CSC 793 report

Evaluation of Axiom Selection Techniques

Student: Zihao Wang

Supervisor: Geoff Sutcliffe

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

# Abstract

Automated theorem proving (ATP) is concerned with the development and use of systems that automated sound reasoning: the derivation of conclusions that follow inevitably from facts. However, the search complexity for finding proofs of theorems is enormous. To tackle this challenge, this research uses artificial intelligence methods to extract a small subset of axioms sufficient for proving that a given conjecture is a theorem. Machine learning algorithms combined with search strategies are achieved for axiom selection. The experimental results demonstrate that the performance of the local beam search is high than the baseline and it has the best evaluation indices. Therefore, it is a promising algorithm.

*Keywords:* automated theorem proving, axiom selection, artificial intelligence, machine learning, search

# Introduction

“Large Theory” problems in Automated Theorem Proving (ATP) have been defined as having many functors and predicates, and many axioms of which only a few are required for the proof of a theorem. Large theory problems are mostly found in corpora that contain very many problems, e.g., the MPTP2078 corpus [1], the Mizar 40 corpus [2], and the GRUNGE corpus [3]. Large theory problems present challenges to ATP systems, because of the large search space generated by a large number of axioms. One key to solving large theory problems is selecting a subset of the axioms that are adequate for finding proof. There has been significant and successful research on this topic. Many techniques are based on the occurrences of symbols in the formulae, e.g., the SInE method [4] and its derivatives. The fact that large theory problems often occur in large corpora makes the application of machine learning techniques viable[5], as done in the MaLARea system [6].

Evaluation of axiom selection techniques is typically done by:

* 1. Choosing a corpus of large theory problems.
  2. For each problem in the corpus, selecting a subset of the axioms.
  3. Running an ATP system on a reduced problem formed from the selected axioms and the problem’s conjecture.

When such experiments are done on large corpora it is necessary to impose a small time limit when running the ATP system. In the last step of this process is a proof indicates an adequate selection, and a countermodel indicates an inadequate selection. A timeout provides no information - the selection might be inadequate, or the selection might be adequate but the reduced problem is too hard because too many (unnecessary) axioms were selected and the time limit is too small. The results are also influenced by the choice of the ATP system.

This paper presents metrics for evaluating axiom selection techniques without having to run an ATP system on the reduced problems. While the “proof is in the pudding” and it is eventually necessary to evaluate by running an ATP system, the method described in this paper provides a first-pass evaluation that allows axiom selection techniques to be rapidly tested and refined. The approach has the advantage of being independent of a chosen ATP system.

This paper additionally presents some new axiom selection techniques. The new techniques the axiom selection built into the Vampire [7] and E [8] ATP systems, are evaluated using the proposed metrics.

This paper is structured as follows: Section 2describes the evaluation metrics. Section 3describes a distance measure between formulae and comes up with artificial intelligence axiom selection techniques based on that measure. Section 4provides evaluation results. Section 5illustrates theconclusion.

# Selection Metrics

The principle behind the new evaluation metrics is to compare the selected subsets of axioms with known adequate sets of axioms. In this work, the MPTP2078 corpus has been used. The MPTP2078 corpus has two versions of each of the 2078 problems: the Bushy(small) versions contain only the Mizar axioms that are known to be directly used in the proof of the conjecture, and the Chainyversions contain all the axioms that precede the conjecture in the Mizar library order. The bushy problems contain between N and M axioms, while the Chainy versions contain between N and M axioms.

To extract known adequate sets of axioms for each problem, Vampire and E were run on the problems with a 300s CPU limit. This produced proofs for 325 of the bushy problems (260 by Vampire and 325 by E) and 325 of the Chainy problems (304 by Vampire and 325 by E). For the problems solved, the axioms used in proofs were extracted as known adequate sets of axioms, and new problems formed from those adequate sets with the corresponding conjectures.

Additionally, in testing the new axiom selection techniques some further different adequate sets were found, and further new problems created. This resulted in new problems for 1551 of the bushy problems and 705 of the Chainy problems. The Pruney problems provide adequate known sets of axioms against which selected sets of axioms can be compared.

# Selection Techniques

The artificial intelligence axiom selection techniques rely on the fully connected graph that marks the distance between each formula. Section 3.1 gives an introduction of the distance measurement and section 3.2 demonstrates the axiom selection algorithms proposed.

## The Symmetric Weighted-average Extended Hutchinson Distance

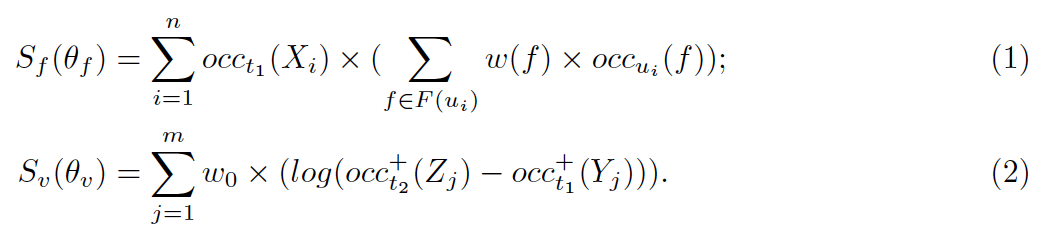
Terms are the most basic structure in the first-order logic. A reasonable term metric used for evaluating the term difference can guarantee its extended atom and formula metrics perform well to some extent. For two arbitrary terms *t*1, *t*2, suppose that there exists a substitution *θ* such that *t*1*θ* = *t*2, which reflects the changes from *t*1 to *t*2. Not all two arbitrary terms exist a substitution, and it is thus unrealistic to evaluate the term difference by substitutions directly. To address the problem, we take the least general generalization *lgg* of terms as a medium to evaluate the term difference. Besides, we also propose a way to evaluate the term functional and variable differences separately.

Let θ = {X1 → u1, ..., Xn → un, Y1 → Z1, ..., Ym → Zm} be an arbitrary substitution, where u1, ..., un are functional terms, Z1, ..., Zm are variables, and X1, ..., Xn, Y1, ..., Ym are distinct variables. Functional substitution θf and variable substitution θv are defined as:

θf = {X1 → u1, ..., Xn → un},

θv = {Y1 → Z1, ..., Ym → Zm}.

Obviously, θf ∩ θv = ∅ and θf ∪ θv = θ. In some special cases, θf or θv may be an empty substitution. We then put forward two functions Sf and Sv, mapping θf and θv to real numbers, in which Sf mainly considers the functional difference while Sv only takes the variable difference into account. Sf , Sv are defined as:



Where,

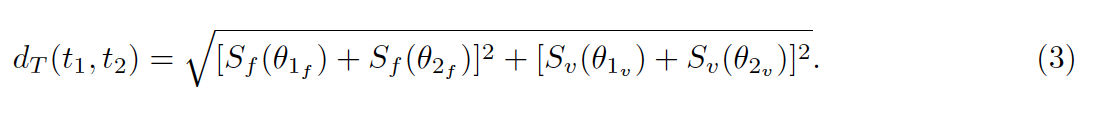
* + - *w* is a weight function that maps every function symbol to a non-negative integer;
    - *w*0 (*w*0 *>* 0) is a constant representing the weight for every variable;
    - *F* (*t*) denotes the set of all function symbols appearing in a term *t*;
    - *occt*(*s*) denotes the number of occurrences of a function or variable symbol *s* in a term *t*;

*occt* (*v*) denotes the number of deep occurrences of a variable *v* in a term *t*, which takes the depth of *v* (the number of symbols nested *v*) into consideration. For every occurrence *I* of *v*, the depth of *v* is ni (ni >=0) 

+

*•*

Given two terms *t*1 and *t*2, *t* is their the least general generalization with substitutions *θ*1 and *θ*2, such that *tθ*1 = *t*1 and *tθ*2 = *t*2. Based on proposed functions, the term difference function *dT* between *t*1 and *t*2 is defined as:



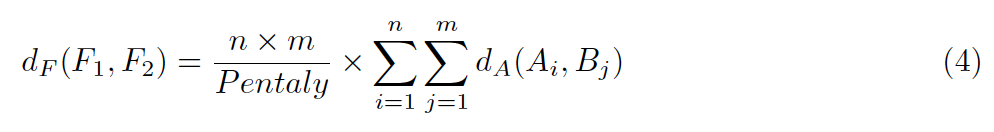
Compatible atoms can also construct the most general generalization as terms do in the same way, thus the dT can apply to them naturally. As for incompatible atoms, we simply think their difference is extremely huge due to they cannot be unified. Hence,

* + - if *A*1, *A*2 are compatible atoms, *dA*(*A*1*, A*2) = *dT* (*A*1*, A*2);
    - if *A*1, *A*2 are incompatible atoms, *dA*(*A*1*, A*2) = +*∞*.

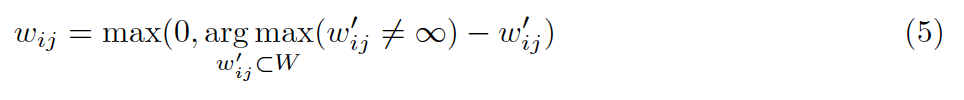
First-order formulas are the connection of atoms, logical connectives and quantifiers. Ignoring all the logical connectives and quantifiers, formulas are sets of atoms. The inference will not happen in such formulas, if all atoms in them are incompatible. We assert that the more incompatible atoms two formulas have, the less similar the formulas are.

Suppose that F1, F2 are two formulas. Let D1 = {A1, ..., An} and D2 = {B1, ..., Bm}, which denote the corresponding atom sets, respectively.

Penatly = |{dA(Ai, Bj)|dA(Ai, Bj) = (+∞, +∞)}| is the number of incompatible atom pairs (Ai, Bj) in formulas. The formula difference function dF between F1 and F2 is defined as:



First, we converted a logic problem into a fully-connected graph G = (V, E), where each node vi represents a logical formula, and each edge eij has weight w’ij representing the dissimilarity between nodes vi and vj. We devised the following method to convert each dissimilarity weight w’ij into a similarity weight wij:



The conversion method consists of three cases:

* + - If the dissimilarity weight w’ij is equal to the largest dissimilarity weight that is not infinity, or if w’ij = *∞*, then our method will assign a similarity weight wij= 0 to the edge eij.
    - If the dissimilarity weight wij= 0, then our method will assign a similarity weight equal to the largest dissimilarity weight that is not infinity
    - Any dissimilarity weight w’ijbetween these two extremes will yield a similarity weight *wij*

that is between 0 and 

We used the method to convert the dissimilarity matrix W’ into a similarity matrix W.

## Axiom selection algorithms design

In this section, some axiom selection algorithms based on the graph were proposed. These algorithms rely on machine learning and artificial intelligence search techniques. In machine learning fields, spectral clustering is used to select about half of the needed axioms. In search methods, local search is used to find a path from the conjecture to one of its infinity distance axioms. The local search algorithm is a greedy strategy by finding the optimal solution only based on the current state. The local beam search algorithm is implemented. There are 2 experiments. The first one is finding a random path in which each node is visited its nearest neighbor. The process starts from conjecture to one of the axioms that are the infinity distance from the conjecture to them. And the second one is to find all paths from the conjecture to the infinity distance node connected to the conjecture. Besides the search path, 2 methods are coming up to improve the robustness of axiom selection.

### **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, spectral clustering is implemented by using python sklearn package 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 cluster

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 into 2 parts which have 1 part includes all the needed axioms.

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

The local beam search uses a greedy search strategy which is finding the local optimal state based on the current state. However, the local beam search not only finds one local optimal state but also keeps track of more local optimal states. In this study, the definition of the locally optimal solution is 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 have the minimum dissimilarity. The BFS starts at the conjecture and finishes once it reaches 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 solution. The second is to 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 reaching 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 hasn’t be visited yet. Gray means the node is been visiting now. Black means the node’s all neighbors have been visited. The attributed 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 enqueueing is random. After dequeue, the first node u, its child v will recode its parent to be u. After a terminate node is visited. The backtracking process will be implemented by recursively search the node and its parent. The path is saved in a set without considering 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 continues until the queue is empty. The process is shown below:



if u != terminate node:



if u = terminate node:





path.append(u)

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

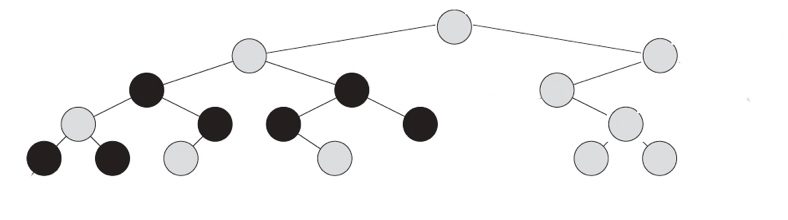


Figure 1: local beam search path

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 finds 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 calculates 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 paths above.

The framework is based on 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 the total number of nodes in the 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 [9]

In Figure 2, the green point needs 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 the 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 can 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 is added in the local beam search; the third element represents the second closets neighbor, and so on.

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

If size(SUDS)>=N:

add N closest dissimilarity axioms to the Allpath set  
 else if size(SUDS)<N:

add 1 closest dissimilarity axioms to the Allpath set.

Figure 3 provides a simple illustrative example, in which the C node is the conjecture and the A nodes are the axioms. The edge weights are the similarities between the formulae (the figure omits edges that do not affect the example). Starting at the conjecture node, a greedy tree is built by iteratively visiting the nodes most similar to the current leaves of the tree, until nodes that have zero similarity (or equivalently, infinite dissimilarity) to the conjecture are reached, (these infinitely dissimilar nodes are ignored). In Figure 3 the thicker edges are those that are followed, and the nodes with thicker circles are the nodes at which the tree growth stops because they have zero similarity to the conjecture. At that stage, the light grey axiom nodes are in the tree. As a final step, all the unvisited nodes have most similar to the nodes in the greedy tree are added to the tree. In Figure 3, that adds all dark grey axiom nodes to the tree. The axiom nodes in the tree are selected.



Figure 3: a greedy tree

# Evaluation Results

This chapter comes up with several performance indices and evaluates the performance of axiom selection techniques above.

## Results

Two datasets (Bushy and Chainy) from the MPTP 2078 is used to evaluate the algorithm. There are both 325 tasks in the Bushy and Chainy dataset. There are 5 features to evaluate algorithm: the precision, selectivity, adequacy, adequate precision, adequate selectivity. The definitions of the 5 indices are illustrated below.

1. The definition of **precision** is
2. The definition of **selectivity** is
3. The definition of **adequacy** is the number of tasks that the accuracy is not 0.
4. The definition of **adequate precision** is
5. The definition of **adequate selectivity** is

The evaluation result for Bushy and Chainy is illustrated in Table 1 and for new datasets are shown in Table 2.

Table 1: evaluation result

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bushy | | | | | Chainy | | | | |
| Algorithm | precision | selectivity | adequacy | Adequate precision | Adequate selectivity | precision | selectivity | adequacy | Adequate precision | Adequate selectivity |
| E | 0.35 | 1 | 325 | 0.35 | 1 | 0.06 | 1 | 325 | 0.06 | 1 |
| vampire | 0.32 | 0.8 | 260 | 0.39 | 0.84 | 0.08 | 0.55 | 304 | 0.09 | 0.56 |
| Qinf | 0.43 | 0.54 | 241 | 0.58 | 0.61 | 0.08 | 0.53 | 277 | 0.09 | 0.56 |
| Spectral clustering | 0.39 | 0.68 | 300 | 0.42 | 0.70 | 0.06 | 0.94 | 325 | 0.06 | 0.94 |
| Local beam search random path | 0.33 | 0.73 | 277 | 0.39 | 0.78 | 0.02 | 0.63 | 154 | 0.04 | 0.63 |
| Local beam search all path | 0.36 | 0.77 | 304 | 0.39 | 0.78 | 0.05 | 0.91 | 294 | 0.05 | 0.93 |
| A\* search all path | 0.13 | 0.40 | 148 | 0.28 | 0.79 | 0.02 | 0.75 | 176 | 0.04 | 0.82 |
| Knn search all path (K=1) | 0.36 | 0.57 | 235 | 0.50 | 0.61 | 0.05 | 0.79 | 278 | 0.06 | 0.85 |

Table 2: evaluation result of new sets

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bushy | | | | | Chainy | | | | |
| Algorithm | precision | selectivity | adequacy | Adequate precision | Adequate selectivity | precision | selectivity | adequacy | Adequate precision | Adequate selectivity |
| Spectral clustering | 0.33 | 0.74 | 1475 | 0.35 | 0.76 | 0.04 | 0.95 | 705 | 0.04 | 0.95 |
| Local beam search random path | 0.17 | 0.58 | 747 | 0.36 | 0.70 | 0.01 | 0.65 | 407 | 0.02 | 0.66 |
| Local beam search all path | 0.30 | 0.80 | 1392 | 0.33 | 0.83 | 0.33 | 0.96 | 682 | 0.03 | 0.96 |
| A\* search all path | 0.13 | 0.58 | 718 | 0.27 | 0.80 | 0.01 | 0.80 | 448 | 0.02 | 0.85 |
| Knn search all path (K=1) | 0.30 | 0.65 | 1228 | 0.38 | 0.70 | 0.03 | 0.89 | 673 | 0.04 | 0.90 |

## Discussions

In Table 1, the results of E, vampire and Qinf are baselines. Qinf takes a problem consisting of a conjecture C and axioms A, and selects all axioms Φ such that . This simply means that each axiom contains at least one atom whose predicate symbol matches that of an atom in the conjecture.The selector E is the worst baseline and the Qinf is the best baseline. According to the definition of selectivity, the lower selectivity, the lower the unnecessary axioms it has. The selectivity of E is1 that means it selects all axioms in the tasks and Qinf selects less than the vampire. The machine learning algorithm is spectral clustering and extras are graph search algorithms.

Among all techniques, the spectral clustering technique gets the highest precision in both the Bushy and Chainy dataset. Its selectivity is lower than the vampire but higher in Chainy. Thus, the spectral clustering is not an optimal method.

In graph search strategies, the best search structure is the local beam search for all path-based strategy. Moreover, in improve robustness, KNN has better performance compare to spectral clustering. In Table 1, the local beam search of all paths combined with KNN has the best performance of axiom selection with an accuracy of 0.36 on Bushy and 0.05 in Chainy. In Table 2, as the accuracy of 0.3 in Bushy and 0.03 in Chainy.

# Conclusion

This paper has presented artificial intelligence axiom techniques. It also evaluates these techniques without having to run an ATP system. The results demonstrate that the local beam search for all path-based strategy has the best performance.

Future work includes a more comprehensive evaluation of the metrics, to confirm that the metrics provide a meaningful evaluation of the axiom selection techniques. The result demonstrates there are more spaces for improvement in axiom selection. One of the directions is spectral clustering combined with the local beam search.

Moreover, the spectral clustering has some drawbacks that can be improved in the future. One of the drawbacks is its select different initial centroids each time so that the cluster classification various among each experiment. To solve this problem, a new spectral clustering algorithm could be introduced [10]. A different similarity matrix can be calculated by using the normalized Laplacian matrix from the graph dissimilarity matrix.

Another problem is the optimal number of clusters. It can be solved by experiment with each task of N logical formulas starting with the number of clusters from 2 to N/2. Then using statistical methods to define the optimal value over the dataset. Finally, pipelining axiom selection tools to increase precision and decrease selectivity.

# Acknowledgment

Foremost, I would like to express my sincere gratitude to my advisor Professor Dr. Geoff Sutcliffe for the continuous support of my research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master's study.

Besides my advisor, I would like to appreciate that Professor Dr. Liang Liang gave me technical supports on machine learning. Moreover, I would like to thank Professor Dr. Ubbo Visser taught me the course of the introduction of artificial intelligence. I also thanks Professor Dr. Odelia Schwartz and Professor Dr. Victor Milenkovic taught me algorithm design related courses. These courses gave me theorems on search strategies and data structure supports on implementation.

I also sincere thanks to our team members Qinghua Liu and Zishi Wu. Qinghua contributed the technical supports about the mathematical model of the dissimilarity matrix in section 3.1. Without her help, I could not have great achievements. Zishi gave me helps on algorithm optimization and another way of spectral clustering that can pipeline with the graph search strategies.

# References

[1] J. Alama, T. Heskes, D. Kühlwein, E. Tsivtsivadze, and J. Urban, “Premise Selection for Mathematics by Corpus Analysis and Kernel Methods,” *J Autom Reasoning*, vol. 52, no. 2, pp. 191–213, Feb. 2014, doi: 10.1007/s10817-013-9286-5.

[2] C. Kaliszyk and J. Urban, “MizAR 40 for Mizar 40,” *J Autom Reasoning*, vol. 55, no. 3, pp. 245–256, Oct. 2015, doi: 10.1007/s10817-015-9330-8.

[3] C. E. Brown, T. Gauthier, C. Kaliszyk, G. Sutcliffe, and J. Urban, “GRUNGE: A Grand Unified ATP Challenge,” *arXiv:1903.02539 [cs]*, vol. 11716, pp. 123–141, 2019, doi: 10.1007/978-3-030-29436-6\_8.

[4] E. Kuksa and T. Mossakowski, “Prover-independent Axiom Selection for Automated Theorem Proving in Ontohub,” p. 13.

[5] J. Urban, “MaLARea: a Metasystem for Automated Reasoning in Large Theories,” p. 14.

[6] A. Armando, P. Baumgartner, and G. Dowek, Eds., *Automated Reasoning: 4th International Joint Conference, IJCAR 2008 Sydney, Australia, August 12-15, 2008 Proceedings*, vol. 5195. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[7] L. Kovács and A. Voronkov, “First-Order Theorem Proving and Vampire,” in *Computer Aided Verification*, Berlin, Heidelberg, 2013, pp. 1–35, doi: 10.1007/978-3-642-39799-8\_1.

[8] “(PDF) E - A Brainiac Theorem Prover,” *ResearchGate*. https://www.researchgate.net/publication/2556169\_E\_-\_A\_Brainiac\_Theorem\_Prover (accessed Apr. 18, 2020).

[9] “*k*-nearest neighbors algorithm,” *Wikipedia*. Apr. 14, 2020, Accessed: May 01, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=K-nearest\_neighbors\_algorithm&oldid=950855987.

[10] U. von Luxburg, “A Tutorial on Spectral Clustering,” *arXiv:0711.0189 [cs]*, Nov. 2007, Accessed: Apr. 27, 2020. [Online]. Available: http://arxiv.org/abs/0711.0189.