Technical Report TR-ARP-09-00

Automated Reasoning Group, Computer Sciences Laboratory
Research School of Information Sciences and Engineering
Australian National University

October 9, 2000

Homogeneous Sets of ATP Problems

Matthias Fuchs
Automated Reasoning Group
Australian National University, Australia
Email: fuchs@arp.anu.edu.au

Geoff Sutcliffe
School of Information Technology
James Cook University, Australia
Email: geoff@cs.jcu.edu.au

Abstract This report describes how the homogeneity of sets of ATP problems can be measured with respect to the performance of ATP systems. Measuring homogeneity is important as a basis for empirical evaluation of ATP systems and problems. The method developed assigns problems to nodes in a graph, and finding homogeneous sets of problems is reduced to finding maximal cliques in the graph. The method is robust to the realities of empirical data collection. A machine learning approach has been used to differentiate between types of problems in situations where heterogeneity is apparent.

1 Introduction

A key concern of automated theorem proving (ATP) research is the development of more powerful systems, capable of solving more difficult problems. In order to build more powerful systems, it is important to know which systems, and hence which techniques (understanding that a system is a collection and combination of techniques), work well for what types of problems. This knowledge is a key to further development, as it precedes any investigation into why the techniques and systems work well or badly. This knowledge is also crucial for users: given a specific problem, a user would like to know which systems are most likely to solve it. Inextricably intertwined with the evaluation of ATP systems is the evaluation of ATP problem difficulty. Evaluation of problem difficulty is important, as it simplifies problem selection according to a user's intentions, and over the years, changes in problem ratings provide a quantitative indicator of advancement in ATP.

Analytic approaches to ATP system evaluation provide insights into theoretical system capabilities. Complete analysis of the search space at the 1st order level is of course impossible (for otherwise 1st order logic would be decidable). It is therefore necessary to make empirical evaluations of ATP systems. [SS00] presents methodologies for the empirical evaluation of ATP systems and problems. Due to the specialisation of ATP systems and techniques to problems with certain characteristics, e.g., special techniques are deserved for problems with equality, the evaluations must be done in the context of sets of problems that are reasonably homogeneous with respect to the systems.[1] These sets of problems are called *Specialist Problem Classes* (SPCs). The evaluation methodologies of [SS00] evaluate systems and problems within individual SPCs as follows. Initially a partial ordering of the systems is determined by *subsumption*: a system that solves a strict superset of the problems solved by another system subsumes (is better than) the other system. Problems are then rated according to the fraction of non-subsumed systems that fail to solve the problem within realistic resource limits. (Roughly, a realistic resource limit for a given ATP system is one beyond which a linear increase in resources would not lead to the solution of significantly more problems; see [SS00] for details.) Problems with a rating of zero are *easy*, with a rating between zero and one are *difficult*, and with a rating of one are *unsolved*. Finally, the ATP systems are rated according to the fraction of difficult problems they can solve within realistic resource limits.

A cornerstone of the evaluation methodologies is the identification of the SPCs that are "reasonably homogeneous with respect to the systems". The SPCs are based on logical, language, and syntactic characteristics of the problems. The choice of what characteristics are used has been based on community input and ad hoc analysis of system performance data. The range of characteristics that have so far been identified as relevant are:

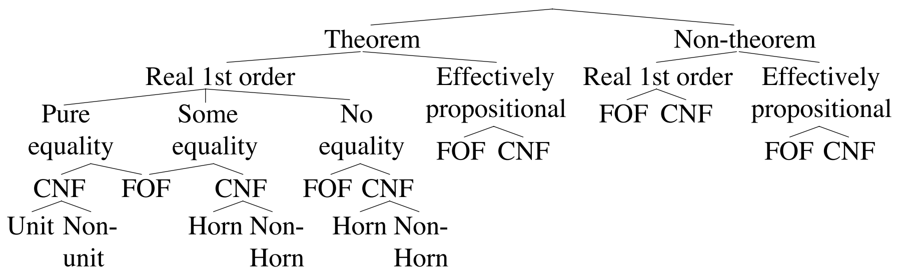
- Theoremhood: Theorems vs Non-theorems

[1]In the same way that athletes who specialise in a certain event should be evaluated in that type of event, rather than over all possible events.

- Order: Essentially propositional vs Real 1st order
- Equality: No equality vs Some equality vs Pure equality
- Form: CNF (Clause Normal Form) vs FOF (First Order Form)
- Hornness: Horn vs Non-Horn
- Unit equality: Unit equality vs Non-unit pure equality

Based on these characteristics 14 SPCs have been defined, as indicated by the leaves of the tree in Figure 1.

Figure 1: Specialist Problem Classes



The evaluation methodologies have been applied to current 1st order ATP systems, using performance data on the TPTP (Thousands of Problems for Theorem Provers) problem library [SS98]. The evaluation results produced using these SPCs are apparently meaningful and realistic; systems that are rated highly within a SPC perform well within the SPC, and are recognised by the community as powerful for that type of problem. Similarly, TPTP problems that have high ratings are hard to solve using current ATP technology, and no anomalies in the problem ratings have been pointed out by the ATP community. It would seem that the SPCs are adequately homogeneous. It is important that this situation be maintained, and to this end some mechanical analysis and measurement of SPC homogeneity is desirable.

This report describes a method for checking the homogeneity of a set of problems, with respect to performance data on those problems. The method can also be used to identify homogeneous subsets of a set of problems. The report also shows how a machine learning technique can be used to identify problem characteristics that differentiate homogeneous problem sets. These methods and techniques have been used to check the homogeneity of the SPCs shown in Figure 1, and hence to affirm the basis for the system and problem evaluations done using the TPTP.

Section 2 introduces the use of graphs to represent problem types and homogeneity relationships between problem types. In this representation, finding homogeneous sets of problems corresponds to finding cliques in the graph, and the clique finding algorithm is described. Section 3 reports on the testing performed, testing the homogeneity of the

SPCs shown in Figure 1 and independently identifying SPCs in the TPTP. Section 4 describes the use of machine learning to find problem characteristics that differentiate SPCs. Section 5 concludes the report.

2 Cliques of Problems

As outlined in Section 1, a requirement for meaningful evaluation of ATP systems is evaluation within the context of sets of problems that are reasonably homogeneous with respect to the performances of the ATP systems. Given a SPC, if there are some problems that are solved by system A_1 but not by A_2, and there are some problems that are solved by A_2 but not by A_1, then the performances are contradictory and the SPC is not homogeneous with respect to A_1 and A_2. In contrast, if there is no such contradictory performance then the SPC is homogeneous with respect to A_1 and A_2. If a SPC is homogeneous with respect to n ATP systems $A_1,\ldots,A_n$, then the systems can be totally ordered using system subsumption, as defined in Section 1. In this case it makes sense to say that one system is better than another, in the context of the SPC. Note that homogeneity does not prevent there being a substantial variation in the number of problems solved by the systems. To measure the homogeneity of a SPC, the performance data of ATP systems can be examined for contradictory behaviour as follows.

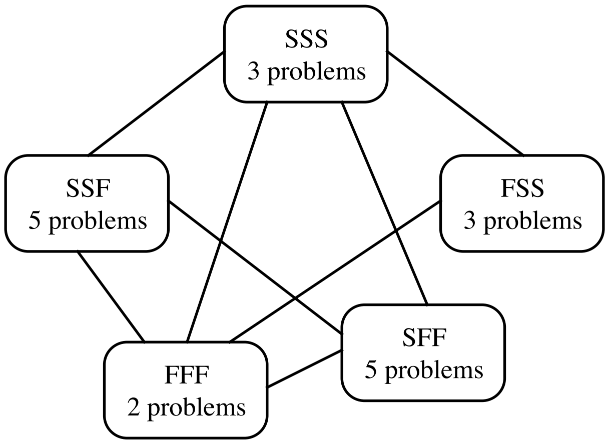
2.1 Basic Compatibility and Homogeneity Measures

For each problem in a SPC, an ATP system either solves it or fails to solve it (within reasonable resource limits). These two cases are represented by S and F. Each problem P in a SPC is associated with a performance vector $v^P \in \{\mathtt{S},\mathtt{F}\}^n$, where v_i^P indicates the performance of ATP A_i on P. Two performance vectors are *compatible* to the extent that the systems' performances do not contradict each other. Two performance vectors u and v are compatible if there are no two vector positions i and j such that $u_i = \mathtt{S}$ and $u_j = \mathtt{F}$ but $v_i = \mathtt{F}$ and $v_j = \mathtt{S}$. For example, for $n = 3$, the performance vectors SSS, SSF, SFF, and FFF are pairwise compatible, whereas SSF and FSS are each compatible with SSS but are not compatible with each other. (Finer grained levels of compatibility are considered in Section 2.2 below.)

The performance vectors that occur for a SPC are used to divide the SPC into disjoint problem groups. Each group, with the associated performance vector, is a node in a *compatibility graph* G. An edge connects two nodes in G if the respective performance vectors are compatible. The cliques of G then identify homogeneous sets of problems. The homogeneity of a SPC is expressed in terms of the maximal clique size in G. Two *homogeneity measures* are defined, using two ways of measuring clique size. The first measure is the ratio of the number of problems in the clique's nodes and the number of problems in the SPC, i.e., measuring the homogeneity of the actual problems in the SPC. The second measure is the ratio of the number of nodes in the clique and the number of nodes in the graph. This measure acknowledges that the number of problems may be biased by the source of the problems, and thus puts the problems in a node into an equivalence class, i.e., measuring homogeneity of the problem types.

If one clique covers most of a SPC, then the SPC can be considered homogeneous. Conversely, there is a strong argument for splitting a SPC if it generates two or more rather large cliques. Figure 2 shows a compatibility graph, in which each node is shown with its performance vector and the number of problems. The maximal clique is {`SSS`, `SSF`, `SFF`, `FFF`}, containing 15 of the 18 problems and 4 of the 5 nodes. The problem set is thus 83% homogeneous by problem count and 80% homogeneous by node count.

Figure 2: Simple Compatibility and Clique Example



2.2 Refinements of the Compatibility and Homogeneity Measures

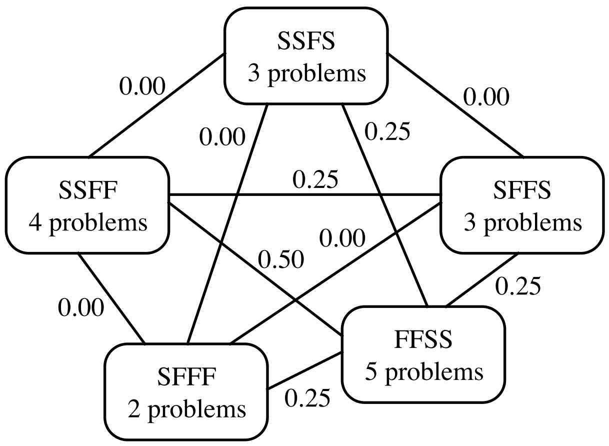
The notion of compatibility introduced in Section 2.1 is strictly Boolean: two nodes are either compatible or not, depending on which systems solve and which systems fail to solve the associated problems. Although decent ATP systems have reasonably stable performance characteristics, it is a feature of ATP that small changes in problems can sometimes lead to strange changes in system performance. Such aberrant behaviour may cause the performance vectors of two problems to appear incompatible, when for the underlying purpose of system evaluation the problems really do fall into the same homogeneous SPC. To make the homogeneity measures robust, a *degree of compatibility* between two performance vectors is defined. The connectivity of the compatibility graph is then determined with respect to a degree of compatibility, with a subsequent effect on the homogeneity measure.

The compatibility of two performance vectors u and v is defined as the minimal fraction of ATP systems that have to exchange `S` for `F`, or vice versa, to make u and v fully compatible. For example, if u = `SSFS` and v = `FFSS` (i.e., there are four systems), by changing the third position of u (v) to `S` (`F`), the two vectors become fully compatible.

The degree of compatibility therefore is $1/4 = 0.25$. The degree of compatibility can range from 0 to 0.5 (less than 0.5 if there is an odd number of ATP systems). The value 0 indicates full compatibility (no change required), as is the case for the performance vectors `SSFF` and `SFFF`. The value 0.5 indicates complete incompatibility, as is the case for `SSFF` and `FFSS`.

When constructing a compatibility graph G, a *compatibility threshold* $d \in [0; 0.5]$ is specified, which allows two nodes to be connected if their degree of compatibility is less than or equal to d. In this way absolute compatibility need not be expected, but excessive contradictory behaviour can be avoided in order to support a reasonable evaluation of (relative) system performances. Note that for $d = 0$ we have the original case that requires full compatibility. Figure 3 shows a compatibility graph, in which the edges are annotated with the degree of compatibility between the nodes. With a compatibility threshold of 0.00 the maximal clique is {`SSFS`, `SSFF`, `SFFF`} containing 9 of the 17 problems and 3 of the 5 nodes. With an increased compatibility threshold of 0.25 the maximal clique is {`SSFS`, `SFFF`, `SFFS`, `FFSS`} containing 13 of the 17 problems and 4 of the 5 nodes.

Figure 3: Degree of Compatibility Example

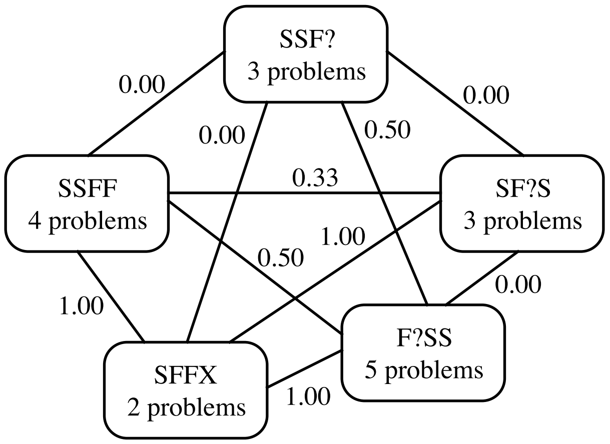


The development so far assumes that every system has attempted every problem in the SPC being considered. In reality this may not be the case, for two reasons. Firstly, if a problem is added to the TPTP after a certain ATP system was tested, there is no performance data available for that system on that problem. In the performance vectors such cases are denoted by `?`. When computing the compatibility of two performance vectors, positions where either vector has a `?` entry are ignored. For example, the degree of compatibility of `SSF?` and `F?SS` is $1/2 = 0.5$. Secondly, certain ATP systems are incapable of attempting certain types of problems, e.g., systems based on unfailing completion can deal with only unit equality problems, and are therefore not tested on some problems. In the performance vectors such cases are denoted by `X`. If two problems P and Q have performance vectors v^P and v^Q respectively, and there is an ATP system A_i (position i) so that $v_i^P = \texttt{X}$ and $v_i^Q \in \{\texttt{S}, \texttt{F}\}$, this indicates that P

and Q should be considered to be inherently different in nature. Consequently these two problems should not be in the same SPC and hence should not be connected in the compatibility graph G. This is achieved by assigning a degree of compatibility 1.0 to v^P and v^Q if the above situation occurs. 1.0 can be viewed as "infinity", given that the degree of compatibility is bounded by 0.5 under normal circumstances.

Figure 4 shows the degrees of compatibility between the five nodes, some of which have `?` and `X` entries.

Figure 4: Missing Data Example



2.3 Finding Maximal Cliques

Maximal clique detection is known to be NP complete [Meh84]. Hence, except for small graphs, it is necessary to resort to efficient approximation algorithms. A fair amount of research has gone into approximation algorithms of this kind, most recently in the field of evolutionary computation [Hay98]. The algorithm used to find the maximal cliques in the compatibility graphs is best characterised as a greedy or steepest ascent hill-climbing algorithm. Starting with the graph $G_0 = G$, the node of minimal degree is removed from G_i, resulting in G_{i+1}. If there is more than one such node, the one containing the smallest number of problems is removed, thus creating a bias towards cliques containing more problems. The process is stopped as soon as G_i is a clique. After a clique is identified, this procedure is applied to the graph consisting of the remaining nodes, until the original graph G has been partitioned into $k \geq 1$ cliques $C_1, \ldots, C_k$. C_{i+1} may contain more problems than C_i since the number of problems is only a secondary criterion. C_{i+1} may also have more nodes than C_i because the algorithm is an approximation algorithm and is therefore not guaranteed to find the clique with the maximal number of nodes. The test results presented in Section 3 demonstrate that this simple algorithm performs satisfactorily well for the compatibility graphs.

3 Testing and Results

3.1 Testing the Existing SPCs for Homogeneity

The homogeneities of the 14 SPCs shown in Figure 1 have been measured, using data from systems tested on the TPTP since the release of v2.0.0 on 5 June 1997, up to 14 September 2000. Data from 16 systems was used, except for the SPCs containing non-theorems, i.e., those starting `SAT_` in the tables below, for which data from 14 systems was used.[2] For each SPC the homogeneity measures by problem count and node count were computed for compatibility thresholds 0.000, 0.0625, 0.1250, 0.2500, and 0.500. The results are shown in Tables 1 and 2.

`SAT_EPR_CNF`

C.T.	31 nodes		139 problems	
0.0000	15	(48%)	77	(55%)
0.0625	15	(48%)	77	(55%)
0.1250	26	(83%)	125	(89%)
0.2500	31	(100%)	139	(100%)
0.5000	31	(100%)	139	(100%)

`SAT_EPR_FOF`

C.T.	16 nodes		83 problems	
0.0000	11	(68%)	73	(87%)
0.0625	11	(68%)	73	(87%)
0.1250	15	(93%)	82	(98%)
0.2500	16	(100%)	83	(100%)
0.5000	16	(100%)	83	(100%)

`SAT_RFO_CNF`

C.T.	38 nodes		88 problems	
0.0000	14	(36%)	41	(46%)
0.0625	14	(36%)	41	(46%)
0.1250	25	(65%)	56	(63%)
0.2500	30	(78%)	68	(77%)
0.5000	30	(78%)	68	(77%)

`SAT_RFO_FOF`

C.T.	4 nodes		9 problems	
0.0000	4	(100%)	9	(100%)
0.0625	4	(100%)	9	(100%)
0.1250	4	(100%)	9	(100%)
0.2500	4	(100%)	9	(100%)
0.5000	4	(100%)	9	(100%)

`THM_EPR_CNF`

C.T.	41 nodes		401 problems	
0.0000	14	(34%)	345	(86%)
0.0625	18	(43%)	359	(89%)
0.1250	32	(78%)	391	(97%)
0.2500	40	(97%)	400	(99%)
0.5000	40	(97%)	400	(99%)

`THM_EPR_FOF`

C.T.	22 nodes		235 problems	
0.0000	15	(68%)	225	(95%)
0.0625	15	(68%)	225	(95%)
0.1250	21	(95%)	234	(99%)
0.2500	22	(100%)	235	(100%)
0.5000	22	(100%)	235	(100%)

`THM_RFO_EQU_FOF`

C.T.	32 nodes		323 problems	
0.0000	16	(50%)	154	(47%)
0.0625	16	(50%)	154	(47%)
0.1250	26	(81%)	310	(95%)
0.2500	32	(100%)	323	(100%)
0.5000	32	(100%)	323	(100%)

`THM_RFO_NEQ_FOF`

C.T.	6 nodes		21 problems	
0.0000	5	(83%)	19	(90%)
0.0625	5	(83%)	19	(90%)
0.1250	6	(100%)	21	(100%)
0.2500	6	(100%)	21	(100%)
0.5000	6	(100%)	21	(100%)

Table 1: Statistics for the currently used SPCs, part I

[2] This difference is due to two systems not being tested on problems known to be non-theorems at the time, whereas more of the problems are actually non-theorems.

`THM_RFO_NEQ_CNF_HRN`					`THM_RFO_NEQ_CNF_NHN`				
C.T.	72 nodes		379 problems		C.T.	85 nodes		430 problems	
0.0000	18	(25%)	240	(63%)	0.0000	20	(23%)	322	(74%)
0.0625	37	(51%)	307	(81%)	0.0625	23	(27%)	288	(66%)
0.1250	48	(66%)	312	(82%)	0.1250	46	(54%)	379	(88%)
0.2500	69	(95%)	376	(99%)	0.2500	79	(92%)	422	(98%)
0.5000	72	(100%)	379	(100%)	0.5000	85	(100%)	430	(100%)
`THM_RFO_SEQ_CNF_HRN`					`THM_RFO_SEQ_CNF_NHN`				
C.T.	67 nodes		381 problems		C.T.	154 nodes		1194 problems	
0.0000	20	(29%)	246	(64%)	0.0000	29	(18%)	959	(80%)
0.0625	38	(56%)	325	(85%)	0.0625	42	(27%)	955	(79%)
0.1250	55	(82%)	349	(91%)	0.1250	82	(53%)	1080	(90%)
0.2500	66	(98%)	380	(99%)	0.2500	153	(99%)	1193	(99%)
0.5000	67	(100%)	381	(100%)	0.5000	154	(100%)	1194	(100%)
`THM_RFO_PEQ_CNF_NUE`					`THM_RFO_PEQ_CNF_UEQ`				
C.T.	44 nodes		122 problems		C.T.	126 nodes		424 problems	
0.0000	22	(50%)	94	(77%)	0.0000	31	(24%)	279	(65%)
0.0625	29	(65%)	105	(86%)	0.0625	44	(34%)	315	(74%)
0.1250	38	(86%)	112	(91%)	0.1250	74	(58%)	359	(84%)
0.2500	44	(100%)	122	(100%)	0.2500	116	(92%)	414	(97%)
0.5000	44	(100%)	122	(100%)	0.5000	126	(100%)	424	(100%)

Table 2: Statistics for the currently used SPCs, part II

For all but one of the SPCs, a homogeneity measure in excess of 80% by problem count is reached with a compatibility threshold of 0.125 (and in many cases with a compatibility threshold of 0.0625). The exception is `SAT_RFO_CNF`. Here the homogeneity measure remains below 100% even with a compatibility threshold of 0.5000, indicating that some nodes are incompatible due to `X` entries in performance vectors. Closer examination shows that some systems had (quite reasonably) been tested on only the unit equality problems in that SPC. Excluding those systems from consideration produces a homogeneity measure of 86% by both node count and problem count is reached with a compatibility threshold of 0.1250. This suggests a possible split for the SPC, based on equality characteristics.

For all but four of the SPCs (excluding `SAT_RFO_CNF` discussed above), a homogeneity measure in excess of 80% by node count is reached with a compatibility threshold of 0.125. The exceptions are the SPCs `THM_RFO_NEQ_CNF_HRN`, `THM_RFO_NEQ_CNF_NHN`, `THM_RFO_SEQ_CNF_NHN`, and `THM_RFO_PEQ_CNF_UEQ`. For SPCs `THM_RFO_NEQ_CNF_NHN` and `THM_RFO_SEQ_CNF_NHN`, homogeneity measures of 78% and 81% by node count are reached respectively with a compatibility threshold of 0.14. For `THM_RFO_NEQ_CNF_HRN`, 12 of the 15 nodes in the second clique have one problem. Interestingly, there is one node with 24 problems, all but one of which are blocks world problems from the TPTP's `PLA` domain. These problems have a high proportion of unit clauses. The node is excluded from the first clique due to different performance from two tableau based (i.e., strongly goal oriented) systems, possibly suggesting that tableau based systems have better performance when there are many unit clauses present. For `THM_RFO_PEQ_CNF_UEQ`, the second and subsequent cliques found have reasonably large numbers of nodes, but all nodes have only very few (typically one or two) problems. This low homogeneity by node count is somewhat surprising, given the specialised nature of unit equality problem solving. One possible cause is that some systems use some highly specialised techniques, which may be suited to particular problems.

3.2 Generating Homogeneous Problem Sets

The technique of finding maximal cliques in a compatibility graph has also been used to divide up the TPTP into subsets that are homogeneous with respect to the performance data. The subsets are formed from the problems in the nodes of the cliques found in the compatibility graph for the performance data over the whole TPTP. Using a compatibility threshold of 0.125 the 4229 problems in the TPTP are divided into 35 homogeneous subsets, with 3972 problems falling into the first seven subsets. All of the remaining 28 subsets contain 25 or less problems, and are ignored as insignificant.

The generated homogeneous subsets have been compared to the existing SPCs. If any generated subset is a strict superset of a SPC, then the SPC is 100% homogeneous. If multiple SPCs fall within a single homogeneous subset, then the union of those SPCs is homogeneous with respect to the systems' performances, and merging them may be appropriate. In contrast, if a SPC is split across multiple subsets, then the SPC is apparently heterogeneous and may need to be split. In order to make such judgements, for each homogeneous subset and each SPC, the ratio of the size of their intersection and the size of the SPC has been computed. The results are shown in Table 3.

The first subset encompasses 2418 of the 3558 CNF problems in the TPTP, including most of those in the `THM_EPR_CNF`, `THM_RFO_NEQ_CNF_HRN`, `THM_RFO_NEQ_CNF_NHN`, `THM_RFO_SEQ_CNF_HRN`, `THM_RFO_SEQ_CNF_NHN`, and `THM_RFO_PEQ_CNF_NUE` SPCs. This shows that there are techniques (and hence systems) that are effective for all these problem types. The third subset identifies the unit equality SPC. The fifth subset covers all the FOF SPCs. Evidently the techniques suitable for one type of FOF problem are adequate for most types. The sixth subset identifies the CNF non-theorems, both the effectively propositional problems and the real first order ones. The only SPC that does not have a large fraction contained within only one problem clique is `SAT_RFO_CNF`. This heterogeneity was also identified in Section 3.1, and the reason discussed.

4 Using Machine Learning to Differentiate SPCs

Section 3 shows that the SPCs in Figure 1, formed using the problem characteristics listed in Section 1, are mostly homogeneous. In the situations where some heterogeneity is apparent, it is useful to have a method of identifying problem characteristics that differentiate between the types of problems. The clique approach to measuring homo-

SPC	Homogeneous Subset							Problems
	1	2	3	4	5	6	7	
`SAT_EPR_CNF`	0.00	0.00	0.00	0.00	0.00	0.86	0.00	139
`SAT_EPR_FOF`	0.00	0.00	0.00	0.00	0.99	0.00	0.00	83
`SAT_RFO_CNF`	0.03	0.00	0.00	0.00	0.00	0.57	0.00	88
`SAT_RFO_FOF`	0.00	0.00	0.00	0.00	1.00	0.00	0.00	9
`THM_EPR_CNF`	0.81	0.00	0.00	0.00	0.00	0.00	0.15	401
`THM_EPR_FOF`	0.00	0.00	0.00	0.00	1.00	0.00	0.00	235
`THM_RFO_EQU_FOF`	0.00	0.00	0.00	0.00	0.96	0.00	0.00	323
`THM_RFO_NEQ_FOF`	0.00	0.00	0.00	0.00	1.00	0.00	0.00	21
`THM_RFO_NEQ_CNF_HRN`	0.81	0.08	0.00	0.03	0.00	0.01	0.01	379
`THM_RFO_NEQ_CNF_NHN`	0.77	0.08	0.00	0.06	0.00	0.00	0.01	430
`THM_RFO_SEQ_CNF_HRN`	0.78	0.00	0.00	0.13	0.00	0.00	0.03	381
`THM_RFO_SEQ_CNF_NHN`	0.88	0.05	0.00	0.02	0.00	0.00	0.02	1194
`THM_RFO_PEQ_CNF_NUE`	0.80	0.03	0.00	0.04	0.00	0.00	0.05	122
`THM_RFO_PEQ_CNF_UEQ`	0.00	0.00	0.85	0.00	0.00	0.00	0.00	424
Problems	2418	128	359	119	656	174	118	4229

Table 3: Correlation between Problem Cliques and SPCs

geneity, described in Section 2, assigns graph nodes, and hence performance vectors and problems, to cliques. This process can be considered to be a problem classification process, with cliques representing classes. Taking this viewpoint, machine learning (ML) techniques can be used to obtain a classifier based on problem characteristics. The classifier then provides the information needed to differentiate between types of problems. There are many ML techniques dealing with supervised classification, but not all of them are suitable for this purpose. When determining the SPCs for ATP system and problem evaluation, it is important that the classifier be in a comprehensible form, so that it is intuitive to ATP researchers and users. Conventional *decision-tree* algorithms appear to be the best choice, since the way they perform classification is easily presentable to and understandable by a human reader. `C4.5` [Qui93] is one of the most popular classification systems based on decision trees, and has been used here.

Classification methods in general, and `C4.5` in particular, require that a feature vector be associated with each object to be classified. Each feature captures a certain property of the objects, and expresses that property with a numerical value.[3] Given a set of features $a_1, \ldots, a_m$, each problem P is then associated with its feature vector $(a_1(P), \ldots, a_m(P))$. The features used to differentiate between types of problems are syntactic problem characteristics, which are supplied with each TPTP problem. They are different for CNF and FOF problems, and are listed in Table 4.

The use of a classifier provides useful information when examining apparently heterogeneous SPCs. For example, Figure 5 shows the output of `C4.5` that suggests that the SPC `SAT_RFO_CNF` can be split on equality characteristics, as concluded in Section 3.1.

[3]Other types of values can also be used, but numerical (integer) values are sufficient here.

CNF		FOF	
Cls	Number of clauses	Form	Number of formulae
NHn	Number of non-Horn clauses	Unit	Number of unit formulae
Unit	Number of unit clauses	Atom	Number of atoms
RR	Number of range restricted cls	EqA	Number of equality atoms
Lits	Number of literals	FD	Maximal formula depth
EqL	Number of equality literals	AvFD	Average formula depth
Sz	Maximal literals in a clause	Pred	Number of predicates
Av	Average literals in the clauses	Prop	Number of propositions
Pred	Number of predicates	MinPArity	Minimal predicate arity
Prop	Number of propositions	MaxPArity	Maximal predicate arity
MinPArity	Minimal predicate arity	Conn	Number of connectives
MaxPArity	Maximal predicate arity	Not	Number of ~
Vars	Number of variables	Or	Number of |
Sgn	Number of singletons	And	Number of &
Dp	Maximal term depth	Equivalent	Number of <=>
AvDp	Average term depth	Implies	Number of =>
Func	Number of functors	ImpliedBy	Number of <=
Cnst	Number of constants	XOr	Number of <~>
MinFArity	Minimal functor arity	NOr	Number of ~|
MaxFArity	Maximal functor arity	NAnd	Number of ~&
Cls==Unit	1 if Cls == Unit, else 0	Vars	Number of variables
Cls==RR	1 if Cls == RR, else 0	Sgn	Number of singletons
Cls==NHn	1 if Cls == NHn, else 0	Forall	Number of !
Cls==EqL	1 if Cls == EqL, else 0	Exists	Number of ?
Lits==EqL	1 if Lits == EqL, else 0	Dp	Maximal term depth
Pred==Prop	1 if Pred == Prop, else 0	AvDp	Average term depth
Func==Cnst	1 if Func == Cnst, else 0	Func	Number of functors
		Cnst	Number of constants
		MinFArity	Minimal functor arity
		MaxFArity	Maximal functor arity

Table 4: Features associated with problems in CNF or FOF.

Figure 6 shows the output of C4.5, highlighting the role played by constants in the 24 problem node of the second clique found in the SPC THM_RFO_NEQ_CNF_HRN, as also discussed in Section 3.1.

Figure 5: Differentiating in SAT_RFO_CNF

```
Simplified Decision Tree:

Lits==EqL = 0: CL01 (43.0)
Lits==EqL = 1:
|   Dp <= 2 : CL01 (9.0)
|   Dp > 2 :
|   |   Sz <= 4 : CL02 (22.0/2.0)
|   |   Sz > 4 : CL01 (2.0)
```

Figure 6: Differentiating in THM_RFO_NEQ_CNF_HRN

```
Simplified Decision Tree:

Unit <= 15 : CL01 (335.0/31.2)
Unit > 15 : CL02 (27.0/6.0)
```

Another use is to apply the classifier to the cliques generated from the TPTP, as described in Section 3.2. Figure 7 shows the differentiation between the 2043 problems in the first two cliques (1702 and 341 problems respectively) generated from the 3331 unsatisfiable CNF problems in the TPTP. The output clearly separates the unit equality problems (clique CL02) from the others (clique CL01).

5 Conclusion

This report describes how the homogeneity of sets of ATP problems can be measured with respect to the performance of ATP systems. The method developed assigns problems to nodes in a graph, and finding homogeneous sets of problems is reduced to finding maximal cliques in the graph. An approximation algorithm for finding the maximal cliques has proved satisfactory, thus overcoming the NP-completeness of finding maximal cliques in general. Some extensions to the basic idea have made the measurement robust to the realities of empirical data collection. In addition, a machine learning approach has been used to differentiate between types of problem in situations where heterogeneity is apparent.

The techniques developed are important, as they can be used to check the homogeneity of the Specialist Problem Classes (SPCs) used as a basis for system and problem

Figure 7: Differentiating unsatisfiable CNF problems

```
Simplified Decision Tree:

Pred > 1 : CL01 (1382.0/1.4)
Pred <= 1 :
|   EqL <= 0 : CL01 (227.0/1.4)
|   EqL > 0 :
|   |   Lits==EqL = 0: CL01 (21.0/1.3)
|   |   Lits==EqL = 1:
|   |   |   Cnst <= 5 :
|   |   |   |   Sgn <= 2 :
|   |   |   |   |   RR <= 6 : CL02 (322.0/9.6)
|   |   |   |   |   RR > 6 :
|   |   |   |   |   |   MaxFArity > 2 : CL01 (6.0/1.2)
|   |   |   |   |   |   MaxFArity <= 2 :
|   |   |   |   |   |   |   Func > 7 : CL02 (8.0/1.3)
|   |   |   |   |   |   |   Func <= 7 :
|   |   |   |   |   |   |   |   Dp <= 5 : CL01 (8.0/1.3)
|   |   |   |   |   |   |   |   Dp > 5 : CL02 (2.0/1.0)
|   |   |   |   Sgn > 2 :
|   |   |   |   |   Sgn > 5 : CL02 (5.0/1.2)
|   |   |   |   |   Sgn <= 5 :
|   |   |   |   |   |   Dp <= 3 : CL02 (2.0/1.0)
|   |   |   |   |   |   Dp > 3 : CL01 (14.0/1.3)
|   |   |   Cnst > 5 :
|   |   |   |   Vars <= 24 : CL02 (7.0/1.3)
|   |   |   |   Vars > 24 :
|   |   |   |   |   Cls <= 23 : CL01 (36.0/1.4)
|   |   |   |   |   Cls > 23 : CL02 (3.0/2.1)
```

evaluation using the TPTP. The testing done shows that the SPCs are apparently almost all highly homogeneous. In the exceptional case of the SPC `SAT_RFO_CNF`, where heterogeneity is apparent, equality has been identified as the problem characteristic that differentiates between the types of problems. As a result the SPCs can now be refined to take this into account.

References

[Hay98] T. Haynes. Perturbing the Representation, Decoding, and Evaluation of Chromosomes. In J. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. Fogel, M. Garzon, D. Goldberg, H. Iba, and R. Riolo, editors, *Proceedings of* *the 3rd Annual Conference on Genetic Programming*, pages 122–127. Morgan Kaufmann, 1998.

[Meh84] K. Mehlhorn. *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. Monographs on Theoretical Computer Science. Springer-Verlag, 1984.

[Qui93] R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1993.

[SS98] G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[SS00] G. Sutcliffe and C.B. Suttner. Evaluating General Purpose Automated Theorem Proving Systems. Technical Report 2000/2, School of Information Technology, James Cook University, Townsville, Australia, 2000.