IEEE Symposium Series Workshop on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP), Singapore, pp. 1-7, 2013.

Efficient Determination of Spatial Relations Using Composition Tables and Decision Trees

Nathan Eloe, Jennifer Leopold, Chaman Sabharwal, Douglas McGeehan
Computer Science Department
Missouri University of Science and Technology
Rolla, MO - 65409, USA
{nwe5g8, leopoldj, chaman, djmvfb}@mst.edu

***Abstract*—In order for Qualitative Spatial Reasoning applications to be both useful and usable, the information feedback loop between the computational engine and the user must be as seamless as possible. Inherently, computational geometry can be quite expensive, and every effort must be made to avoid inefficient or unnecessary calculations. Within the field of Region Connection Calculi, the 9-Intersection model often is used to determine the spatial relation between two regions. Consequently, optimization efforts typically focus on calculations involving the intersections between the interiors, boundaries, and exteriors of the regions, or the use of composition tables to narrow down the possibilities for the relations that can hold between two regions. The few implementations of spatial reasoners that have been attempted have been simply proofs-of-concept and/or have been limited to two dimensions. Herein we present a novel approach that combines the use of composition tables and decision trees to efficiently determine the spatial relation between two objects in 3D considering both connectivity and obscuration. This approach has been fully implemented for the VRCC-3D+ spatial reasoning system, and benchmarks are included to corroborate our claims of efficiency.**

***Index Terms*—Region Connection Calculus, 9-Intersection, ID3, Decision Trees, Composition Tables, VRCC-3D+**

I. Introduction

Qualitative Spatial Reasoning (QSR) has the potential to further enhance the functionality of applications for diverse fields such as Geographic Information Systems (GIS), visual programming language semantics, and digital image analysis. Unfortunately, the practicality of automated spatial reasoning may be diminished if expensive and unnecessary calculations are not avoided; every delay in the information feedback loop inhibits not only real-time user interaction, but also knowledge discovery.

The foundation of many QSR systems are Region Connection Calculi (RCC), which use a 9-Intersection model to distinguish spatial relations based on the intersections of the interiors, boundaries, and exteriors of two or more regions. It has been shown that the 9-Intersection model can be reduced to a 4-Intersection problem [1] through the use of a decision tree. Although examining fewer intersections provides a tremendous savings in computational effort, the amount of time to determine the spatial relation between two regions still can be unacceptable in real time if additional optimizations are not employed.

There are a variety of different RCC models, each with different degrees of expressivity. Unfortunately, few actual implementations of RCC systems exist. Those that do exist are incomplete (e.g., Albath's original RCC-3D system [2]), still in the alpha or proof-of-concept stage [3], [4], or limited to two dimensions [3]. Many libraries in existence that allow computation of the 9-Intersection model, and thus computation of the RCC-8 relations, also share these limitations [5] or require special representations of data [6], [7], [8]. Not only has the lack of full implementations limited the actual use of these models, it also has concealed many of their computational challenges.

Herein we present a novel approach for efficiently calculating spatial relations. It utilizes decision trees and composition tables to avoid as many unnecessary computations as possible. This approach has been implemented for a QSR system (VRCC-3D+ [4]) that determines the spatial relation between two objects in 3D considering both connectivity and obscuration. We also provide benchmarks to show the viability of our approach.

II. Background and Related Work

A. RCC-8

Introduced in 1992 by Randall, Cohn, and Cui, RCC was initially an acronym for the names of the authors; later Region Connection Calculus was deemed a more appropriate name for what was being modeled. RCC-8 was the earliest system to define relationships between regions based on connection [9]. Randall, Cohn, and Cui define eight relationships (see Figure 1):

- Disconnected (DC)
- Externally Connected (EC)
- Partial Overlap (PO)
- Equal (EQ)
- Tangential Proper Part (TPP)
- Tangential Proper Part Converse (TPPc)
- Non-Tangential Proper Part (NTPP)
- Non-Tangential Proper Part Converse (NTPPc)

These relationships are Jointly Exhaustive Pairwise Disjoint (JEPD), meaning that there is no configuration of physically feasible regions that cannot be described by one of the relations, and no pair of regions can be described by more than one relationship (they are not ambiguous).

B. VRCC-3D+

VRCC-3D+ [4] is a GUI for and an extension of the RCC-3D [2] system that was designed to maximize both computational feasibility and the comprehensiveness of resulting information. It defines relations in three dimensions using a composite relation of the form R_O(A, B) for two regions A and B. The R part of the VRCC-3D+ relation (herein referred to as the base relation) is one of the eight RCC-8 relations, although computed in three dimensions, not two dimensions.

The O part of the VRCC-3D+ relation represents obscuration as determined for a particular two dimensional projection plane; there are fifteen possible obscuration relations, characterized by intersections for the interiors and exteriors of two regions, and a qualitative depth parameter (see Table I). An obscuration relation has a base type of either No Obscuration (nObs), Partial Obscuration (pObs), Complete Obscuration (cObs), or Equal Obscuration (eObs). Additional qualifiers, including converse obscuration (_c) or equal depth from camera (_e) are added to further enhance the expressive power of the obscuration relation. Every obscuration relation is conversely related to exactly one other obscuration. Not every obscuration relation is applicable to every base relation; in all, there are 46 VRCC-3D+ relations (as shown in Table II). See [4] for a more detailed discussion of the VRCC-3D+ model.

TABLE I
Characterization Table for Obscuration Relations [10]: Int = interior, Ext = exterior; F = No (empty intersection), T = True (Non-empty) intersection. InFront = Y means A is closer to the viewing plane than B. N means B is closer than A. E means mutual obscuration or A and B are equidistant to the viewing plane

	IntInt	IntExt	ExtInt	InFront
nObs	F	F	F	Y
nObs_c	F	F	F	N
nObs_e	F	F	F	E
pObs1	T	F	T	Y
pObs2	T	T	T	Y
pObs_c1	T	T	F	N
pObs_c2	T	T	T	N
pObs_e	T	T	T	E
eObs	T	F	F	Y
eObs_c	T	F	F	N
eObs_e	T	F	F	E
cObs	T	T	F	Y
cObs_c	T	F	T	N
cObs_e1	T	T	F	E
cObs_e2	T	F	T	E

C. Decision Trees

The field of research in Top-Down Induction of Decision Trees produced many algorithms for the creation of such trees. Iterative Dichotomiser 3 (ID3) [11] creates a simple and minimal decision tree from a sample of classified training data. Traversing the tree leads to a leaf node that provides a

TABLE II
The Possible Obscurations for RCC-8 Relationships [10]

	DC	EC	PO	EQ	TPP	TPPc	NTPP	NTPPc
nObs	*	*						
nObs_c	*	*						
nObs_e	*	*						
pObs1	**	*						
pObs2	**	*						
pObs_c1	*	*	*					
pObs_c2	*	*	*					
pObs_e			*					
eObs	*	*	*			*		
eObs_c	*	*	*		*			
eObs_e			*	*	*			
cObs	*	*	*			*		*
cObs_c	*	*	*		*		*	
cObs_e1			*		*	*		
cObs_e2			*		*	*		

classification for a given entity based on the values of certain attributes of the entity. The construction of the tree requires that each non-leaf node makes a decision based on the attribute that results in the greatest gain of information. ID3 produces a tree by using the informational entropy of the training set; this is the amount that making a decision decreases the entropy in the sub groups generated by splitting the training set based on the value of the attribute.

Decision trees have been used previously to classify spatial relations. Sabharwal et al. [1] used decision trees to reduce the 9-Intersection characterization of spatial relations to 4-Intersections. A decision tree for classifying a relation in RCC-8 is shown in Figure 2. This decision tree allows for identification of an RCC-8 relation in as few as two truth predicates, or at most four truth predicates. Clementini et al. [12] also proposed decision trees as the standard way of classifying relations, and gave multiple decision trees for the eight RCC-8 relations, based on different ways of calculating the probability of a specific intersection being empty.

While having a decision tree might increase the ease and efficiency of spatial relation determination, manual creation of the tree might be difficult and error prone, particularly for some of the RCC models that have many relations (e.g., RCC-23 [13] has 23 relations, VRCC-3D+ [14] has 46 relations, and RCC-62 [15] has 62 relations). A decision tree also exposes another problem: the predicates are not all equal in their execution time or complexity. For example, the first decision node in the tree proposed by Sabharwal et al. [1] is IntInt. This is because IntInt distinguishes DC and EC from all of the other RCC-8 relations (i.e. PO, EQ, TPP, NTPP, TPPc, and NTPPc). However, in the current implementation of VRCC-3D+, any intersection calculation that does not involve the boundary (i.e., IntInt) is avoided. This decision is influenced by the fact that our datasets are Wavefront OBJ formatted

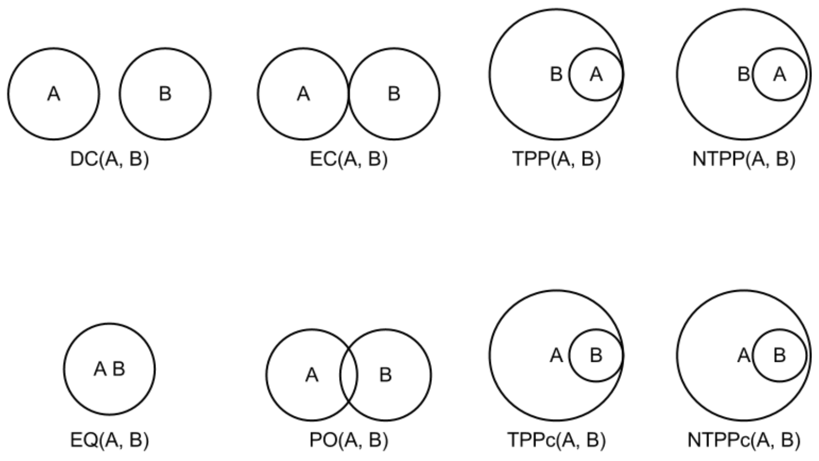

Fig. 1. Examples of the eight JEPD RCC-8 relations.

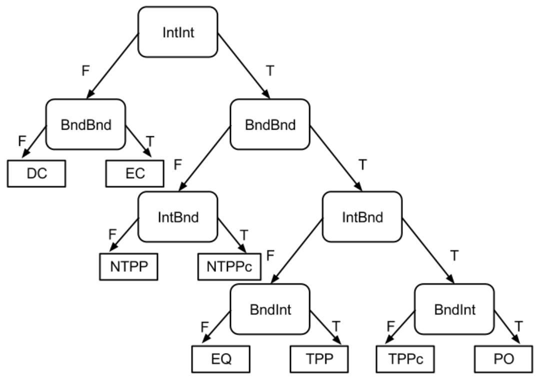

Fig. 2. The hand generated decision tree [1] hierarchy of the intersection predicates for the RCC-8 relations.

files that only provide boundary information. Other factors such as whether a GPU (e.g. OpenCL, CUDA, or Stream) is being utilized could influence the decision to prioritize the computation of certain predicates over others in a particular implementation.

Our goal is to develop a method by which the nodes in the decision tree can be dynamically (rather than manually) generated based on the relations in the composition table. Recall that given objects (or regions) A, B, and C, and knowledge of relations R(A,B) and R(B,C), the composition table entry R(A,B) $^\circ$ R(B,C) is the set of all possible relations for R(A,C). We want to avoid complete pre-computation of the decision trees for the following reasons:

- A general mechanism for deciding the most appropriate decision predicate can be adapted for use with other RCC systems (e.g., RCC-23 and RCC-62), as well as for relations that do not simply use binary terms in the predicates (e.g., in VRCC-3D+ the obscuration relations utilize a ternary term, InFront).
- Intersection predicates are not necessarily equal in terms of computational efficiency. A sequential execution core (standard CPU) may have an efficient algorithm for implementing the intersection of interiors. However, a massively parallel computation platform (GPGPU through CUDA, OpenCL, or Stream) may be able to perform a naïve pairwise facial intersection test (e.g., for BndBnd) much faster than it could perform a tree implementation for IntInt. A generalized predicate selection algorithm can more easily be modified to utilize whichever predicates execute most efficiently for a particular hardware configuration.
- There are 247 total subsets of the RCC-8 relations of cardinality 2 through 8. However, in the composition table, a relatively small percentage (≈10%) of these subsets actually occur. Precomputation of all 247 (or indeed even the actual subsets) would be an inefficient use of memory and effort.

It should be noted that although this discussion has focused on RCC-8 relations (which represent the base relations in VRCC-3D+), the method presented herein is not limited to RCC-8, and can be extended directly to any RCC system in which the relations can be represented as combinations of binary predicates.

III. Implementation

A. Algorithm

Quinlan's ID3 [11] generates decision trees based on a training set of data, classifying inputs based on attributes. Quinlan also introduced another decision tree algorithm, C4.5 [16], that addresses some of the weaknesses in ID3, including: (1) trees can become excessively large, depending on the number of attributes and classes in the system, and (2) some decision paths may result in unknown decisions if the training set is not complete.

We mitigate these problems by having a very small number of attributes (i.e., the intersection predicates), and having complete knowledge of the determination of the relations with respect to the predicates. As such, we can show that no matter what subset of RCC-8 relations we want to be able to differentiate between, we can come up with a tree of decision nodes of finite size that will always yield a single RCC-8 relation as a terminating node. The additional overhead of C4.5 over ID3, although negligible, is unnecessary for our application. Consequently, we use the attribute selection step of ID3 to allow selection of the most informative predicate to calculate.

Generating the entire decision tree for a subset of RCC-8 relations is inefficient: with unambiguous binary predicates, determining the order of the decision nodes for the unused sub-tree would be wasted computation. Only determining the decision nodes along the path to the bottom of the tree also would not result in a useful representation.

In the following pseudo-code, truePreds is a hash map, containing key-value pairs where the key is an intersection predicate and the value is the set of possible relations that result from the intersection predicate, BndInt, being true (e.g. truePreds[BndInt] = {NTPP, PO, TPP}).

```
predicates = {IntInt, IntBnd, BndInt,
              BndBnd, BndExt, ExtBnd}

def ratio(S, pred):
  intersection = S & truePreds[pred]
  return |intersection| / |S|

def entropy(S, pred):
  if |S| < 2:
    return 0.0
  sum = 0.0
  r = ratio(S,pred)
  //if r is 1 or 0, that means it does
  //not contribute to the entropy
  //and we have already
  //partitioned on that predicate
  if r > 0:
    sum += r * lg(r)
  if 1-r > 0:
    sum += (1-r) * lg(1-r)
  return -sum

def entropy(S):
  if |S| < 2:
    return 0.0
  return lg(|S|)

def gainRatio(S, pred):
  if |S| == 0:
    return 0.0
  branchT = S & truePreds[pred]
  branchF = S - truePreds[pred]
  tEntropy = entropy(branchT)
  fEntropy = entropy(branchF)
  splitH = |branchT| * tEntropy
  splitH += |branchF| * fEntropy
  splitH /= |S|
  if splitH > 0:
    return (entropy(S) / splitH) - 1
  return 0.0
```

We can then rank the predicates by their informational gain ratio. The predicate with the highest gain ratio is the predicate that should be chosen next to calculate.

This algorithm presents another route with which we can shorten the information feedback loop. Optimization of algorithms is still a necessity, but the potential gain from doing so is lost if time is wasted calculating uninteresting or redundant predicates. Again we emphasize that the overall goal is to determine the spatial relation between two objects in space, and to do so in the most efficient way possible. Previous efforts have addressed this problem from two different directions. Composition tables [9] reduce the number of possible relations by using global information about the relations of the objects relative to other objects in the scene. Decision trees, specifically hand generated decision trees, reduce the problem from 9-Intersection predicates to at most 4-Intersection predicates [1] but do not take global information into account. By algorithmically selecting the decision nodes of the decision tree as needed we aim to combine these methods and use both global information (by iteratively using the global information in the composition table) and local information (the use of decision nodes) to generate the relation between two objects.

By using a function (CTFilter(A,B)) that allows us to use global information to reduce the number of possible relations, we can determine the relationship between two objects now as follows:

```
def calcRCC8(A,B):
  possibleRels = CTFilter(A,B)
  while |possibleRels| > 1:
    //predicates sorted on decreasing
    //information gain ratio
    preds = sorted predicates
    p = calculate preds[0]
    possibleRels = {possibleRels | p}
  return possibleRels
```

This algorithm uses the composition table to use global information to reduce the possible number of relations between objects A and B. As long as the number of relationships in the resulting list is greater than one, the predicates are sorted based on decreasing information gain ratio, the first (most useful) predicate is calculated, and the relations are filtered such that possibleRels only contains the relations in which the calculated predicate holds. These filtered relations are used in the next iteration.

Table III shows the gain ratios obtained for five intersection predicates for three different subsets of RCC-8 relations, each subset obtained from the RCC-8 composition table: TPPc $\circ$ TPP = {PO, TPP, TPPc, EQ}, TPP $\circ$ TPPc = {DC, EC, PO, TPP, TPPc, EQ}, and TPPc $\circ$ EC = {EC, PO, TPPc, NTPPc}.

TABLE III
GAIN RATIOS FOR VARYING INPUT SETS

	$S = \{PO, TPP, TPPc, EQ\}$	$S = \{DC, EC, PO, TPP, TPPc, EQ\}$	$S = \{EC, PO, TPPc, NTPPc\}$
BndBnd	0.0	0.3359	0.2619
IntBnd	1.0	0.6309	0.2619
BndInt	1.0	0.5510	0.2619
ExtBnd	1.0	0.5510	1.0
BndExt	1.0	0.5510	0.0

In the first two cases, we see that calculating the BndBnd intersection predicate gives us the least information. However in the first case, that calculation, in fact, would be wasted effort, as it would result in no new information. The third input set shows that not only would calculating BndExt be wasted effort, but ExtBnd gains us the most information, as it has a significantly higher gain ratio than the other predicates. Based on the results for the first example, we would calculate any predicate except BndBnd; BndBnd would give us no new information, and in fact is a scenario where having a modification that allows us to bias the gain ratio for faster algorithms/implementations would be useful. In the second example, the IntBnd predicate would be selected for calculation. The ExtBnd predicate would be chosen for calculation in the third example. This sample input set also shows a scenario in which we have a definitively more useful predicate, but also a predicate that would be wasted effort (BndExt).

B. Complexity Considerations

The complexity of calculating any of the intersection predicates is dependent on many factors, not the least of which is the specific implementation. For example, our implementation of BndBnd uses trees of Axis Aligned Bounding Boxes (AABBs) to narrow down the number of triangular faces between the objects that could possibly intersect before resorting to computing triangle-triangle intersections [17]. Because of this, the worst case complexity for this predicate is $O(f_A * f_B)$, where f_A and f_B are the number of faces in the respective objects A and B. This only happens if the objects are situated such that all Axis Aligned Bounding Boxes in the objects overlap. The best case is if none of the Axis Aligned Bounding Boxes in the tree overlap, in which case we have a constant time comparison that is not dependent on the number of faces.

Hence, because the complexity of these calculations depends on the face counts of the objects and the relative orientation, the cost of calculating the next predicate to use is negligible compared to the cost of calculating the predicate.

IV. Experimental Timing and Results

A. Experimental Setup

We timed the execution of our implementation of three predicates: BndBnd, BndInt, BndExt. It has been shown [18] that implementing these three predicates is sufficient to uniquely determine the RCC-8 relation of two objects because some predicates can be implemented with respect to others; IntBnd(A,B) == BndInt(B,A) and ExtBnd(A,B) == BndExt(B,A). The experiments were run 100 times on each of 44 model files that contained two 3D spheres in various configurations. Each sphere consisted of approximately 2000 faces; the execution time of the predicates depends mainly on the relative configuration of the spheres. File IO and internal representation generation were ignored across all executions of the predicates. Timing was performed on an AMD Bulldozer processor running at 3.1 GHz, with 12 GB of RAM available. Table IV shows the results of the timings.

TABLE IV
TIMING STATISTICS FOR PREDICATE CALCULATIONS.

	BndBnd	BndInt	BndExt
Average (sec)	0.0103	6.66	0.188
Min (sec)	0.00000431	0.000433	0.000459
Max (sec)	0.849	71.8	1.17
StdDev	0.2170442	17.846823	0.261670587

The complexity of the predicate chooser is dependent on the number of relations in the input set. For every execution of the predicate selector, a random subset of the RCC-8 relations with cardinality $1 < k \leq 8$ was chosen out of all possible subsets of RCC-8. The timings reported are the average of 100 executions of the predicate selector.

B. Results

Table IV shows some statistical analysis of the runtimes of the three implemented intersection predicates. Figure 3 shows the runtime of the predicate selection algorithm for varying sized subsets of the RCC-8 relations. Selecting a predicate takes on average 1/100th of the time it takes to calculate the predicate itself. Using this approach avoids unnecessary calculations, and also allows us to bias calculations that are more efficient and precise on a given system. Currently, BndBnd is the most precise and efficient algorithm based on the data representation we use in our implementation. This ability to bias the selection allows us to avoid the most computationally expensive algorithms unless absolutely necessary.

V. Additional Optimizations: Memoization

A. Definition

Memoization is an optimization technique in which a function‘s result for a unique set of inputs is cached. When the function is called with the inputs again, the cached results are reused and the computation is avoided. This approach

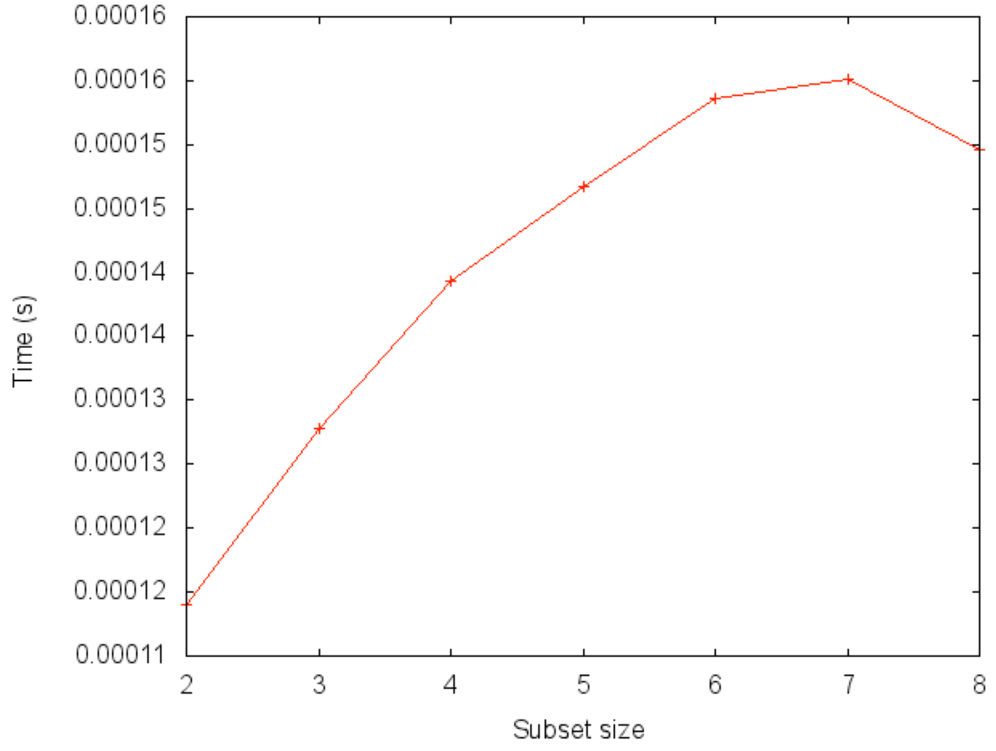

Fig. 3. Predicate Selection runtime vs. input subset size.

has two tradeoffs: memory usage and the overhead associated with checking the cache key. If checking a cache key is computationally negligible compared to the computation, the speedup is significant. Part of the overhead of checking the cache key is associated with the time it takes to access the cache: if the cache is too large to fit in memory, then cache hits become expensive. The tradeoffs between memory and speed must be weighed against each other. In our application, it was decided that the relatively small maximum number of possible cache keys (247) (namely, the number of possible subsets of RCC-8 relations) made the cost of additional memory negligible when compared to the theoretical magnitude of the speedup.

B. Experimental Setup

The effect of memoization on the predicate selection algorithm was tested by executing the algorithm 10^n times, for $0 \preceq n \preceq 7$. Between each set of executions, the cache for the memoizer was cleared so that solutions for keys would have to be regenerated. For each input n, the experiment was run 100 times. The runtime reported for each input is an average over all runs for that parameter value.

C. Results

Figure 4 shows that as the memoizer is allowed to run for more iterations, we reach the overhead of checking a cache key. Using memorization for a small number of iterations is more expensive then no memorization at all due to the overhead of checking the cache. After several iterations, the cost of checking the cache key is negligible compared to choosing the next predicate, resulting in more than a 100x speedup. It is worth noting that these timings were determined as if every subset of RCC-8 was possible. As a small number of these subsets can actually occur in the composition table, the chance of a cache miss will go to zero significantly faster, and as such the runtime will reach the time of checking the cache key in fewer iterations.

VI. Future Work

Shortening the feedback loop is vital to efficient analysis of three dimensional data. Eliminating unnecessary calculations is just one step in achieving this goal. Future efforts will focus on further optimizing the calculations involved in the intersection predicates by investigating more efficient intersection algorithms and exploiting the resources available on modern computers, including general purpose computing on graphics cards and distributed computing.

VII. Conclusions

In order for Qualitative Spatial Reasoning applications to be both feasible and useful, the information feedback loop between the computational engine and the user must be made as efficient as possible. Herein we presented a novel approach that combined the use of composition tables and decision trees to efficiently determine the spatial relation between two objects. Specifically, we showed how to leverage the benefits of calculating the intersection predicate with the highest informational gain. We also showed that the cost of caching the

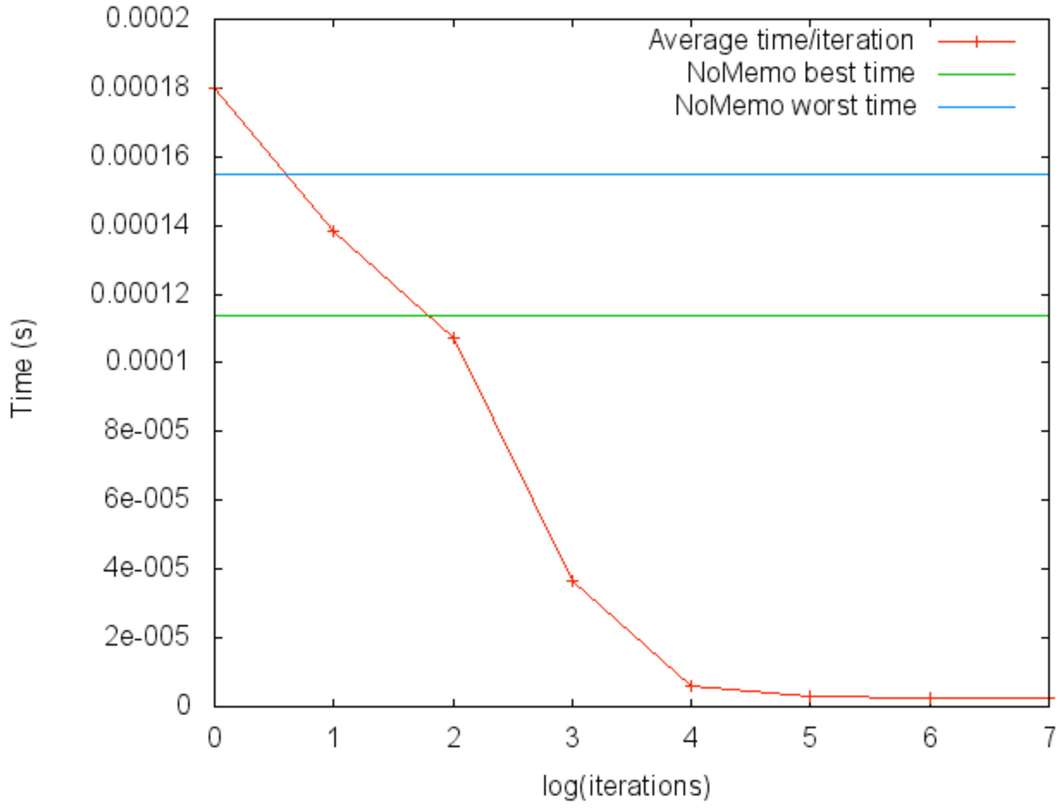

Fig. 4. Memoized Predicate Selection runtime per number of iterations.

decision tree information greatly increases the efficiency of the predicate selection even at the cost of requiring additional memory. As the proliferation of 2D and 3D datasets continue, we hope that this work facilitates the implementation of other spatial reasoning systems in the future.

References

[1] C. Sabharwal and J. L. Leopold, "Reducing 9-Intersection to 4-Intersection for identifying relations in region connection calculus," in *The 24th International Conference on Computer Applications in Industry and Engineering*, 2011, pp. 118–123.

[2] J. Albath, J. L. Leopold, C. Sabharwal, and A. M. Maglia, "RCC-3D: Qualitative spatial reasoning in 3D," in *CAINE*, 2010, pp. 74–79.

[3] M. Stocker and E. Sirin, "Pelletspatial: A hybrid region connection calculus RCC-8 and rdf/owl reasoning and query engine," *Proceedings of Web Ontology Language (OWL): Experiences and Directions 2009 (OWLED 2009)*, 2009.

[4] C. Sabharwal, J. Leopold, and N. Eloe, "A more expressive 3D region connection calculus," in *Proceedings of the 2011 International Workshop on Visual Languages and Computing (in conjunction with the 17th International Conference on Distributed Multimedia Systems (DMS'11))*, 2011, pp. 307–311.

[5] A. Murta, "A general polygon clipping library," *Advanced Interfaces Group, Department of Computer Science, University of Manchester. On-line resource. URL: http://www. cs.-man. ac. uk/aig/staff/alan/software/gpc. html*, 2000.

[6] P. Ramsey *et al.*, "Postgis manual," *Refractions Research Inc*, 2005.

[7] A. Karlsson, "Gis and spatial extensions with mysql," *MySQL Developer Zone, http://dev. mysql. com/tech-resources/articles/4.1/giswith-mysql. html*, 2007.

[8] D. Wong and J. Lee, *Statistical analysis of geographic information with ArcView GIS and ArcGIS*. Wiley, 2005, vol. 1.

[9] D. A. Randell, Z. Cui, and A. Cohn, "A spatial logic based on regions and connection," in *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, B. Nebel, C. Rich, and W. Swartout, Eds. San Mateo, California: Morgan Kaufmann, 1992, pp. 165–176. [Online]. Available: citeseer.ist.psu.edu/randell92spatial.html

[10] C. Sabharwal and J. L. Leopold, "Smooth transition neighborhood graphs for 3D spatial relations," in *2013 IEEE Symposium Series on Computational Intelligence*, 2013.

[11] J. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[12] E. Clementini, J. Sharma, and M. J. Egenhofer, "Modelling topological spatial relations: Strategies for query processing," *Computers and Graphics*, vol. 18, no. 6, pp. 815 – 822, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0097849394900078

[13] A. G. Cohn, B. Bennett, J. Gooday, and N. M. Gotts, "Qualitative spatial representation and reasoning with the region connection calculus." *GeoInformatica*, vol. 1, no. 3, pp. 275–316, 1997.

[14] J. Albath, J. L. Leopold, and C. Sabharwal, "Visualization of spatio-temporal reasoning over 3D images," in *Proceedings of the 2010 International Workshop on Visual Languages and COmputing*, 2010, pp. 277–282.

[15] J. Ouyang, Q. Fu, and D. Liu, "A model for representing topological relations between simple concave regions," in *Proceedings of the 7th international conference on Computational Science, Part I: ICCS 2007*, ser. ICCS '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 160–167.

[16] J. Quinlan, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993, vol. 1.

[17] C. Sabharwal, "Survey of implementations of cross intersection between triangular surfaces," MDC Report Q0909 (Now Boeing at St. Louis, MO-USA), 1987.

[18] N. Eloe, J. L. Leopold, C. Sabharwal, and Z. Yin, "Efficient computation of object boundary intersection and error tolerance in VRCC-3D+," in *Distributed Multimedia Systems '12*, 2012, pp. 67–70.