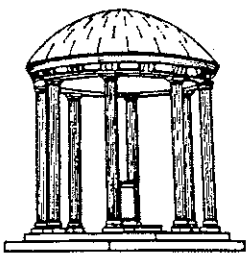
Some Experimental Results on Dynamic Subgoal Reordering

TR87-027

September,1987

Xumin Nie, David Plaisted



The University of North Carolina at Chapel Hill
Department of Computer Science
Sitterson Hall, 083A
Chapel Hill, NC 27514

Some Experimental Results on Dynamic Subgoal Reordering

Xumin Nie
David A. Plaisted
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27514

Abstract. In this paper, we report our investigations into the effect of dynamic subgoal reordering on the performance of a natural deduction theorem prover, based on some simple syntactic characteristics of the subgoals such as size, number of variables, function symbols and predicate symbols. It is shown that these simple characteristics, when used as heuristics for reordering subgoals, have a considerable impact on the overall performance of the prover on a large group of examples. The merit of our approach seems to be that we are considering the syntactic aspect of theorem proving. This aspect is simple in form and imposes low overhead in its evaluation; and it often provides good heuristics to guide a theorem prover, in spite of its simplicity and apparent lack of intelligence.

This research was supported in part by the National Science Foundation under grant DCR - 8516243.

1. Introduction

Natural deduction is one of the main approaches in mechanical theorem proving. Natural deduction systems, sometimes known as goal oriented systems or Gentzen type systems, represent similar problem solving frameworks. On one hand, these systems all have the goal-subgoal structure. That is, goals are expressed in terms of subgoals. As a consequence, the solutions for a goal are composed of the solutions for the subgoals into which the original goal is decomposed. On the other hand, the order in which the subgoals for one goal are solved often does not affect the solvability of the goal in these systems; that is, this order does not affect the completeness of these systems. One of the advantages of natural deduction systems is that it is easy to incorporate heuristic considerations with these systems. One important kind of heuristic considerations, for example, is to detect unachievable subgoals by semantic test[8]. Another heuristic considerations is to choose the "best" subgoal to work on first if there are a set of subgoals belonging to one goal and the order in which these subgoals are solved does not affect the solvability of the goal. In this paper, we will describe and discuss our research and results on this aspect of heuristic considerations -- selecting the "best" subgoal among a set of subgoals.

We will first introduce the theorem prover we use in section 2, emphasizing the formal underlying deduction system. In section 3, we will describe the search strategy the theorem prover exploits to implement the deduction system and the manner in which we incorporate our heuristic considerations for selecting the "best" subgoal into the search strategy. Section 4 will contain the discussion of our heuristics. Several evaluation functions will be introduced. Section 5 will give the results of our experiments. Some final comments will be given in section 6.

2. Modified Problem Reduction Format

The theorem prover we use is based on a natural deduction system in first order logic. This theorem prover, which will be referred to as **sprfn**, is an implementation of a modification of the theorem proving strategy described in [1]. Before we present the formal deduction system, we would like to provide some flavor of the structure of the Problem Reduction Format. The formal deduction system is a refinement of

the problem reduction format. Both of them embody the same goal-subgoal structure, as can be seen from what follows. Notice that we omit a lot of subtle details. For a complete discussion of the problem reduction format, see [4]. The following is a simplified description about problem reduction format adapted from [4].

The structure of the problem reduction format is as follows. One begins with a conclusion G to be established and a collection of assertions presumed to be true. Assertions are of the form C:-A_1,A_2, . . . ,A_n (implication) or P (premises) where A_i, P and C are literals or negation of literals. The implication assertion is understood to mean $A_1 \& A_2 \cdots \& A_n \rightarrow C$. The A_i's are antecedent statements, or simply antecedents, and C is the consequent of the implication. We call the conclusion G the top goal. The process of attempting to confirm the conclusion begins with a search of the premises to see if one premise matches (identical to G or can be made identical by unification) the goal G. If a premise P_g matches G then the conclusion is confirmed by P_g. Otherwise, the set of implications whose consequent matches G is found. If the antecedents of one implication can be confirmed then one has confirmed the consequent, and hence G which the consequent matches. Otherwise we consider the antecedents as new goals to be confirmed, one implication at a time. These goals are called subgoals because they are not the primary goal. Attempt will be made to confirm the subgoals one by one by repeating the method described above to confirm the top goal.

The deduction system underlying sprfn -- the modified problem reduction format -- embodies the same structure, although some refinements are added for the sake of completeness of the deduction system. We do not have room to describe these refinements. The following description on the modified problem reduction format is given for the completeness of our presentation. For a complete discussion, see[6].

A **clause** is a disjunction of literals. A **Horn-like clause**, converted from a clause, is of the form L :-$L_1, L_2, \ldots, L_n$ where L and L_i's are literals. L is called the head literal. L_i's constitute the clause body. A clause is converted to a Horn-like clause as follows. For each clause, one of the positive literals is chosen as the head literal and all other literals are negated and put in the clause body. For an all-negative clause, we use **false** as the head literal. Assume S is a set of Horn-like clauses. A set of inference rules

from S is obtained as follows. For each clause $L :- L_1, L_2, \ldots, L_n$ in S, we have the following clause rule:

Clause Rules

$$\frac{\Gamma_0 \rightarrow L_1 \Rightarrow \Gamma_1 \rightarrow L_1, \Gamma_1 \rightarrow L_2 \Rightarrow \Gamma_2 \rightarrow L_2, \ldots, \Gamma_{n-1} \rightarrow L_n \Rightarrow \Gamma_n \rightarrow L_n}{\Gamma_0 \rightarrow L \Rightarrow \Gamma_n \rightarrow L}$$

We also have assumption axioms and case analysis (splitting) rule. In the assumption axioms, L is a positive literal.

Assumption Axioms

$$\Gamma \rightarrow L \Rightarrow \Gamma \rightarrow L \quad \text{if } L \in \Gamma$$

$$\Gamma \rightarrow \bar{L} \Rightarrow \Gamma, \bar{L} \rightarrow \bar{L}$$

Case Analysis (splitting) Rule

$$\frac{\Gamma_0 \rightarrow L \Rightarrow \Gamma_1, \bar{M} \rightarrow L, \ \Gamma_1, M \rightarrow L \Rightarrow \Gamma_1, M \rightarrow L}{\Gamma_0 \rightarrow L \Rightarrow \Gamma_1 \rightarrow L}$$

The goal-subgoal structure of this deduction system is evident. The input clause $L :- L_1, L_2, \ldots, L_n$ merely states that $L_1, L_2, \ldots, L_n$ have to be confirmed in order to confirm L. The corresponding clause rule for $L :- L_1, L_2, \ldots, L_n$ states that, if the initial subgoal is $\Gamma \rightarrow L$, then make $L_1, \ldots, L_n$ subgoals in succession; add to Γ successively the literals that are needed to make each subgoal provable; finally, return $\Gamma_n \rightarrow L$ where Γ_n contains all the literals needed to make $L_1, \ldots, L_n$ provable. The assumption lists Γ's just represent the refinement of the modified problem reduction format to the problem reduction format. None of the listing of subgoals in the input clause and the clause rule implies any particular order in which the subgoals should be attempted. Because no order is implied, when $\Gamma_0 \rightarrow L$ is attempted and the inference rule corresponding to the input clause $L :- L_1, L_2, \ldots, L_n$ is invoked, we can exercise our intelligence to choose a L_1' among the n literals $L_1, L_2, \ldots, L_n$ so that $\Gamma_0 \rightarrow L_1'$ would be the first subgoal to be attempted. After $\Gamma_0 \rightarrow L_1'$ returns $\Gamma_1 \rightarrow L_1'$, L_2' could be chosen among the remaining n-1 literals so that $\Gamma_1 \rightarrow L_2'$ would be the second subgoal, and so forth. Thus, each time the inference rule for the same input clause $L :- L_1, L_2, \ldots, L_n$ is invoked, a possibly different sequence of the literals $L_1, L_2, \ldots, L_n$ would be used.

This natural observation above leads us to ask how do we choose the "best" subgoal at any moment, i.e., what heuristics should we use to perform the selection.

We will discuss our experiments motivated by the above observation. Before we do this, however, we need to discuss the search strategy sprfn exploits. We will also discuss the manner in which the heuristics for selecting the best subgoal are applied. The implication of applying the heuristics on the search strategy is also discussed.

3. Search Strategy and Subgoal Reordering

Sprfn exploits the Prolog-style depth-first iterative deepening search. This search strategy involves repeatedly performing exhaustive depth-first search with increasing depth bounds. This complete search strategy can be efficiently implemented in Prolog, taking advantage of the Prolog built-in depth-first search with backtracking. This search strategy also requires much less memory in comparison with the other two search strategies -- breadth-first search and A^* search. A detailed analysis of the depth-first iterative deepening search strategy can be found in [2].

To apply the heuristics to select the best subgoal among a set of subgoals for one goal, sprfn will work as follows. An evaluation function will be defined. This function embodies our heuristics for measuring the "quality" of an each subgoal from a set of subgoals and selecting the best one from the set. Sprfn starts with the top-level goal. For any goal $\Gamma_0 \rightarrow L$, whenever an inference rule corresponding to the input clause $L :- L_1, L_2, \ldots, L_n$ is invoked, sprfn uses the evaluation function to select L_1' among the n literals $L_1, L_2, \ldots, L_n$ and attempts $\Gamma_0 \rightarrow L_1'$. After $\Gamma_0 \rightarrow L_1'$ returns with $\Gamma_1 \rightarrow L_1'$, the evaluation function is used again to select L_2' among the remaining n-1 literals, etc. This process will dynamically order the subgoals for one goal during the search process. We call this process **subgoal reordering**. We emphasize that the above is a recursive process and involves every invocation of a clause rule. We also emphasize that the selection is made only among the remaining subgoals belonging to one common goal, i.e., the choice is only the locally best according to the evaluation function.

It is appropriate to note now that the process of subgoal reordering outlined above preserves the completeness and the soundness of the modified problem reduction format.

We would like to point out that our depth-first iterative deepening search strategy with subgoal reordering as described above is different from iterative-deepening A^* search in [3]. In our strategy, the selection is made among a limited set of subgoals which belong to one common goal. In iterative deepening A^*, search, a queue containing all the subgoals awaiting to be attempted is maintained and the best is selected from this global queue. Another difference comes from the nature of the heuristics. As a consequence, the design of the evaluation functions are very different. In iterative-deepening A^* search, the evaluation function has to take into account the cost in reaching a particular subgoal from the top-level goal and the estimates of the cost for solving that subgoal. Thus the heuristics are global heuristics. In our strategy, the selection is made locally, the cost in reaching every subgoal in question would be the same, so it is not considered by the evaluation function. Since we are interested in the effect of syntactic properties of subgoals, the evaluation functions are designed to consider this aspect more than anything else. Of course, the evaluation functions could be designed based on some other heuristics.

A similar process called *level subgoal reordering* is implemented in [7]. This process is incorporated in a goal-oriented theorem prover based on a deduction procedure called hierarchical deduction procedure. The idea is to select the first literal to resolve upon in a goal clause. In [7], the evaluation function for selecting the first literal is based on the concepts of **twin symbols** and **mass** of function and predicate symbols. This evaluation function, by the way, also considers the syntactic aspect of literals, although probably takes a more global measurement than our evaluation functions. In [7], level subgoal reordering is applied locally, within a goal clause. This is also similar to our method.

4. Evaluation Functions

The evaluation functions are designed to measure the "quality" of subgoals based on certain heuristics. As can be seen from the above discussion, it is going to be a frequent activity for sprfn to apply the evaluation functions to a set of subgoals to select the best subgoal if subgoal reordering is incorporated in

it. This requires that the application of evaluation functions impose low overhead. However, evaluation functions must be designed to improve the efficiency of sprfn. The low cost of evaluation function application requires that the quality of subgoals be measured based on some simple characteristics about subgoals. A positive contribution to the performance of sprfn by subgoal reordering can be achieved if the evaluation functions can make sprfn concentrate on the important subgoals or reduce the branching factors of the search space. Based on these considerations, we investigated the following evaluation functions by experimenting on a large number of examples.

[1] **The size of subgoals.** The size of subgoals is the number of occurrences of predicate symbols, function symbols and variables. Larger subgoals can be regarded as more important since they may contain more useful variable bindings. Also, the larger size imposes more constraints on unifications so the branching factors for larger subgoals tend to be smaller.

[2] **The number of distinct variables in subgoals.** Since variables contribute to the size of subgoals, the same reasoning applied to the size of subgoals can be applied here. However, the number of distinct variables may contribute more to the branching factors of the search space since more can be unified with variables.

[3] **The ground size of subgoals.** The ground size of subgoals is the number of occurrences of predicate symbols and function symbols. Note that the occurrences of variables are not counted. It is easy to see that the ground size of subgoals affects the search process in the similar way as the size of subgoals does.

[4] **The number of solutions with the same predicate symbols as the subgoal does.** The evaluation function based on this takes a more global measurement of the quality of subgoals. If the number of solutions in question is small, the branching factor will probably be small also.

[5] **The number of solutions unifiable with the subgoal.** This measurement is a refinement of the measurement above, as can be easily seen.

Accordingly, we use the following evaluation functions to select the best subgoal. These functions will be referred to as $function_1$, $function_2$, $function_3$, $function_4$ and $function_5$ respectively.

[1] Select the subgoal with the largest size. In case of a tie, the order in the input is preserved.

[2] Select the subgoal with the largest number of distinct variables. In case of a tie, the subgoal with the largest ground size is chosen. If there is still a tie, the order in the input is preserved.

[3] Select the subgoal with the largest ground size. The subgoal with the largest size is selected. In case of a tie, the order in the input is preserved.

[4] Select the subgoal with the least number of solutions with the same predicate symbol as the subgoal. If several subgoals have the same predicate or the numbers of solutions for each several subgoals are the same, the subgoal with the largest ground size is selected. If there is still a tie, the order in the input is preserved.

[5] Select the subgoal with the least number of solutions which are unifiable with this subgoal. If several subgoals have the same number of solutions unifiable with them, again, the subgoal with the largest ground size is selected. If there is still a tie, the order in the input is preserved. We use Prolog built-in unification to save time.

In the next section, we will give the results of our experiments using the above five evaluation functions.

5. Implementation and Results

A nice and convenient Prolog interface in sprfn provides an easy vehicle to carry out our experiment. In the input to sprfn, a subgoal of the form prolog(L) represents a call to the Prolog procedure L. We write a Prolog subroutine, called *best_subgoal*, to select the best subgoal among a list of subgoals. Another Prolog subroutine is written to translate the standard input format into the format which includes the calls to the Prolog subroutine *best_subgoal*. For example, the input clause

$$L :- L_1,\ L_2,\ L_3$$

is translated into

$$L :- prolog\,(best\_subgoal\,([L_1,L_2,L_3],[X_1 \mid Y])),\ X_1,$$

$$prolog\,(best\_subgoal\,(Y,[X_2,X_3])),X_2,X_3.$$

This is further translated into the internal representation of input clauses used by sprfn.

We performed tests on 56 examples from different sources, using the five evaluation functions in the last section. Performance statistics of sprfn are given in the appendix. We would like to point out that none of the five evaluation functions makes sprfn perform better on all the examples when used for subgoal reordering. Nevertheless, some good heuristics are suggested by these data.

Larger subgoals should be preferred. This heuristic is suggested by the data for $function_1$ and $function_3$. Using these two functions, we achieve overall improvements both in speed and in terms of inferences. One reason for the improvements is that the search space for a larger subgoal has potentially smaller branching factors since the larger number of symbols impose more constraints on unification. Another reason seems to be that the size of a subgoal is a good indicator of the "importance" and "relevance" of the subgoal. If a subgoal is larger, it contains more information about the current problem, thus more important. Yet another reason for the improvements seems to have something to do with the implementation of this particular prover. In sprfn, the size of the old solutions used constitutes part of the effort to derive a subgoal. The reason for doing this is to avoid the possible uncontrolled expansion of the search space due to a large number of big subgoals. If a goal is not solvable because its subgoals are too large, selecting the larger subgoals over smaller ones makes sprfn stop the search along the branch for this unsolvable subgoal earlier, thus reducing the fruitless search effort along this dead-end branch. However, we will not have room here to explain this further.

Branches with smaller branching factors should be preferred. This is similar to the general heuristics that one should do the easier things first. This heuristics is suggested by the data for $function_4$ and $function_5$. These two evaluation functions are designed to direct the prover to prefer the branches with the smaller branching factors, using the number of solutions as the indicator of the potential branching fac-

tors. It improves sprfn's performance on a large number of examples in terms of inferences, to use these two evaluation functions for subgoal reordering. It is very interesting to note that sprfn performs roughly the same using $function_4$ and $function_5$. But it imposes much more overhead to evaluate $function_5$ than to evaluate $function_4$. We would like to mention that this is very representative of our experience so far with the research on sprfn. It is often the case that simple things provide the most dramatic improvements.

One interesting observation deserves mentioning. We note that, although the performance of sprfn changes considerably on some examples when subgoal reordering is performed, the default prover always perform better or as well in the average ratios. This can be seen from the data in Table 0 in the appendix. The reason seems to be that we have been careful in ordering the input clauses when we enter them by hand. The order thus obtained is probably "optimal" since we have a reasonably good idea about what these problems are. The best we can hope for subgoal reordering to accomplish is to approximate the "goodness" of hand ordering of the input clauses. We think our heuristics for subgoal reordering have achieved this nicely.

6. Conclusion

The heuristics suggested by our experiments are by no means new. But there are two things which are interesting about our investigations. First, the method of incorporating subgoal reordering into the depth-first iterative deepening search strategy is very simple. One of the advantages of the depth-first iterative deepening search strategy is its minimal requirements for memory. Our subgoal reordering scheme preserves this advantage. Second, the evaluation functions used for subgoal reordering are all very simple. They are almost purely syntactic in nature. However, their impact on the performance of this prover is considerable. The merit of using the evaluation functions like these seems to lie in the fact we are considering the syntactic aspect of the problems. This aspect is simple in form and imposes low overhead in its evaluation; and it often provides good heuristics. In general, we think that the importance of the syntactic aspect in mechanical theorem proving is not to be ignored, although it may not play a decisive role in the success of this field.

References

[1] Plaisted, D.A. <<A simplified Problem Reduction Format>> Artificial Intelligence 18 (1982) 227-261

[2] Stickel, M.E & Tyson, W.M. <<An Analysis of Consecutively Bounded Depth-First Search with Applications in Automated Deduction>> IJCAI 1983, 1073-1075

[3] Korf, R.E. <<Iterative-Deepening-A*: An Optimal Admissible Tree Search>> IJCAI 1985, 1034-1036

[4] Loveland, D.W. <<Automated Theorem Proving: A Logical Base>> Chapter 6. North-Holland Publishing Company, 1978

[5] Plaisted, D.A <<Another Extension of Horn Clause Logic Programming to Non-horn Clauses>> lecture notes, 1987

[6] Plaisted, D.A <<Non-Horn Clause Logic Programming Without Contrapositives>> unpublished manuscript, 1987

[7] Wang, T.C, <<Hierarchical Deduction>>, 1987

[8] Reiter, R <<A Semantically Guided Deductive System for Automatic Theorem Proving>> IEEE Trans. Elec. Comput. 25 (1976) 328-334.

Appendix -- Performance Statistics of Sprfn for Subgoal Reordering

This section contains the statistics of sprfn's performance using the different evaluation functions for subgoal reordering. Table 0 contains some summary and comparative data. Table 1-5 contain the statistics of sprfn's performance on each individual problem . Each of these five tables is for one of the five evaluation functions defined in section 4. In table 1-5, sprfn's performance data without subgoal reordering are listed on the left for easy comparison. Note that in each of these five tables, two sets of data are listed. The set on the left is obtained using each of the evaluation functions given in section 4. The set on the right is obtained using the same function, but the value is negated. Thus the ordering used would be reversed. For instance, there are two sets of data in table 3 which is for *function* $_3$. The data set on the left is obtained by selecting the subgoal with the largest ground size. The data set on the right is obtained by selecting the subgoal with the smallest ground size. Data in table 0 are calculated from the statistics in table 1-5. There are two parts in table 0. Each part has two rows for each evaluation function. The upper row contains the data for the original evaluation function; the lower row contains the data for the negated evaluation function. Note that the statistics for wos1 and wos10 are not used when calculating the data for table 0.

Table 0. Summary and Comparative Data (part I)

	Average Time Per Theorem(sec.)		Average Inferences Per Theorem	
	absolute	Average Ratio with default	absolute	Average Ratio with default
function1	285.78	1.10	654.9	1.03
	511.81	2.62	1020.1	2.03
function2	376.09	1.96	787.1	1.43
	572.73	7.87	988.5	2.58
function3	295.28	1.16	666.9	1.07
	615.14	9.67	1245.5	3.18
function4	348.19	1.15	633.3	0.99
	460.39	3.21	1144.4	2.90
function5	366.89	1.21	638.3	1.00
	486.46	3.51	1183.8	2.86
default	329.77	1.00	696.8	1.00

In this part, two items under **Average Time Per Theorem** are calculated using the following two formulae:

$$absolute = \frac{\sum_{i=1}^{i=N} T_{ji}}{N}$$

$$average\ ratio\ with\ default = \frac{\sum_{i=1}^{i=N} \frac{T_{ji}}{T_{di}}}{N}$$

where T_{ji} is the cputime taken by sprfn to prove the i^{th} theorem using evaluation function $function_j$. T_{di} is the cputime taken by sprfn to prove the i^{th} theorem without subgoal reordering. N is the number of theorems we tested. The two items under **Average Inferences Per Theorem** are calculated using the similar formulae. But the number of inferences for each theorem is used instead.

Table 0. Summary and Comparative Data (part II).

functions	Better Example		Worse Example		Even Example
	number	percent(%)	number	percent(%)	number
function1	15	12.0	16	21.9	25
	14	35.8	18	348.0	24
function2	8	16.0	30	84.3	18
	25	25.2	14	678.8	17
function3	24	12.2	12	57.5	20
	10	32.3	27	464.0	19
function4	27	20.9	10	51.0	19
	7	19.6	36	299.8	13
function5	31	18.7	12	46.9	13
	10	23.3	32	332.9	14
default	0	0	0	0	56

The data in this part concern solely with the number of inferences. There are two items under **better example**. The item under number is the number of theorems on which sprfn performs better using the corresponding evaluation function for subgoal reordering. The item under **percent** is the average speedup of sprfn with subgoal reordering using this evaluation function relative to the default prover for these problems. For example, two numbers under **better example** for $function_1$ are 15 and 12.0. They mean that sprfn performs better on 15 theorems using $function_1$ for subgoal reordering; the average speed up for

these 15 theorems is 12.0% relative to the default prover. There are also two items under **worse example**. Similarly, the item under **number** is the number of theorems on which sprfn performs worse using the corresponding evaluation function for subgoal reordering. The item under **percent** is the average slow-down of sprfn with subgoal reordering using this evaluation function relative to the default prover. The item under **even example** is the number of theorems on which the default prover and the prover with subgoal reordering using the evaluation function perform equally well.

Table 1. data when using size to reorder subgoals

theorem	data for default		data for function1			
			largest first		smallest first	
	cputime	inference	cputime	inference	cputime	inference
ances1	12.38	27	15.38	27	13.13	27
burstall	6.90	63	8.12	63	7.47	63
dbabhp	26.40	190	26.75	192	83.53	545
dm	1.15	6	0.95	5	1.63	8
ew1	1.93	6	2.12	6	2.08	6
ew2	1.27	5	1.45	5	1.47	5
ew3	4.62	15	5.75	15	5.45	15
ex1	1.13	6	1.00	5	1.63	8
ex2	19.32	259	21.00	259	20.60	259
ex3	3.30	29	3.70	28	3.77	29
ex4	3.45	30	3.72	29	3.87	30
ex5	0.55	4	0.60	4	0.58	4
ex6	14.18	157	15.20	156	14.98	157
ex7	2.55	16	2.77	16	3.20	18
ex8	9.62	64	10.45	64	9.05	50
ex9	12.87	40	13.55	40	10.07	25
example	45.78	236	47.68	236	47.57	236
fex4t2	850.03	1002	855.45	1002	340.98	337
group1	1.13	6	1.07	5	1.70	8
group2	19.27	259	21.58	259	20.68	259
hasparts1	4.85	28	5.13	19	5.52	29
hasparts2	9.37	54	9.45	37	10.73	54
ls100	0.68	4	0.75	4	0.75	4
ls103	30.60	131	54.86	239	34.63	135
ls105	1.10	5	1.10	6	1.27	5
ls106	1.12	5	1.03	6	1.27	5
ls111	1.10	5	1.15	7	1.25	5
ls115	53.50	207	63.00	201	3251.63	9736
ls17	16.53	76	17.83	80	17.57	76
ls23	41.47	236	48.25	255	69.86	413
ls26	34.60	199	34.33	198	36.22	203
ls28	714.80	1117	741.15	1117	414.68	768
ls29	577.75	941	585.50	941	348.32	664
ls35	41.82	336	42.92	320	53.80	368
ls41	12.98	80	14.37	86	10.95	49
ls5	1.93	5	2.07	5	2.03	5
ls55	50.60	369	57.42	406	37.00	147
ls68	47.87	291	55.88	322	25.05	95
mqw	2.13	5	2.35	5	2.33	5
num1	2.50	17	3.05	17	3.20	18
prim	5.12	32	8.75	46	6.05	31
qw	3.25	11	3.62	11	3.58	11
rob1	1.08	3	1.37	3	1.38	3
rob2	19.58	259	21.20	259	20.92	259
schubert.abst	202.98	953	320.27	1374	281.34	1228
shortburst	3.50	24	4.02	24	3.90	25
wos11	1502.30	4901	1416.43	4526	2620.66	6166
wos12	3.23	36	3.48	36	1.78	11
wos13	1671.12	3969	1672.12	3971	2281.80	3763
wos14	4440.81	7208	1700.60	4107	2181.30	3631
wos2	72.64	443	72.87	446	264.05	1487
wos3	1.22	9	1.32	9	1.32	9
wos6	5542.98	7896	5542.88	7889	7409.25	6939
wos7	229.12	951	343.58	1493	3816.08	6078
wos8	1694.97	4120	1693.85	4109	2228.57	3758
wos9	388.03	1676	397.48	1684	2618.14	8855
wos1	169.17	683	170.14	702	failure	
wos10	821.992	2878	865.61	2921	failure	

Table 2. data when using number of distinct variables to reorder subgoals

theorem	data for default		data for function2			
			largest first		smallest first	
	cputime	inference	cputime	inference	cputime	inference
ances1	12.38	27	13.33	27	13.68	27
burstall	6.90	63	7.33	63	7.40	63
dbabhp	26.40	190	173.82	894	8401.27	8750
dm	1.15	6	1.03	5	1.77	8
ew1	1.93	6	1.97	6	1.93	6
ew2	1.27	5	1.35	5	1.37	5
ew3	4.62	15	5.05	15	5.03	15
ex1	1.13	6	1.05	5	1.78	8
ex2	19.32	259	25.95	343	19.32	231
ex3	3.30	29	3.90	28	3.75	25
ex4	3.45	30	6.05	45	3.98	27
ex5	0.55	4	0.60	4	0.57	4
ex6	14.18	157	17.03	181	14.42	145
ex7	2.55	16	2.75	16	3.03	18
ex8	9.62	64	12.32	84	9.10	50
ex9	12.87	40	13.42	40	10.03	25
example	45.78	236	49.20	253	62.95	286
fex4t2	850.03	1002	855.45	1002	341.86	337
group1	1.13	6	1.07	5	1.80	8
group2	19.27	259	26.00	343	19.30	231
hasparts1	4.85	28	5.07	19	5.70	28
hasparts2	9.37	54	9.10	37	10.85	54
ls100	0.68	4	0.70	4	0.72	4
ls103	30.60	131	52.35	217	35.98	126
ls105	1.10	5	0.97	5	1.27	5
ls106	1.12	5	1.02	5	1.32	5
ls111	1.10	5	0.95	5	1.40	6
ls115	53.50	207	2174.02	3244	3205.47	9294
ls17	16.53	76	18.12	80	17.65	76
ls23	41.47	236	60.84	344	22.42	123
ls26	34.60	199	36.37	219	35.86	197
ls28	714.80	1117	719.00	1124	389.67	739
ls29	577.75	941	582.35	948	324.80	634
ls35	41.82	336	53.27	354	46.75	331
ls41	12.98	80	13.70	86	10.48	53
ls5	1.93	5	2.03	5	2.03	5
ls55	50.60	369	56.95	406	35.06	147
ls68	47.87	291	54.32	322	24.22	99
mqw	2.13	5	3.22	9	2.32	5
num1	2.50	17	2.78	16	3.07	18
prim	5.12	32	7.83	46	6.10	31
qw	3.25	11	5.45	24	3.73	11
rob1	1.08	3	1.65	3	1.65	3
rob2	19.58	259	26.45	343	19.50	231
schubert.abst	202.98	953	330.16	1392	187.44	847
shortburst	3.50	24	3.70	24	3.87	25
wos11	1502.30	4901	1447.41	4654	2602.84	6108
wos12	3.23	36	3.43	36	1.82	11
wos13	1671.12	3969	1674.47	4002	2166.65	3614
wos14	4440.81	7208	4435.95	7236	2342.43	3829
wos2	72.64	443	106.29	544	184.50	1076
wos3	1.22	9	1.30	9	1.30	9
wos6	5542.98	7896	5543.88	7936	7380.16	6915
wos7	229.12	951	234.94	977	686.41	1710
wos8	1694.97	4120	1695.31	4150	2214.87	3743
wos9	388.03	1676	476.98	1891	1168.36	4974
wos1	169.17	683	179.84	703	failure	
wos10	821.992	2878	1218.89	3740	failure	

Table 3. data when using ground size to reorder subgoals

theorem	data for default		data for function3			
			largest first		smallest first	
	cputime	inference	cputime	inference	cputime	inference
ances1	12.38	27	13.02	27	13.48	27
burstall	6.90	63	7.60	63	7.12	63
dbabhp	26.40	190	175.45	881	10350.90	10115
dm	1.15	6	0.97	5	1.73	8
ew1	1.93	6	2.02	6	1.95	6
ew2	1.27	5	1.38	5	1.37	5
ew3	4.62	15	5.12	15	5.03	15
ex1	1.13	6	0.98	5	1.72	8
ex2	19.32	259	19.00	231	25.58	343
ex3	3.30	29	3.40	24	3.80	29
ex4	3.45	30	3.58	26	6.67	52
ex5	0.55	4	0.60	4	0.53	4
ex6	14.18	157	14.18	144	16.73	181
ex7	2.55	16	2.75	16	2.72	16
ex8	9.62	64	11.02	64	9.00	50
ex9	12.87	40	14.05	40	10.03	25
example	45.78	236	62.88	286	48.83	253
fex4t2	850.03	1002	853.98	1002	340.60	337
group1	1.13	6	1.05	5	1.70	8
group2	19.27	259	19.08	231	25.47	343
hasparts1	4.85	28	4.73	19	5.37	28
hasparts2	9.37	54	8.60	37	10.52	54
ls100	0.68	4	0.77	4	0.72	4
ls103	30.60	131	54.70	238	34.18	131
ls105	1.10	5	1.00	6	1.22	5
ls106	1.12	5	1.05	6	1.32	5
ls111	1.10	5	1.13	7	1.22	5
ls115	53.50	207	66.82	200	4202.68	12276
ls17	16.53	76	17.73	80	17.32	76
ls23	41.47	236	22.00	121	75.77	465
ls26	34.60	199	33.68	192	37.47	222
ls28	714.80	1117	723.37	1117	416.48	768
ls29	577.75	941	578.53	941	350.23	664
ls35	41.82	336	40.90	318	55.90	376
ls41	12.98	80	13.72	86	13.65	82
ls5	1.93	5	2.03	5	2.00	5
ls55	50.60	369	50.28	370	59.37	401
ls68	47.87	291	48.23	291	56.67	319
mqw	2.13	5	2.32	5	3.20	9
num1	2.50	17	2.80	16	2.77	16
prim	5.12	32	9.13	64	5.77	31
qw	3.25	11	3.62	11	5.30	24
rob1	1.08	3	1.53	3	1.47	3
rob2	19.58	259	19.43	231	26.05	343
schubert.abst	202.98	953	299.09	1216	165.62	714
shortburst	3.50	24	3.97	24	3.72	25
wos11	1502.30	4901	1401.87	4473	2856.17	6902
wos12	3.23	36	3.52	37	1.63	9
wos13	1671.12	3969	1668.75	3959	1701.37	3998
wos14	4440.81	7208	2323.42	5203	2610.83	5226
wos2	72.64	443	71.79	442	263.20	1529
wos3	1.22	9	1.30	9	1.28	9
wos6	5542.98	7896	5537.02	7855	5550.34	7936
wos7	229.12	951	222.43	927	700.48	1985
wos8	1694.97	4120	1692.88	4097	1697.56	4130
wos9	388.03	1676	389.16	1659	2633.88	9084
wos1	169.17	683	136.03	539	failure	
wos10	821.992	2878	830.16	2924	failure	

Table 4. data when using solution with same predicate to reorder subgoals

theorem	data for default		data for function4			
			smallest first		largest first	
	cputime	inference	cputime	inference	cputime	inference
ances1	12.38	27	10.70	13	14.25	27
burstall	6.90	63	7.58	63	7.65	63
dbabhp	26.40	190	142.27	742	98.23	625
dm	1.15	6	0.98	5	1.72	8
ew1	1.93	6	2.03	6	2.03	6
ew2	1.27	5	1.43	5	1.35	5
ew3	4.62	15	5.68	24	5.22	15
ex1	1.13	6	1.00	5	1.72	8
ex2	19.32	259	18.78	231	25.50	343
ex3	3.30	29	3.47	24	3.83	29
ex4	3.45	30	3.72	26	6.62	52
ex5	0.55	4	0.58	4	0.62	4
ex6	14.18	157	14.05	143	17.62	184
ex7	2.55	16	2.80	16	3.10	19
ex8	9.62	64	8.67	49	12.33	84
ex9	12.87	40	10.10	25	13.48	40
example	45.78	236	63.15	286	49.28	253
fex4t2	850.03	1002	639.47	580	331.36	313
group1	1.13	6	0.98	5	1.77	8
group2	19.27	259	18.98	231	25.60	343
hasparts1	4.85	28	4.83	19	5.57	29
hasparts2	9.37	54	8.63	37	10.67	54
ls100	0.68	4	0.73	4	0.77	4
ls103	30.60	131	33.27	127	52.88	213
ls105	1.10	5	1.25	5	1.22	7
ls106	1.12	5	1.30	5	1.22	7
ls111	1.10	5	1.27	5	1.18	7
ls115	53.50	207	67.13	205	5725.48	18932
ls17	16.53	76	17.80	76	17.63	78
ls23	41.47	236	22.29	121	76.45	465
ls26	34.60	199	34.54	194	38.42	224
ls28	714.80	1117	719.58	1117	418.35	768
ls29	577.75	941	581.43	941	351.40	664
ls35	41.82	336	41.02	318	56.30	376
ls41	12.98	80	13.77	86	14.03	82
ls5	1.93	5	2.07	5	2.75	13
ls55	50.60	369	50.68	370	59.82	401
ls68	47.87	291	47.80	291	56.93	319
mqw	2.13	5	2.33	5	3.23	9
num1	2.50	17	2.80	16	3.10	19
prim	5.12	32	5.65	32	7.93	46
qw	3.25	11	3.72	11	5.38	24
rob1	1.08	3	1.50	3	1.40	3
rob2	19.58	259	19.03	231	25.60	343
schubert.abst	202.98	953	181.33	802	389.07	1656
shortburst	3.50	24	3.87	25	3.78	24
wos11	1502.30	4901	1942.86	5095	1748.50	5344
wos12	3.23	36	1.78	11	3.55	37
wos13	1671.12	3969	2059.90	3676	1765.50	3941
wos14	4440.81	7208	2239.72	3908	4590.96	7229
wos2	72.64	443	85.96	475	114.02	565
wos3	1.22	9	1.30	9	1.33	9
wos6	5542.98	7896	7007.35	7019	5863.30	7832
wos7	229.12	951	624.22	1765	253.57	899
wos8	1694.97	4120	2114.05	3807	1783.10	4076
wos9	388.03	1676	595.56	2167	1704.34	6958
wos1	169.17	683	209.23	863	failure	
wos10	821.992	2878	1411.15	3837	failure	

Table 5. data when using number of unifiable solutions to reorder subgoals

theorem	data for default		data for function5			
			smallest first		largest first	
	cputime	inference	cputime	inference	cputime	inference
ances1	12.38	27	14.02	26	14.38	27
burstall	6.90	63	8.32	64	7.53	64
dbabhp	26.40	190	144.77	742	100.22	625
dm	1.15	6	1.05	5	1.80	8
ew1	1.93	6	2.02	6	1.95	6
ew2	1.27	5	1.42	5	1.48	5
ew3	4.62	15	5.12	15	5.62	15
ex1	1.13	6	1.08	5	1.78	8
ex2	19.32	259	20.80	226	29.53	358
ex3	3.30	29	3.83	24	5.70	39
ex4	3.45	30	3.98	25	7.42	54
ex5	0.55	4	0.65	4	0.65	4
ex6	14.18	157	15.32	141	18.87	185
ex7	2.55	16	2.93	16	3.27	18
ex8	9.62	64	9.12	49	12.73	84
ex9	12.87	40	13.00	33	11.17	32
example	45.78	236	67.45	283	50.25	253
fex4t2	850.03	1002	650.09	580	335.72	313
group1	1.13	6	1.10	5	1.82	8
group2	19.27	259	20.90	226	29.70	358
hasparts1	4.85	28	4.80	19	5.50	28
hasparts2	9.37	54	8.77	37	11.00	54
ls100	0.68	4	0.75	4	0.63	4
ls103	30.60	131	50.43	206	39.38	154
ls105	1.10	5	1.07	6	1.25	5
ls106	1.12	5	1.08	6	1.27	5
ls111	1.10	5	1.05	6	1.27	5
ls115	53.50	207	69.54	191	6213.39	18365
ls17	16.53	76	18.33	78	18.52	76
ls23	41.47	236	24.60	117	93.05	527
ls26	34.60	199	36.68	189	40.92	226
ls28	714.80	1117	719.42	1049	454.85	830
ls29	577.75	941	585.93	903	353.77	679
ls35	41.82	336	50.65	328	58.52	367
ls41	12.98	80	9.55	50	14.02	82
ls5	1.93	5	2.10	5	2.07	5
ls55	50.60	369	35.52	217	65.08	416
ls68	47.87	291	32.35	185	74.27	407
mqw	2.13	5	2.33	5	3.20	9
num1	2.50	17	2.93	16	3.40	18
prim	5.12	32	5.93	32	8.10	46
qw	3.25	11	3.78	10	5.63	24
rob1	1.08	3	1.48	3	1.77	3
rob2	19.58	259	21.03	226	29.77	357
schubert.abst	202.98	953	301.12	1125	666.70	2183
shortburst	3.50	24	3.83	24	3.83	25
wos11	1502.30	4901	2433.96	5516	1802.22	5373
wos12	3.23	36	3.47	36	2.48	13
wos13	1671.12	3969	2069.22	3674	1757.17	3936
wos14	4440.81	7208	2316.22	4029	4537.41	7088
wos2	72.64	443	77.50	355	294.80	1592
wos3	1.22	9	1.40	9	1.05	7
wos6	5542.98	7896	7139.03	7027	5816.50	7828
wos7	229.12	951	737.52	1657	269.07	1003
wos8	1694.97	4120	2124.58	3815	1772.95	4068
wos9	388.03	1676	660.78	2107	2175.48	8020
wos1	169.17	683	146.38	491	failure	
wos10	821.992	2878	1113.63	2886	failure	