This space is reserved for the EPiC Series header, do not use it

Evaluation of Axiom Selection Techniques

Qinghua Liu1, Zihao Wang2, Zishi Wu2, and Geoff Sutcliffe2

1 System Credibility Automatic Verification Engineering Lab of Sichuan Province, Southwest Jiaotong University, China, [qhliu@my.swjtu.edu.cn](mailto:qhliu@my.swjtu.edu.cn)

2 University of Miami, USA, zxw526@miami.edu,ry04ert39@miami.edu,geoff@cs.miami.edu

**Abstract**

Evaluation of Axiom Selection Techniques

# Introduction

GEOFF:

Intro about axiom selection. Most evaluation by running ATP, the ”proofs in the pudding”. Takes time, propose Quantitative metrics based on selection. Our methods. Evaluation vs Vampire and E. Section on selection techniques - cuttion (eg Isabelle) and projection (eg SInE).

# Selection Metrics

GEOFF: Description of metrics.

# Our Selection Techniques

GEOFF: Intro

QINGHUA: Qinghua’s distance

## Infinity Cut

1. Qinghua’s infinity cut

## A(nother) Machine Learning Approach

1. Qinghua’s ML?

## Spectral clustering and path search algorithms

1. Zihao

In this chapter, 3 methods of axiom selection based on the graph was proposed. First, using spectral clustering to select about half of the needed axioms. Second, based on the first step, using local search to find path from the conjecture to one of it’s infinity distance axiom. The local search algorithm is a greedy strategy by finding the optimal solution only based on the current state. There are local beam search algorithm is implemented. There are 2 experiments. The first one is finding a random path which each node is visited its nearest neighbor. The process starts from conjecture to one of the axiom that is the infinity distance from the conjecture to them. And second one is find all paths from the conjecture to infinity distance node connected to the conjecture. After step 2 and 3, a spectral cluster algorithm which include more clusters is implemented. Based on the path from the step 2 and 3, each path node’s cluster set node will be added to the path set.

### **Spectral clustering binary classification**

The graph will be divided into 2 clusters by using spectral clustering. Most of the partitions can include all need axioms in one cluster.

Based on the graph weighted by the dissimilarity, a new graph with the same amount of vertices and edges are generated by weighted by similarity by using the Gaussian Radial basis function kernel showed below:

S=e-D

In the formula, S is the similarity matrix, D is the dissimilarity matrix. After generating the similarity matrix, the spectral clustering is implemented by using python sklearn based on the similarity matrix.

The algorithm of spectral clustering of sklearn is illustrated below:

input: similarity

output: an array construct of integers which means each element’s cluser

Step 1: Generate degree matrix D from the similarity matrix

Step 2: Generate Laplace matrix L

Step 3: Calculate normalized Laplace matrix L1 by using Ncut algorithm

Step 4: Calculate L1’s eigenvalue and each eigenvalue’s eigenvector f

Step 5: Use k-means based on the normalized eigenvector

The steps illustrated above are packaged in the sklearn library. After spectral clustering, the majority automated theorem proof (ATP) task can separate in to 2 parts which has 1 part includes all the needed axioms.

### **Local beam search for axioms selection**

The local beam search uses greedy search strategy which is find the local optimal state based on the current state. However, the local beam search not only find one local optimal state, but also keeps track more local optimal states. In this study, the definition of the local optimal solution are the minimum dissimilarity axiom nodes connected to the current node. The local beam search is implemented based on the breadth first search (BFS).

In this study, the BFS only considers the neighbors which has the minimum dissimilarity. The BFS start at the conjecture and finish once reach an infinity dissimilarity node of the conjecture (terminate node). There are 2 experiments. The first is randomly find a minimum dissimilarity node if there are more than 1 local optimal solutions. The second is search all minimum dissimilarity nodes for all local optimal solutions.

#### **Local beam search random greedy path**

In the first experiment, BFS started at the conjecture and stop when reach an infinity dissimilarity node connected with the conjecture. A FIFO queue is implemented for the BFS search. The pseudocode is illustrated below.



19 path.append(u)

In the pseudocode, s is the start vertex. White means the node haven’t be visited yet. Gray means the node is been visiting now. Black means the node’s all neighbors have been visited. The attribute d means the level of the vertex. Lines 1–4 mark every vertex except the start vertex, set u.d to be infinity for each vertex u, and set the parent of every vertex to be NIL. Line 5 paints s gray,. Line 6 initializes s.d to 0, and line 7 sets the predecessor of the source to be NIL. Lines 8–9 initialize Q to the queue containing just the vertex s.

The randomness happened due to the order of enqueue is random. After dequeue the first node u, its child v will recode its parent to be u. After an terminate node is visited. The backtrack process will be implement by recursively search node and its parent. The path is saved in a set without consider the order.

#### **Local beam search all greedy path**

The second experiment is finding all greedy path if one node has more than 1 minimum dissimilarity neighbors. The same BFS process is used. If the dequeue vertex u is a terminate node, then it’s neighbor will not visited and mark u’s color to black and add to the path set. The BFS continue until the queue is empty. The process is shown below:



if u != terminate node:



if u = terminate node:



path.append(u)



path.append(u)

The example of search path is illustrated in the Figure 1.

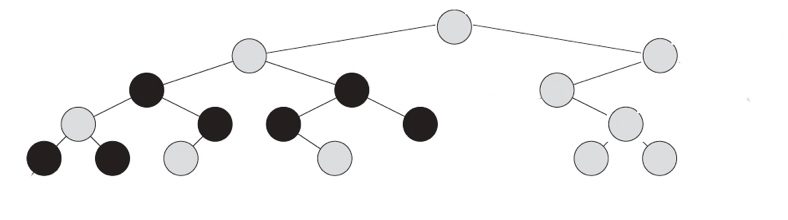


Figure 1: local beam search path

The root represents the conjecture and the black nodes represent the terminate node.

### **A\* search algorithm**

The A\* search algorithm is a kind of heuristic algorithm that find the shortest path from the start node to the goal node. It uses evaluate function f(x) to choose the next step by considering the minimum value of f(x).

The evaluate function f(x) is defined as: f(x)=g(x)+h(x). g(x) is the sum weights of visited edges at the current state and h(x) is the heuristic function which calculate the distance from the next selection to the goal state.

In the axiom selection task. The start state is the conjecture and goal states are a set of all terminate nodes. The task will find the short path from the conjecture to each goal state. The selection set is the union of all shortest path above.

The framework is based on the BFS. The neighbor is selected by using f(x). If there are more minimum f(x) value of the current state, then there are more neighbor are considered to search by using BFS.

### **Deep spectral clustering to improve robustness**

Based on the axiom path set, a deep spectral clustering by using amortized analysis implemented to improve the selectivity of the axiom selection. The amortized idea is that suppose one node is connected with k neighbors which are necessary to prove the conjecture. In this study, k equal to 5 defined by randomness. The spectral clustering’s number of clusters is defined by total number of nodes in subgraph divided by k. For each node in the path, it’s cluster will be found and the neighbors will be added to the path set.

### **K-nearest neighbor algorithm to improve robustness**

The K-nearest neighbor (KNN) algorithm is one of the machine learning algorithms of classification. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors, details illustrated in Figure 2.



Figure 2: example of KNN

In Figure 2, the green point need to be classified into blue or red. In this case, k=3. The green dot finds its closest 3 neighbors. In the 3 neighbors, the majority number of the category is the green dot’s category.

In axiom selection task, the KNN is used to add more axioms to the path of Local beam search all greedy path. For each axiom in the all path set, finding all neighbors and its unique dissimilarity set (UDS). The UDS is defined below:

The UDS is ordered by non-decrease order. There are 3 kinds of UDS classified by the size of less than 3, and greater than 3 or equal to 3. If size(UDS)>=3, then this axiom has the ability to add more axioms because the first element of UDS is 0 which means the dissimilarity of this axiom itself; the second element represents the closest neighbor that are added in the local beam search; the third element represents the second closets neighbor, and so on.

In KNN algorithm, k=5. A subset of UDS (SUDS) is generated by removing the first and the second element. The algorithm is illustrated as follows:

If size(SUDS)>=5:

add 5 closest dissimilarity axioms to the Allpath set  
 else if size(SUDS)>=3:

add 1 closest dissimilarity axioms to the Allpath set

### **Discussion**

#### **Result**

Two datasets (Bushy and Chainy) from the MPTP 2078 is used to evaluate the algorithm. There are both 325 tasks in Bushy and Chainy dataset. There are 5 features to evaluate algorithm: the selection evaluation (All score), selected enough number (Enuf), selection evaluation on selected enough tasks (Enuf score) , selectivity and enough selectivity (Enuf selectivity). The definitions of the 4 features are illustrated below.

The all score is the average accuracy of successful selected tasks.

accuracy= the number of need axioms/ the number of selected axioms

If there are some needed axioms not in the selected axioms, then accuracy is 0.

Enuf is the number of tasks that the accuracy is not 0.

Enuf score is the average accuracy based on Enuf tasks.

The selectivity is the average of the number of selected axioms/total number of axioms in the task. The enough selectivity is the average of the number of selected axioms/ Enuf.

The evaluation result is illustrated in the Table 1.

Table 1 : evaluation result

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bushy | | | | | Chainy | | | | |
| Algorithm | All score | Enuf | Enuf score | selectivity | Enuf selectivity | All score | Enuf | Enuf score | selectivity | Enuf selectivity |
| Spectral clustering | 0.39 | 299 | 0.42 | 0.64 | 0.70 | 0.06 | 325 | 0.06 | 0.94 | 0.94 |
| Local beam search random path | 0.33 | 278 | 0.39 | 0.65 | 0.76 | 0.02 | 157 | 0.04 | 0.30 | 0.63 |
| Local beam search all path | 0.36 | 304 | 0.39 | 0.73 | 0.79 | 0.05 | 294 | 0.05 | 0.84 | 0.93 |
| A\* search all path | 0.13 | 154 | 0.28 | 0.38 | 0.79 | 0.02 | 177 | 0.04 | 0.45 | 0.81 |
| Knn search all path | 0.37 | 315 | 0.38 | 0.77 | 0.80 | 0.05 | 313 | 0.05 | 0.93 | 0.96 |

#### **Analysis**

Evaluation of Axiom Selection Techniques Liu, Wang, Wu, Sutcliffe

## Our Selection Techniques

1. Zishi’s way

# Evaluation Results

Section on evaluation 1. The test set(s)... Should we add tptp based set? 2. The results 3. The conclusions

Data on MPTPTP2078, Number of problems in test set, how selected (proofs, hence already solved, but possibly with axiom selection), numbers of different adequate subsets, average ratio nntp/all

# Conclusion

GEOFF: 1. Future correlate metrics with ptover performance (or do now!)

2