PyRCC8
A Efficient Qualitative Spatial Reasoner

Michael Sioutis

Department of Informatics and Telecommunications
National and Kapodistrian University of Athens

December 24, 2011

Table of Contents

About PyRCC8..

- PyRCC8 is an efficient qualitative spatial reasoner written in pure Python. It employs PyPy[1], a fast, compliant implementation of the Python language (2.7.1).

- Qualitative spatial reasoning is based on qualitative abstractions of spatial aspects of the common-sense background knowledge, on which our human perspective on the physical reality is based.

[1] http://pypy.org/

Reasons for Qualitative Spatial Reasoning

- Two main reasons why non-precise, qualitative spatial information may be useful:
  1. Only partial information may be available (e.g. we may know that one region is *disconnected* from another without knowing the precise geometry of the regions)
  2. General constraints holding among geographical objects are often most naturally stated in qualitative terms (e.g. we may wish to state that one region is *part of* another region)

Applications of Qualitative Spatial Reasoning

- Qualitive spatial reasoning is an important subproblem in many *applications*, such as:
  - Natural language understanding
  - Document interpretation
  - Geographical information systems

Region Connection Calculus

- The Region Connection Calculus (RCC) is a first-order language for representation of and reasoning about topological relationships between extended spatial regions
- RCC abstractly describes regions, that are non-empty regural subsets of some topological space which do not have to be internally connected
- Relationships between spatial regions can be defined based on the $C(a, b)$ *connected* relation, which is true if the topological closures of the regions a and b share a common point

Region Connection Calculus RCC8

- The Region Connection Calculus RCC8 is a *constraint* language formed by the combination of the following eight jointly exhaustive and pairwise disjoint *base relations*:
  - disconnected (DC)
  - externally connected (EC)
  - equal (EQ)
  - partially overlapping (PO)
  - tangential proper part (TPP)
  - tangential proper part inverse (TPPi)
  - non-tangential proper part (NTPP)
  - non-tangential proper part inverse (NTPPi)

The eight basic relations of the RCC8 calculus

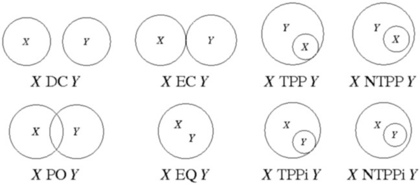


Figure: Two dimesional examples for the eight base relations of RCC8

From these basic relations, combinations can be built. For example, proper part (PP) is the union of TPP and NTPP.

The RCC8 composition table

PyRCC8
A Efficient Qualitative Spatial Reasoner

Michael Sioutis

Outline

Introduction

The RCC8 Calculus

Path Consistency

Consistency

Experimental results

Conclusions

Future Work

Bibliography

∘	DC	EC	PO	TPP	NTPP	TPP$^{\smile}$	NTPP$^{\smile}$	EQ
DC	*	DC,EC PO,TPP NTPP	DC,EC PO,TPP NTPP	DC,EC PO,TPP NTPP	DC,EC PO,TPP NTPP	DC	DC	DC
EC	DC,EC PO,TPP$^{\smile}$ NTPP$^{\smile}$	DC,EC PO,TPP TPP$^{\smile}$,EQ	DC,EC PO,TPP NTPP	EC,PO TPP NTPP	PO TPP NTPP	DC,EC	DC	EC
PO	DC,EC PO,TPP$^{\smile}$ NTPP$^{\smile}$	DC,EC PO,TPP$^{\smile}$ NTPP$^{\smile}$	*	PO TPP NTPP	PO TPP NTPP	DC,EC PO, TPP$^{\smile}$ NTPP$^{\smile}$	DC,EC PO,TPP$^{\smile}$ NTPP$^{\smile}$	PO
TPP	DC	DC,EC	DC,EC PO,TPP NTPP	TPP NTPP	NTPP	DC,EC PO,TPP TPP$^{\smile}$,EQ	DC,EC PO,TPP$^{\smile}$ NTPP$^{\smile}$	TPP
NTPP	DC	DC	DC,EC PO,TPP NTPP	NTPP	NTPP	DC,EC PO,TPP NTPP	*	NTPP
TPP$^{\smile}$	DC,EC PO,TPP$^{\smile}$ NTPP$^{\smile}$	EC,PO TPP$^{\smile}$ NTPP$^{\smile}$	PO TPP$^{\smile}$ NTPP$^{\smile}$	PO,EQ TPP TPP$^{\smile}$	PO TPP NTPP	TPP$^{\smile}$ NTPP$^{\smile}$	NTPP$^{\smile}$	TPP$^{\smile}$
NTPP$^{\smile}$	DC,EC PO,TPP$^{\smile}$ NTPP$^{\smile}$	PO TPP$^{\smile}$ NTPP$^{\smile}$	PO TPP$^{\smile}$ NTPP$^{\smile}$	PO TPP$^{\smile}$ NTPP$^{\smile}$	PO,TPP$^{\smile}$ TPP,NTPP NTPP$^{\smile}$,EQ	NTPP$^{\smile}$	NTPP$^{\smile}$	NTPP$^{\smile}$
EQ	DC	EC	PO	TPP	NTPP	TPP$^{\smile}$	NTPP$^{\smile}$	EQ

Figure: Composition table for RCC8 relations

RCC8 example

- Two houses are connected via a road. Each house is located on an own property. The first house possibly touches the boundary of the property; the second one surely does not. What can we infer about the relation of the properties to the road?

```
house1 DC house2
house1 {TPP, NTPP} property1
house1 {DC, EC} property2
house1 EC road
house2 { DC, EC } property1
house2 NTPP property2
house2 EC road
property1 { DC, EC } property2
road { DC, EC, TPP, TPPi, PO, EQ, NTPP, NTPPi } property1
road { DC, EC, TPP, TPPi, PO, EQ, NTPP, NTPPi } property2
```

- Using a *path consistency* algorithm, we can refine the network in the following way:

```
road { PO, EC } property1
road { PO, TPP } property2
```

RCC8 example

- Two houses are connected via a road. Each house is located on an own property. The first house possibly touches the boundary of the property; the second one surely does not. What can we infer about the relation of the properties to the road?

```
house1 DC house2
house1 {TPP, NTPP} property1
house1 {DC, EC} property2
house1 EC road
house2 { DC, EC } property1
house2 NTPP property2
house2 EC road
property1 { DC, EC } property2
road { DC, EC, TPP, TPPi, PO, EQ, NTPP, NTPPi } property1
road { DC, EC, TPP, TPPi, PO, EQ, NTPP, NTPPi } property2
```

- Using a *path consistency* algorithm, we can refine the network in the following way:

```
road { PO, EC } property1
road { PO, TPP } property2
```

The RSAT reasoning problem

- RSAT in the RCC8 framework, is the reasoning problem of deciding consistency of a set of spatial formula Θ, i.e., whether there is a spatial configuration where the relations between the regions can be described by Θ.
- RSAT is NP-Complete!
- However, tractable subsets S of RCC8 exist for which the consistency problem can be decided in polynomial time

The RSAT reasoning problem

- RSAT in the RCC8 framework, is the reasoning problem of deciding consistency of a set of spatial formula Θ, i.e., whether there is a spatial configuration where the relations between the regions can be described by Θ.
- RSAT is NP-Complete!
- However, tractable subsets S of RCC8 exist for which the consistency problem can be decided in polynomial time

The RSAT reasoning problem

- RSAT in the RCC8 framework, is the reasoning problem of deciding consistency of a set of spatial formula Θ, i.e., whether there is a spatial configuration where the relations between the regions can be described by Θ.
- RSAT is NP-Complete!
- However, tractable subsets S of RCC8 exist for which the consistency problem can be decided in polynomial time

Maximal tractable subsets

- First, we define as NP_8 the set of relations that by themselves result in NP-completeness when combined with the set of base relations:

$$NP_8 = \begin{array}{c} \{\, \mathrm{R} \mid (\{\mathrm{PO}\} \not\subseteq R \text{ and } (\{\mathrm{NTPP}\} \subseteq R \text{ or } \{\mathrm{TPP}\} \subseteq R) \\ \text{and } (\{\mathrm{NTPPi}\} \subseteq R \text{ or } \{\mathrm{TPPi}\} \subseteq R)\} \\ \cup \{\{\mathrm{EC, NTPP, EQ}\}, \{\mathrm{DC, EC, NTPP, EQ,}\}, \\ \{\mathrm{EC, NTPPi, EQ}\}, \{\mathrm{DC, EC, NTPPi, EQ,}\}\} \end{array}$$

- The following subsets are maximal tractable subsets that contain all base relations:

$$\hat{H}_8 = \begin{array}{c} (\mathrm{RCC8} \setminus NP_8) \setminus \{\, \mathrm{R} \mid (\{\mathrm{EQ, NTPP}\} \subseteq R \text{ and } \{\mathrm{TPP}\} \not\subseteq R) \\ \text{or } (\{\mathrm{EQ, NTPPi}\} \subseteq R \text{ and } \{\mathrm{TPP}\} \not\subseteq R)\} \end{array}$$

$$C_8 = \begin{array}{c} (\mathrm{RCC8} \setminus NP_8) \setminus \{\, \mathrm{R} \mid (\{\mathrm{EC}\} \subset R \text{ and } \{\mathrm{PO}\} \not\subseteq R) \text{ and} \\ R \cap \{\mathrm{TPP, NTPP, TPPi, NTPPi, EQ}\} \neq \emptyset\} \end{array}$$

$$Q_8 = \begin{array}{c} (\mathrm{RCC8} \setminus NP_8) \setminus \{\, \mathrm{R} \mid (\{\mathrm{EQ}\} \subset R \text{ and } \{\mathrm{PO}\} \not\subseteq R) \text{ and} \\ R \cap \{\mathrm{TPP, NTPP, TPPi, NTPPi}\} \neq \emptyset\} \end{array}$$

Maximal tractable subsets

- First, we define as NP_8 the set of relations that by themselves result in NP-completeness when combined with the set of base relations:

$$NP_8 = \begin{array}{c} \{ \text{R} \mid (\{\text{PO}\} \not\subseteq R \text{ and } (\{\text{NTPP}\} \subseteq R \text{ or } \{\text{TPP}\} \subseteq R) \\ \text{and } (\{\text{NTPPi}\} \subseteq R \text{ or } \{\text{TPPi}\} \subseteq R)\} \\ \cup \{\{\text{EC, NTPP, EQ}\}, \{\text{DC, EC, NTPP, EQ, }\}, \\ \{\text{EC, NTPPi, EQ}\}, \{\text{DC, EC, NTPPi, EQ, }\}\} \end{array}$$

- The following subsets are maximal tractable subsets that contain all base relations:

$$\hat{H}_8 = \begin{array}{c} (\text{RCC8} \setminus NP_8) \setminus \{ \text{R} \mid (\{\text{EQ, NTPP}\} \subseteq R \text{ and } \{\text{TPP}\} \not\subseteq R) \\ \text{or } (\{\text{EQ, NTPPi}\} \subseteq R \text{ and } \{\text{TPP}\} \not\subseteq R)\} \end{array}$$

$$C_8 = \begin{array}{c} (\text{RCC8} \setminus NP_8) \setminus \{ \text{R} \mid (\{\text{EC}\} \subset R \text{ and } \{\text{PO}\} \not\subseteq R) \text{ and} \\ R \cap \{\text{TPP, NTPP, TPPi, NTPPi, EQ}\} \neq \emptyset\} \end{array}$$

$$Q_8 = \begin{array}{c} (\text{RCC8} \setminus NP_8) \setminus \{ \text{R} \mid (\{\text{EQ}\} \subset R \text{ and } \{\text{PO}\} \not\subseteq R) \text{ and} \\ R \cap \{\text{TPP, NTPP, TPPi, NTPPi}\} \neq \emptyset\} \end{array}$$

Path Concistency

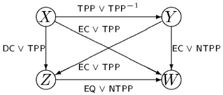
- Approximates consistency and realises *forward checking* in a backtracking algorithm
- Checks the consistency of all triples of relations and eliminates relations that are impossible though iteravely performing the operation

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

  untill a fixed point $\overline{M}$ is reached.
  If $M_{ij} = \emptyset$ for a pair (i, j) then M is inconsistent, otherwise $\overline{M}$ is *path-consistent*.
- Computing $\overline{M}$ is done in $O(n^3)$

Is path consistency sufficient?

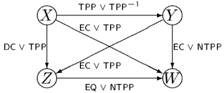
- Path consistency does **not imply** consistency
- The following set of spatial constraints is path-consistent but not consistent



- Still, path consistency is **sufficient** for deciding consistency, if only relations in any tractable subset that contains all base relations, like $\hat{H}_8$, C_8, Q_8, are used

Is path consistency sufficient?

- Path consistency does not imply consistency
- The following set of spatial constraints is path-consistent but not consistent



- Still, path consistency is **sufficient** for deciding consistency, if only relations in any tractable subset that contains all base relations, like $\hat{H}_8$, C_8, Q_8, are used

Algorithm

Path-Consistency(C)
Input: A constraint network C
Output: A refined constraint network C', True or False

$Q \leftarrow \{(i, j) \mid 1 \leq i < j \leq n\}$ // Initialize the queue
while Q is not empty **do**
select and delete an (i, j) from Q
for $k \leftarrow 1$ **to** n, $k = i$ and $k = j$ **do**
$t \leftarrow C_{ik} \cap (C_{ij} \circ C_{jk})$
if $t \neq C_{ik}$ **then**
if $t = \emptyset$ **then**
return False
$C_{ik} \leftarrow t$
$C_{ki} \leftarrow \breve{t}$
$Q \leftarrow Q \cup \{(i, k)\}$
$t \leftarrow C_{kj} \cap (C_{ki} \circ C_{ij})$
if $t \neq C_{kj}$ **then**
if $t = \emptyset$ **then**
return False
$C_{kj} \leftarrow t$
$C_{jk} \leftarrow \breve{t}$
$Q \leftarrow Q \cup \{(k, j)\}$
return True

Implementations

- Based on Simple Queue
  - Implentation with non-weighted arcs
- Based on Priority Queue (process most restrictive arc first)
  - Implentation with exactly weighted arcs
  - Implentation with approximately weighted arcs, using the approach by Van Beek and Manchak [1]

Comparison of Implementations

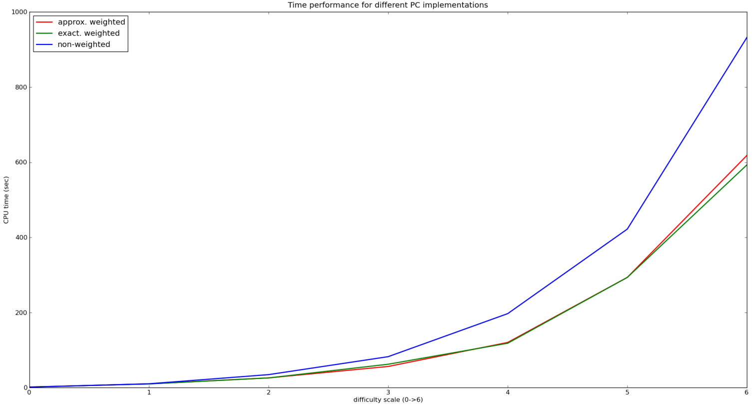


Figure: Heuristic performance based on CPU time

Queue Structure

- Simple Queue (combines set and deque):
  - Membership checking: $O(1)$
  - Push: $O(1)$
  - Pop: $O(1)$
- Priority Queue (combines dictionary and heapq):
  - Membership checking: $O(1)$
  - Push: $O(log(n))$
  - Pop: $O(log(n))$

Comparing PC implementations of different QSRs

- We compare the PC implementation of PyRCC8 to the PC implementations of the following spatial qualitative reasoners:
  - Renz's solver[2]
  - GQR[3]
- Different size n of instances from $A(n, d = 9.5, l = 4.0)$ were used

[2] http://users.rsise.anu.edu.au/%7Ejrenz/software/rcc8-csp-solving.tar.gz

[3] http://sfbtr8.informatik.uni-freiburg.de/R4LogoSpace/Tools/gqr.html

Comparison diagram

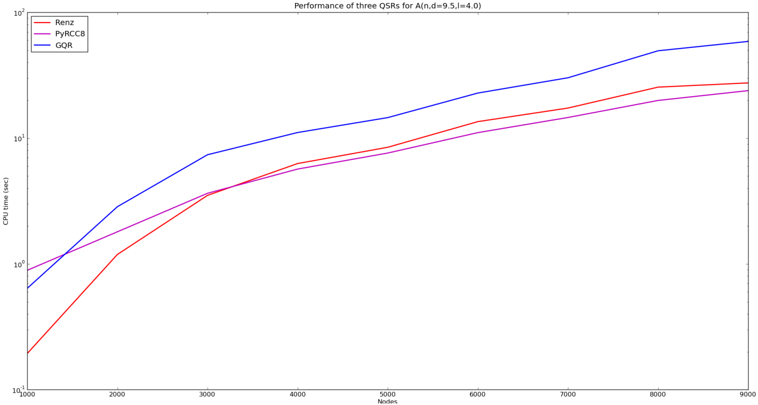


Figure: Comparison of different PC algorithms - best configuration used

Using the AdminGeo ontology to compare PC implementations of different QSRs

- We compare the PC implementation of PyRCC8 to the PC implementations of the following spatial qualitative reasoners:
  - Renz's solver
  - GQR
  - Pellet Spatial[4]
- The AdminGeo[5] dataset (11761 regions / 77910 relations) was used which was properly translated to fit the input format of the different PC implementations

[4] http://clarkparsia.com/pellet/spatial/

[5] http://data.ordnancesurvey.co.uk/ontology/admingeo/

Evaluation with a large dataset

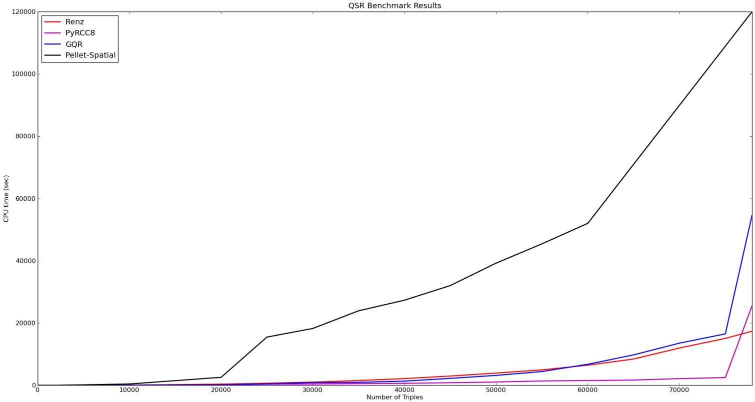


Figure: Comparison with use of the AdminGeo dataset (11761 regions / 77910 relations)

PyRCC8 A Efficient Qualitative Spatial Reasoner

Michael Sioutis

Consistency

- To explore the search space in order to solve an instance Θ of RSAT, some sort of backtracking must be used.
- We implemented two backtracking algorithms:
  1. A strictly recursive one
  2. An equivalent iterative one which resembles tail-recursion

Recursive Algorithm

Consistency(C)
Input: A constraint network C
Output: A refined constraint network C' if C is satisfiable or None

```
if not Path-Consistency(C) then
    return None
if no constraint can be split then
    return C
else
    choose an unprocessed constraint x_i R x_j and split R into S_1, ..., S_k ∈ S: S_1 ∪ ... ∪ S_k = R
    Values ← {S_l | 1 ≤ l ≤ k}
for V in Values do
    replace x_i R x_j with x_i V x_j in C
    result = Consistency(C)
    if result ≠ None then
        return result
return None
```

Iterative Algorithm

Consistency(C)
Input: A constraint network C
Output: A refined constraint network C' if C is satisfiable or None

```
Stack ← {} // Initialize stack
if not Path-Consistency(C) then
  return None
while 1 do
    if no constraint can be split then
      return C
    else
      choose an unprocessed constraint x_i R x_j and split R into S_1, ..., S_k ∈ S: S_1 ∪ ... ∪ S_k = R
      Values ← {S_l | 1 ≤ l ≤ k}
    while 1 do
        if not Values then
          while Stack do
              Values = Stack.pop()
              if Values then
                break
          else
              return None
        V = Values.pop()
        replace x_i R x_j with x_i V x_j in C
        if Path-Consistency(C) then
          break
    Stack.push(Values)
raise RuntimeError, Can't happen
```

Heuristics

- Split set specific
  - Base relations set: Average branching factor of 4.0
  - Horn relations set ($\hat{H}_8$): Average branching factor of 1.4375
- Constraint specific
  - Static/Dynamic: constraint processing is done *statically* before or *dynamically* during the search
  - Local/Global: constraint evaluation based on *local* heuristic weight or *global* heuristic criterion
- Value specific
  - Choice of a sub-relation based on its contrainedness

Comparing PyRCC8 to other QSRs

- We compare PyRCC8 to the following spatial qualitative reasoners:
  - Renz's solver[6]
  - GQR[7]
- Different size n of instances from $A(n, d = 9.5, l = 4.0)$ were used

---

[6] http://users.rsise.anu.edu.au/%7Ejrenz/software/rcc8-csp-solving.tar.gz

[7] http://sfbtr8.informatik.uni-freiburg.de/R4LogoSpace/Tools/gqr.html

Comparison diagram

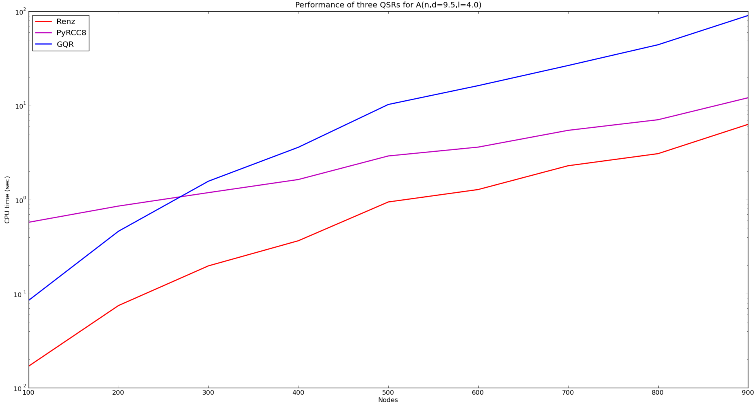


Figure: Comparison of different QSRs - best configuration used

Evaluation of different heuristics with hard instances

- Hard instances are composed from relations of NP_8. We used the QSR benchmark dataset of QSTRLib[8]
- For all hard instances we ran PyRCC8 using the Horn relations set as our split set, and using the static/global and dynamic/local configuration, since dynamic/global and static/local proved to be insufficient, confirming the results in [2]
- Black line sets the # of visited nodes under which Renz's RCC8 reasoner was unable to solve instances for any configuration

[8] `http://qstrlib.org/Benchmarks/`

Results using hard of H instances (1/2)

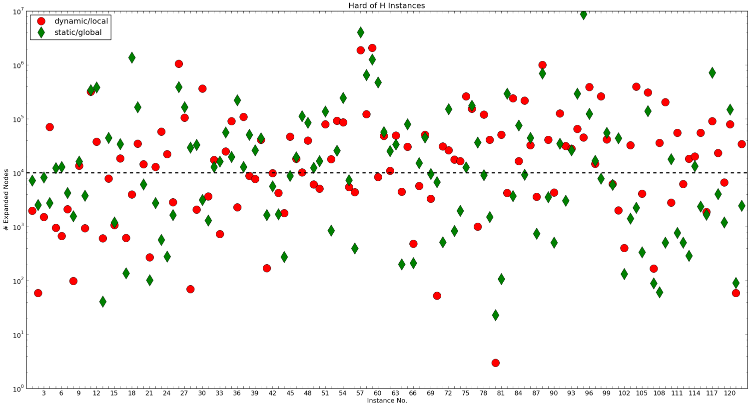


Figure: Comparison based on expanded nodes

Results using hard of H instances (2/2)

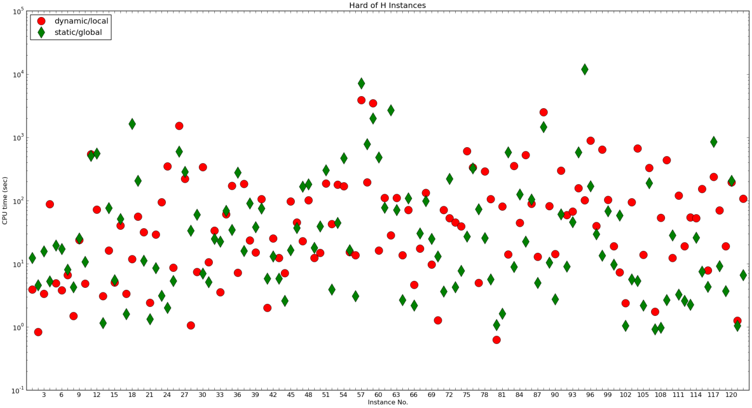


Figure: Comparison based on CPU time

Results using hardest of H instances

	expanded nodes	CPU time
dynamic/local	1404812	2868.427s
static/global	11038473	15174.947s

Table: Comparison based on the average of three hardest of H instances

Conclusions

- PyRCC8 outperforms GQR and Renz's reasoner in terms of path-consistency checking, as shown in Figure 4, and outperforms GQR and converges to the performance of Renz's reasoner in terms of consistency checking, as shown in Figure 6
- Allthough not demonstrated, maximal tractable subsets make a huge difference over the use of base relations, considering the branching factor of the two approaches
- dynamic/local $\geq$ static/global $>$ static/local $\gg$ dynamic/global

Future Work

- Explore intelligent tie-breakers regarding variable and value selection
- Create module to generate spatial CSPs
- Transform PyRCC8 to a generic qualatitave reasoner
- Enhance PyRCC8 with spatial reasoning involving landmarks [3]
- Make use of Python's dynamic features to the fullest

References

[Van Beek and Manchak, 1996]
The design and experimental analysis of algorithms for temporal reasoning.
Journal of Artificial Intelligence Research, 4, 1-18.

[Nebel and Renz, 2001]
Efficient Methods for Qualitative Spatial Reasoning
Journal Of Artificial Intelligence Research, Volume 15, pages 289-318, 2001

[Weiming Liu, Shengsheng Wang, Sanjiang Li, and Dayou Liu, 2011]
Solving qualitative constraints involving landmarks
CP'11 Proceedings of the 17th international conference on Principles and practice of constraint programming

The End

Any Questions? ☻