146.0/146.3/20.0</td>
<td>178.0/177.0/231.0</td>
<td>198.0/198.9/221.0</td>
<td>219.0/222.3/332.0</td>
</tr>
<tr>
<td>5</td>
<td>113.0/145.5</td>
<td>550.0</td>
<td>128.0/154.4/1.0k</td>
<td>155.0/173.3/1.6k</td>
<td>141.0/159.8/1.2k</td>
<td>151.0/177.8/1.2k</td>
<td>178.0/226.3/2.9k</td>
</tr>
<tr>
<td>6</td>
<td>121.0/452.3</td>
<td>372.5k</td>
<td>235.0/47.3k/1.3m</td>
<td>201.0/7.4k/3.8m</td>
<td>172.0/1.4k/469.4k</td>
<td>182.5/933.4/1.6k</td>
<td>223.0/4.0k/2.0m</td>
</tr>
<tr>
<td>7</td>
<td>34.0/24.3</td>
<td>10.4k</td>
<td>51.0/178.9/10.4k</td>
<td>50.0/168.7/7.9k</td>
<td>47.0/10.8/3.4k</td>
<td>49.0/129.4/5.0k</td>
<td>51.0/206.6/22.6k</td>
</tr>
</tbody>
</table>

| TIME 2020 | M. Soutis and D. Walter | XX:11 |
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"

average respectively than the static one in the phase transition of the tested instances, which appears for $m = 6$. The rest of the results are qualitative similar to the previous dataset.

Since the runtime distribution is heavy-tailed for both datasets, the interested reader may want to look into the 0.5th percentile of most difficult instances pertaining to Tables 1 and 2 for each strategy, depicted in Figures 7 and 8 respectively in Appendix A.

5 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 constraint 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 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.

As this is a new approach, there are many directions for future work. In particular, we aim to devise selection protocols that 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. 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 [33, 30], 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.
References


A Evaluation Figures

Figure 7 Insight into the 0.5\textsuperscript{th} percentile of most difficult instances of Table 1 for each strategy

Figure 8 Insight into the 0.5\textsuperscript{th} percentile of most difficult instances of Table 2 for each strategy