Using Association Rules to Guide a Search for Best Fitting Transfer Models of Student Learning

by

Jonathan Freyberger

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

Jonathan Freyberger

April 30, 2004

APPROVED:

Professor Neil Heffernan, Thesis Advisor

Carolina Ruiz

Professor Carolina Ruiz, Thesis Co-Advisor

Michael A Gennert

Professor Michael Gennert, Thesis Reader

Michael A Gennert

Professor Michael Gennert, Head of Department

Abstract

Transfer models provide a viable means of determining which skills a student needs in order to solve a given problem. However, constructing a good fitting transfer model requires a lot of trial and error. The main goal of this thesis was to develop a procedure for developing better fit transfer models for intelligent tutoring systems. The procedure implements a search method using association rules as a means of guiding the search. The association rules are mined from the instances in the dataset that the transfer model predicts incorrectly. The association rules found in the mining process determines what operation to perform on the current transfer model.

Our search algorithm using association rules was compared to a blind search method that finds all possible transfer models for a given set of factors. Our search process was able to find statistically similar models to the ones the blind search method finds in a considerably shorter amount of time. The difference in times between our search process and the blind search method is days to minutes. Being able to find good transfer models quicker will help intelligent tutor system builders as well as cognitive science researchers better assess what makes certain problems hard and other problems easy for students.

Acknowledgments

I would like to thank my advisors, Professor Heffernan and Professor Ruiz for their help and support. In addition, I would like to thank Professor Gennert for being my reader on short notice. I would like to thank Ethan Croteau for his help in creating datasets and help with S-Plus. I would like to thank Keith Pray for creating AprioriSetsAndSequences and for helping me with any questions I had about AprioriSetsAndSequences. I would also like to thank Larissa Orlova for creating the code base that this work built off of.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Intelligent tutoring systems offer a new means of improving education. It has been found that students benefit most from one-on-one human tutoring [B84]. Unfortunately, there are not enough skilled human tutors out there to attend to the needs of every student. Typically one teacher is responsible for instructing an average of 30 students at a time in a public school [B84]. The problem is that usually students in a class are not at the same knowledge level. Thus, some students will have an easier time learning the material in a class and others will struggle. A teacher can easily determine which students are performing well in a class and which ones are performing poorly. However, figuring out why some students are performing poorer than others can be very difficult. It is most likely that not every student is performing poorly due to the same reason. A teacher would probably have to interact one-on-one with each individual student to figure out what they are having difficulty with. Unfortunately, teachers do not always have the time and resources to address every single student's difficulties.

This is where intelligent tutoring systems come in. Recent advances in computer science combined with cognitive science's increasing understanding of how the human mind works and learns has made computer systems more effective as tutors. Intelligent tutor systems can provide a student with individual instruction similar to how human tutors do. In addition, intelligent tutor systems can be fairly inexpensive to build and can be personalized to a student's individual needs.

Two methods that are used to make intelligent tutoring systems effective, personalized tutors are model tracing and knowledge tracing. Model tracing is used to monitor and guide a student through a problem [KAHM97]. Model tracing works by comparing every action the student makes with the actions the model generates. Model tracing systems make use of production rules which are a set of rules that mimic how a human solves a particular problem. Production rules thus give computer systems the ability to model how humans solve problems. With this knowledge, intelligent tutoring systems can provide appropriate hint messages when a student gets stuck along with better feedback when they make an error.

The second method, knowledge tracing, is used to monitor a student's progressive learning [KAHM97]. A problem usually involves one or more skills in order to be solved. Knowledge tracing keeps track of which skills a student has mastered and which skills they need to improve. By knowing which skills a student needs improving in, an intelligent tutoring system can customize which problems it presents to the student. For example if a student has become proficient in writing algebraic expressions with positive slopes but is deficient at writing expressions with negative slopes, an intelligent tutoring system could give that student more problems involving negative slopes.

Creating an accurate model of a student's current knowledge can be quite difficult [KLEG92]. Problems usually have more than one skill associated with them, so when a student gets a problem wrong it is difficult to determine why they answered it incorrectly. They could have answered the problem wrong because they are deficient in just one of the skills needed to solve the problem or they could be lacking in a couple skills or all of them. Another possibility for a student getting a problem wrong is because they made a simple mistake. Perhaps they entered 6 when they meant to enter 7 for example. In addition, a student could be a good guesser, so just because they answered the problem correctly does not always mean they truly have the skills necessary to solve the problem.

Creating a transfer model can make it easier to assess what skills are required for a given problem. Transfer models are also useful in identifying hidden skills. A transfer model is a mapping between different types of problems and the skills needed to solve them. If two different problem types have one or more skills in common then this suggests that transfer between these two different problems exists. Thus, students practicing with one type of problem might cause them to perform better on the other type of problem.

Figuring out which skills problems have in common and which skills are unique to a particular problem is not always easy. For example, is a problem that asks a student to find the area of a square different from one that asks for the area of a rectangle or one that asks for the area of a parallelogram? Since there is more work involved in finding the area of a parallelogram as opposed to a rectangle or square, a skill could be added to a transfer model that just applies to parallelograms. The only way to determine whether adding a skill for parallelograms is worthwhile is to evaluate a transfer model without the skill and then evaluate the transfer model with the parallelogram skill added. If the transfer model with the added parallelogram skill leads to better prediction of whether a student will get a given problem correct or not than the transfer model without the skill, then the additional parallelogram skill is worth having. The process of determining which skills are worth adding can become very tedious as the number of potential skills increases.

An automated search process for transfer models makes it easier to find out which skills are important and which ones are not. The search space for transfer models can become quite huge as the number of factors under consideration grows. For example, if there are three factors under consideration for use in a transfer model where each factor has two different quantitative values then nine initial transfer models can be created from these factors. From each of the nine initial transfer models ten new transfer models can be created (why ten will be shown in section 2.3). Thus, if one is just considering transfer models with at most two skills there are about a hundred different models to evaluate. The number of transfer models that can be created with at most three skills is over a thousand. It is not unreasonable to have a transfer model with more than six skills.

Some previous work on creating an automatic search process for generating transfer models has already been done [OH03]. The previous work, however, used a brute force approach in finding the best fitting model for a given dataset. Our work has developed a guided search process that builds on the previous blind search process that was developed. A data mining technique, association rules, is used to guide our search. It has been found that by mining for association rules in the instances a model predicts incorrectly, information can be obtained on how to guide the search process more efficiently toward finding a better model.

A guided search process allows for quicker discovery of good fitting transfer models for a dataset. In addition, the transfer models generated from the search process have the potential of discovering new information as to what exactly makes a particular problem difficult and others easy.

1.1 Related Work

A considerable amount of work has been done on knowledge tracing. In order to address the difficulty in creating accurate student knowledge models, work has been done with Bayesian networks [GCV98] and fuzzy models [KLEG92]. A common way to model student knowledge is to make use of four parameters: the probability of initial learning, the probability of acquisition, the probability of guess and probability of slip [CA95]. The probability of initial learning is the probability that the student has already learned a rule or skill. The probability of acquisition is the probability that a student will learn a rule or skill on the current problem. The probability of guess, is the likelihood the student will guess the correct answer, and the probability of slip is the likelihood of them making an unintended mistake. As of yet, no work has been found that involves association rules to improve student knowledge and transfer models. However, there exists prior work on using association rules to select relevant features for logistic regression [L04].

1.2 Contributions and Applications

This work has developed a fast method of finding good transfer models that describe student learning for a dataset. This allows a researcher to determine more quickly which skills and factors are worth studying with regards to student learning. This work can also be applied to intelligent tutoring systems. With an accurate model of student learning an intelligent tutoring system can determine how much help it needs to give a student for a particular problem. In addition, with an accurate model of student learning an intelligent tutor system can better choose which problems it gives to a student. If an intelligent tutor system predicts that a student has a very low chance of getting a particular question correct, the system can make sure to give the student easier questions that share skills in common with the difficult question first before presenting the student with the difficult question. This work increases the potential of an intelligent tutoring system being able to revaluate its student model in real time i.e. if a intelligent tutor system finds that it is not predicting a student's performance well it can search for a better model that will improve its interaction with the student.

2 Background

This thesis work draws its resources from four main areas. The areas are association rules, the data gathered from the online algebra tutor Ms. Lindquist, transfer models, and logistic regression.

2.1 Association rules, Apriori and ASAS

Given a collection of data instances, each one described in terms of a set I of attributes or features (sometimes called "items"), association rules are rules of the form X => Y, where X and Y are disjoint subsets of I [AIS93]. Apriori is an algorithm that generates association rules [AS94]. It is probable to find many associations in a small dataset so interest is usually focused on rules that have high accuracy and/or large support. Support for an association rule is defined as the number of dataset instances it predicts correctly divided by the total number of instances. Confidence for an association rule is defined as the number of instances it predicts correctly divided by the number of all the instances it applies to. Another parameter that can be used in addition to support and confidence is lift. Lift for an association rule is calculated by dividing the confidence of the rule by the expected confidence. The expected confidence is the number of instances in the dataset that include the consequent divided by the total number of instances. Yet another measure for evaluating an association rules is Chi square. Chi square is a means of determining how well an association rule would apply to a larger dataset. AprioriSetsAndSequences (ASAS) is an extension to the Apriori algorithm [P04]. ASAS allows one to mine temporal data. In addition, ASAS provides enhanced pruning methods for association rule mining.

2.2 Ms. Lindquist

Ms. Lindquist is an algebra tutor designed to help students learn how to write algebraic expressions [HK03a], [HK03b]. It can be found at http://www.algebratutor.org. The data for this project comes from the log files Ms. Lindquist generates when a student uses the tutor. Ms. Lindquist is targeted towards 7th and 8th grade students and takes a dialog based approach. Every problem Ms. Lindquist gives is a word problem. An example problem is "Mary is 800 ft from the dock. If she rows 40 feet per minute how many feet is she from the dock after x minutes." For an answer, the student is expected to type 800-40*x.

The problems Ms. Lindquist gives are divided into sections. The first section contains only problems that involve one mathematical operation. The next section deals with problems that involve two mathematical operations. There are several other sections in Ms. Lindquist, unfortunately, not enough students make it beyond the second section to provide enough data for analysis.

Every section begins with a pretest and ends with a post test. For the pretest, students are given two to three problems with no help from the tutor. If a student gets every question right on the pretest they have the option of moving to the next section. On the other hand, if a student fails to answer all of the pretest questions correctly then they must complete the current section. The problems Ms. Lindquist gives inside a section are randomized so two students using Ms. Lindquist will most likely get a different set of problems. To graduate to the next section a student must get three problems correct in a row. The post test is administered to the student as part of the current section and the student is able to get help from Ms. Lindquist while taking it.

Ms. Lindquist provides help when a student gets a problem wrong. The type of help Ms. Lindquist provides is based on which strategy is being used for the current section. There are five different strategies available to Ms. Lindquist for dealing with student errors. Ms. Lindquist uses the same strategy throughout each section but can switch strategies from one section to another.

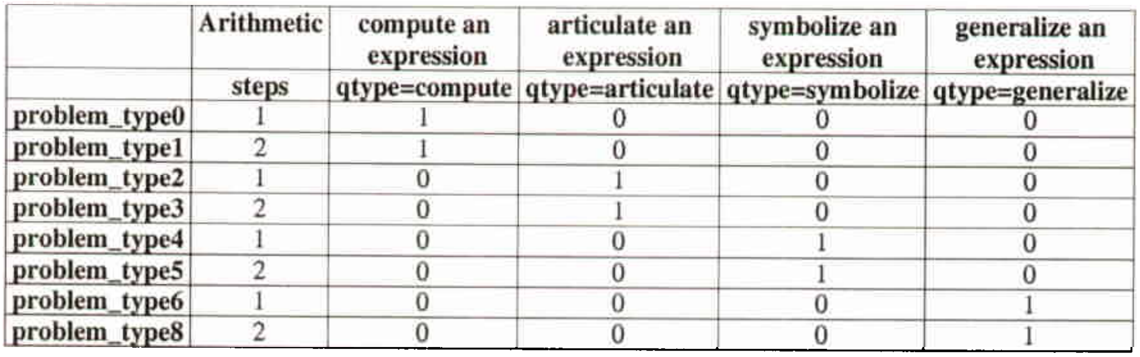
Of the five strategies that Ms. Lindquist uses, we are only looking at two of them. The two strategies are cut to the chase and inductive support. For the cut to the chase strategy, if a student gets a problem wrong Ms. Lindquist just gives them correct answer. With the inductive support strategy, Ms. Lindquist enters into a dialog with the student. The student is asked sub-problems that relate to the original question they answered incorrectly. The sub-problems Ms. Lindquist asks in its dialog with the student attempt to break down the original problem into smaller steps that lead the student to the correct and final answer for the original word problem.

The dialog Ms. Lindquist can have with a student is limited. Therefore if a student is completely lost, Ms. Lindquist will eventually bottom out. Instead of just giving the student the answer it was looking for though, when it bottoms out Ms. Lindquist will ask the student a multiple choice question.

The questions that Ms. Lindquist asks are of three types, compute, symbolize and articulate [HC03]. When Ms. Lindquist asks a student a compute question, the tutor is expecting the student to give a numerical answer. For symbolize questions the student's answer is expected to contain a variable, and for articulate questions the student is expected to write a complete expression that shows all the steps. As mentioned above, the first section consists of one step problems and the second section deals with, two step problems so there are actually six different question types that Ms. Lindquist has: one step compute, one step symbolize, one step articulate, two step compute, two step symbolize and two step articulate.

2.3 Transfer Models

A transfer model is represented by a table where the rows represent the different problem types and the columns indicate the knowledge components or skills needed for each problem type. The number of factors times the number of values each factor can have determines the number of rows a transfer model will have. Factors are information Ms. Lindquist logs about each question a student answers such as the number of steps the question involves or what type of question it is. Thus, if there are two factors, number of steps and question type, where the number of steps for a question can be either one or two and the type of the question can be of one of four types (compute, articulate, symbolize and generalize), a transfer model for these two factors would have eight rows. Table 1 depicts an example of how a transfer model incorporating these two factors might look. The first row in Table 1 shows potential names for the skills in the transfer model. Creating names for skills in a transfer model is a highly creative process and therefore transfer models presented in later sections will not contain names for the skills. Transfer models presented in later sections will only include the second row in Table 1 which shows the associated factor value pair for each skill in the model.

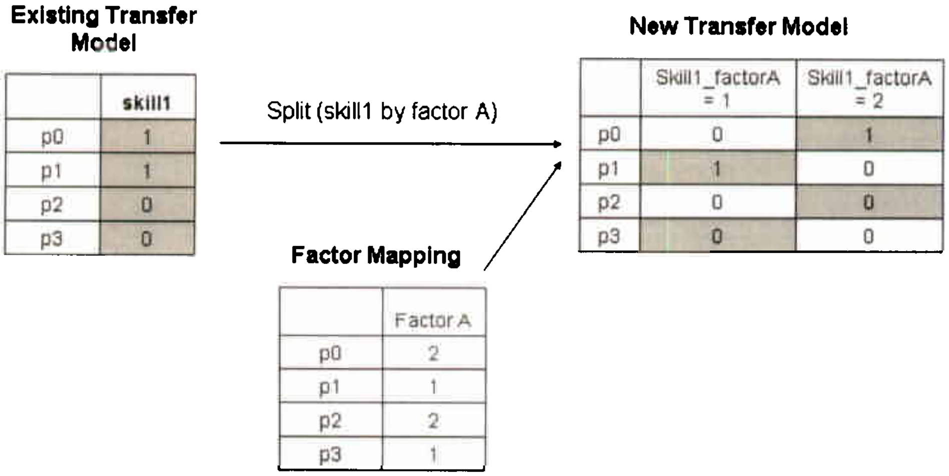


	Arithmetic	compute an expression	articulate an expression	symbolize an expression	generalize an expression
	steps	qtype=compute	qtype=articulate	qtype=symbolize	qtype=generalize
problem_type0	1	1	0	0	0
problem_type1	2	1	0	0	0
problem_type2	1	0	1	0	0
problem_type3	2	0	1	0	0
problem_type4	1	0	0	1	0
problem_type5	2	0	0	1	0
problem_type6	1	0	0	0	1
problem_type8	2	0	0	0	1

Table 1: Example transfer model, the first row shows potential names for the skills, the second row shows which factor value pair the skill is associated with, qtype= question type

When a factor is included in to a transfer model it is referred to as a knowledge component or skill. Thus, the columns in the above transfer model represent skills. Factors are included into a transfer model in one of two ways, map and add. Mapping a factor into a transfer model creates one skill for that factor. In the transfer model above the factor number of steps was mapped into the transfer model. Adding a factor creates a new skill for every value the factor can have. In the example, above, the factor question type had four possible values and thus four skills were created. A refined add can also be performed where only one skill is created for only one value of a factor.

New skills can also be created by another operation called split. Splitting makes use of an existing skill *s* and a factor *f*. The existing skill is replaced with *n* new skills where *n* is the number of values the factor *f* can have. Figure 1 shows how a split operation is performed. A refined split can also be performed where a skill is only split into two new skills based on one value for a factor *f*.



Existing Transfer Model

	skill1
p0	1
p1	1
p2	0
p3	0

Factor Mapping

	Factor A
p0	2
p1	1
p2	2
p3	1

New Transfer Model

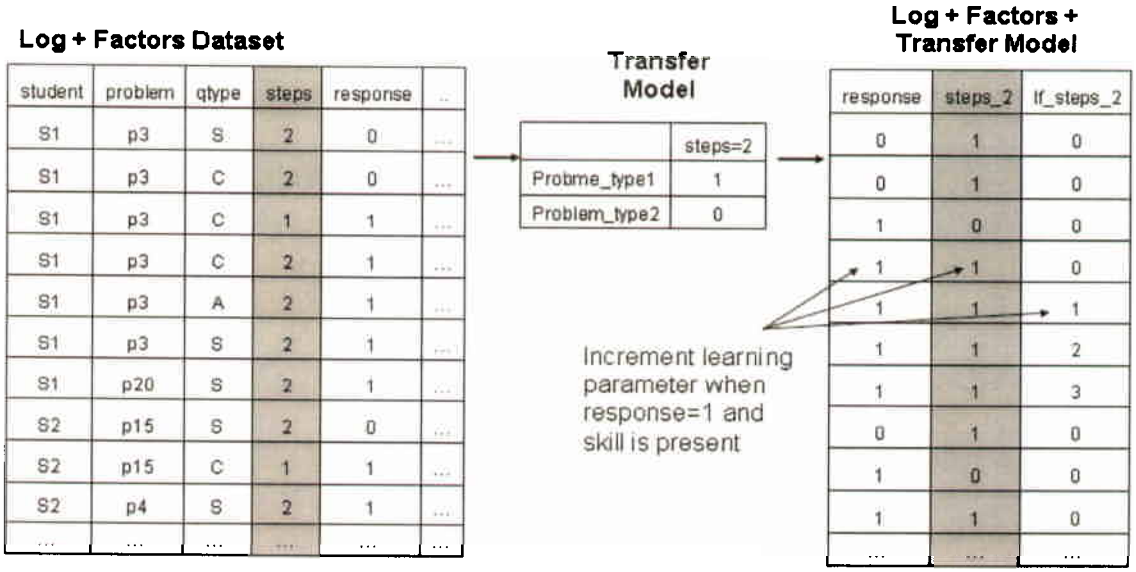
	Skill1_factorA = 1	Skill1_factorA = 2
p0	0	1
p1	1	0
p2	0	0
p3	0	0

Figure 1: Example of a split operation

2.4 Logistic Regression

Logistic regression, a statistical method, is used to evaluate a transfer model. Why a logistic regression best suits a transfer model is best explained in [JKT00]. A logistic regression is used when the dependent variable is binary. In the case of transfer models, we want to use a transfer model with a logistic regression to predict whether a student will get the answer to a given problem correct or not. Thus, whether the student gets the problem correct or not is the dependent variable and is binary. A value of one represents the student answered the problem correctly and a value of zero represents the student answered the problem incorrectly.

When a transfer model is used with a logistic regression, the transfer model is mapped to a dataset. The dataset in this case is called a Log+Factors dataset. The Log+Factors dataset is a set of instances representing information about problems answered by students. Skills in the transfer model, when mapped to a Log+Factors dataset, are referred to as difficulty parameters. In addition, the dataset is augmented with another set of values called learning factors. Leaning factors keep track of the number of times a student has previously encountered a skill in a problem and answered the problem correctly. Every skill has an associated learning parameter. Figure 2 shows an example of a transfer model being mapped to a dataset and the resulting input file for a logistic regression. The input file for a logistic regression is also referred to as a Log+Factors+TransferModel dataset. How a transfer model is mapped to Log+Factors dataset is explained in section 5.



Log + Factors Dataset

student	problem	qtype	steps	response	..
S1	p3	S	2	0	...
S1	p3	C	2	0	...
S1	p3	C	1	1	...
S1	p3	C	2	1	...
S1	p3	A	2	1	...
S1	p3	S	2	1	...
S1	p20	S	2	1	...
S2	p15	S	2	0	...
S2	p15	C	1	1	...
S2	p4	S	2	1	...
...	...	...	...	...	...

Transfer Model

	steps=2
Probme_type1	1
Problem_type2	0

Log + Factors + Transfer Model

response	steps_2	lf_steps_2
0	1	0
0	1	0
1	0	0
1	1	0
1	1	1
1	1	2
1	1	3
0	1	0
1	0	0
1	1	0
...	...	...

Increment learning parameter when response=1 and skill is present

Figure 2: Example of a transfer model being mapped to a Log+Factors dataset and the result Log+Factors+TransferModel dataset. Response = student's response, steps_2 is the skill/difficulty parameter for steps is 2 and lf_steps_2 is the learning factor for steps_2

Once a transfer model has been mapped to a Log+Factors dataset and the resulting Log+Factors+TransferModel dataset is created, a logistic regression is run on the Log+Factors+TransferModel dataset to determine how fit the transfer model is. There are many different ways to determine how fit a model is and to compare how one transfer model measures up to another. For this project, the Bayesian Information Criterion (BIC) will be used as the main method in comparing one model to another. BIC is a statistical number used to compare one model to another. For one model *m* to be more statistically significant than another model *n*, the BIC value of *m* must be at least 10 points lower than *n*'s BIC. Overall accuracy will be used to determine the strength of a given transfer model on its own.

One thing to note when using a logistic regression on a Log+Factors+TransferModel dataset is that the coefficients for the learning parameters in the regression equation created by the logistic regression should be positive (the coefficients are the output of the logistic regression). If the coefficients are negative that means students are unlearning things. Since it is not expected that students unlearn skills as they solve problems, negative learning factors are considered to be invalid.

Transfer models that have learning parameters with negative coefficients can be fixed by removing the learning parameters with negative coefficients and rerunning the logistic regression. If all the remaining learning parameters have positive coefficients after the second run then the transfer model is plausible. If yet more learning factors with negative coefficients are found, the process can be repeated until there are no learning parameters with negative coefficients or no learning parameters at all.

3 Factor Collection

Before any searching for transfer models was done, the problems given by Ms. Lindquist were reviewed to determine if any additional factors could be included in the search process. There are 100 problems that Ms. Lindquist has to give to students. Ms. Lindquist associates some factors with the problems it asks, such as the number of steps the question involves and what type of question it is. The hundred problems were reviewed to see what other potential factors could be used to build transfer models.

Figure 3 shows two examples of the type of algebra problems Ms. Lindquist gives to students.

Problem: h_snowcones_1s-50
"A friend offers you a job selling snow cones. Since he owns the snow cone maker, he will let you use it for $50. You will make $1 on each snow cone you sell. The amount you get from selling the snow cones, minus $50, is your profit. Write an expression that will show your total profit if you sell 's' snow cones."
Expected Answer: 1*s – 50

Problem: h_bamboo_3x_15
"Bamboo plants can grow at the rate of 3 feet per day. One bamboo plant is currently 15 feet high. If it grows for 'x' more days, write an expression for the total height of the tree."
Expected Answer: 15+3*x

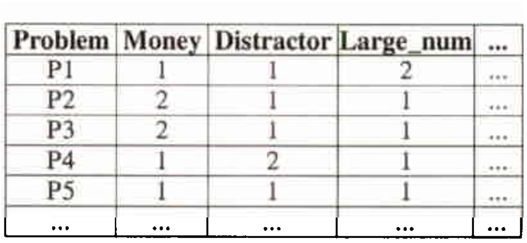
Figure 3: Examples of Ms. Lindquist problems

After reviewing the 100 problems, several new factors were proposed to be used in the search process. These factors were *form*, *large numbers*, *two variables*, *money*, *temperature*, *uncommon units*, *implied operation*, and *distractor*. The factor *form* represents the different type of algebraic expressions the student is expected to input as an answer. All student answers can be classified as being one of the following forms, *variable+number*, *variable-number*, *number-variable*, *variable*number*, *variable/number*, *number/variable*, *number*variable+number*, *number*variable-number*, *number-variable*number*, *-number-variable*number*, *number+number+variable*, or *number+(number-variable)*. The first question in Figure 3 would be classified as being of the form *number*variable-number* and the second question in Figure 3 would be classified as being of the form *number*variable+number*.

As for the other factors, *large numbers* represents whether a problem uses numbers greater than 100. The factor *two variables* determines whether a problem uses one or two variables. The factor *money* represents whether a problem involves money or not. The factor *temperature* determines whether a problem relates to temperature or not. The factor *uncommon units* represents whether a problem uses units that are not necessarily familiar to a student like PSI (Pounds per Square Inch) or doubloons. The factor *implied operation* means the student has to do an extra step that the question does not specifically state in order to get the correct answer. The following problem is an example of one that has an implied operation, "Andrea's mother makes brownies for her and her 4 friends. Each person will receive an equal amount of brownies. If the brownie pan is cut into 'x' pieces, write an expression that shows how many pieces each person receives." The student needs to take into account that it is Andrea and her 4 friends which means there are five people total so the answer is x/5 and not x/4.

The factor *distractor* represents whether the problem contains more information than is needed to solve the problem. A problem that does contain more information than is needed to solve a problem is considered to have a distractor. For all the factors except *form*, a value of 2 means that factor is present in a given problem and a value of 1 means the factor is not present.

A factor table was created for these additional factors. The first column in the table contains the name of each of the 100 problems. The remaining columns in the table are for each factor. Thus, each row in the table shows which factors each problem contains and which factors the problem does not contain. Table 2 shows an example of how a factor table might look.



Problem	Money	Distractor	Large_num	...
P1	1	1	2	...
P2	2	1	1	...
P3	2	1	1	...
P4	1	2	1	...
P5	1	1	1	...
...	...	...	...	...

Table 2: Example of a factor table, P# represents the name of the problem, a value of 2 for a factor means it is present and a value of 1 means it is not present

Once the factor table was created it could be easily seen which factors where shared by many problems and which factors were specific to just a few of the problems. The following is how many problems each of the factors can be found in (from a total of 100 problems):

large numbers = 25 problems
two variables = 10 problems
money = 40 problems
temperature = 2 problems
uncommon units = 7 problems
implied operation = 11 problems
distractor = 3 problems

As can be seen, not many problems contain a *distractor*, *uncommon units* or relate to *temperature*. The number of problems containing an *implied operation* or *large numbers* is not too high either. Thus, although it would be interesting to try to determine if such things as a *distractor* effect students' performance, Ms. Lindquist does not give enough problems containing distractors. As mentioned in the Background Section, Ms. Lindquist randomly chooses which problems it assigns to students. Thus, a student is not even guaranteed to get a problem involving a distractor or temperature. *Money* seems to be the only factor where it can be assumed that students are getting a good mix of problems relating to money and problems not relating to money.

4 Log+Factors Dataset Creation

All data used by our search process comes from the log files of Ms. Lindquist. Ms. Lindquist logs every question given to a student and every answer the student gives. Ms Lindquist also logs whether or not the answer was correct, along with several other things. A typical student session in Ms. Lindquist is shown in Figure 4.

Problem ann_rows

1. **Tutor:** Anne is rowing a boat in a lake and is 800 yards from the dock from which she started. She rows back towards the dock at 40 yards per minute for "m" minutes and stops to rest. How far is she from the dock now? Please write an expression for the distance Anne has left to row.
   Student: 8-40/m

2. **Tutor:** Hmm. No. We will come back to the distance Anne has left to row. Let me try to break this down for you. Please compute the distance Anne has left to row if the number of minutes rowing was 3 minutes?
3. **Student:** 580
   Tutor: Hmm. No. Let me ask you a different question. Please compute the distance rowed so far if the number of minutes rowing was 3 minutes.
   Student: 120
4. **Tutor:** Correct. Now, lets try this question again: Please compute the distance Anne has left to row if the number of minutes rowing was 3 minutes?
   Student: 680
5. **Tutor:** Correct. Using only 800, 40 and 3 write an expression for the distance Anne has left to row.
   Student: 800-40*3
6. **Tutor:** Correct. Now use "m" instead of 3 to again express the distance Anne has left to row.
   Student: 800-40*m
   Tutor: Correct. Lets do a new problem.

Problem mikes_earnings

7. **Tutor:** Michael works as a waiter. Saturday he worked "h" hours. The restaurant pays him an hourly wage of $3 an hour. He also made $40 dollars in tips. Write an expression for the total amount he earned Saturday Night.
   ...

Figure 4: Example of Ms. Lindquist student session

The first question Ms. Lindquist asks for a problem is called a top level question. The top level question is always a *symbolize* question. The student is expected to enter in an algebraic expression that contains one or more variables as their answer. In Figure 4 the first question (denoted by the number one) is an example of a top level question. If the student had answered the first question correctly they would have jumped straight to the next problem (which is denoted by the number seven in Figure 4). However, in the example shown, the student answered the first question incorrectly, therefore Ms. Lindquist provides the student with some tutoring that attempts to lead the student to the correct answer. Questions 2 through 6 are examples of the types of tutoring question Ms. Lindquist gives to the student. These questions can be considered sub questions of question 1 since they are related to that question.

Ms. Lindquist asks two different types of questions when providing tutoring. Question 2 is an example of one type which is called *compute.* For compute type questions, the student is expected to give a numerical value for an answer. The other type of tutoring question Ms. Lindquist uses is called *articulate.* Question 5 is an example of an articulate question. For articulate questions students are expected to write an algebraic expression without using variables.

All this information and more about a student's session is written to a separate log file for each student. To create our dataset for the search process the student log files are parsed and all the data for all students is written to one master file which will be referred to as the Log dataset. Table 3 is an example of how the file looks.

student	*problem*	*qtype*	*steps*	*Response*
S1	ann_rows	S	2	0
S1	ann_rows	C	2	0
S1	ann_rows	C	1	1
S1	ann_rows	C	2	1
S1	ann_rows	A	2	1
S1	ann_rows	S	2	1
S1	mikes_earnings	S	2	1
S2	sara_swims	S	2	0
S2	sara_swims	C	1	1
S2	h_ticket	S	2	1
...	...	...	...	...

Table 3: Example Log dataset created from Ms. Lindquist log files

The first seven rows in Table 3 correspond with the sample Ms. Lindquist session in Figure 4. The first column represents the student's login name. The second column is the name of the problem. The third column is the type of each question given to the student for the problem. S stands for *symbolize.* C stands for *compute* and A stands for *articulate.* The last two columns represent the number of steps the question requires and the students' response. For response, 1 means the student answered the question correctly and 0 means that the student gave an incorrect answer. All student sessions are appended together in the Log dataset, so the last three rows in Table 3 represent the beginning of another students' session.

Table 3 shows how the data obtained from Ms. Lindquist looks. This is not the final dataset that is used in the search process though. The final dataset for the search process is created by mapping the factor table which contains additional factors to be considered. This final dataset is referred to as the Log+Factors dataset. The factor table is mapped to the Ms. Lindquist by problem name. Table 4 represents a factor table with just one factor, money. Table 5 represents what the Log+Factors dataset would look like if the factor table in Table 4 were mapped to the Log dataset shown in Table 3.

Problem	money
ann_rows	1
mikes_earnings	2
sara_swims	1
h_ticket	2

Table 4: Example factor table with just one factor, Money=1 means money is not present. Money=2 means money is present

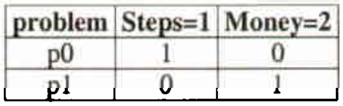
Student	*response*	*Problem*	*steps*	*Qtype*	*money*
S1	0	ann_rows	2	S	1
S1	0	ann_rows	2	C	1
S1	1	ann_rows	1	C	1
S1	1	ann_rows	2	C	1
S1	1	ann_rows	2	A	1
S1	1	ann_rows	2	S	1
S1	1	mikes_earnings	2	S	2
S2	0	sara_swims	2	S	1
S2	1	sara_swims	1	C	1
S2	1	h_ticket	2	S	2
...	...	...	...	...	...

Table 5: Example of a Log+Factors dataset that results from mapping the factor table in Table 4 to the Log dataset in Table 3

The actual Log+Factors dataset used in the search process does not contain a column for problem names. The search process itself has no need for it. The column containing problem names is only needed for mapping a factor table to a Log dataset. Once the mapping is done, the problem column can be removed. The two required columns the search process needs is the student column and response column. The student column allows the process to determine when one session ends and another begins. The response column tells whether the student answered a given question correctly or not. The student column and response column are expected to be the first two columns in the Log+Factors dataset. The remaining columns in the Log+Factors dataset are factors to be used to create transfer models.

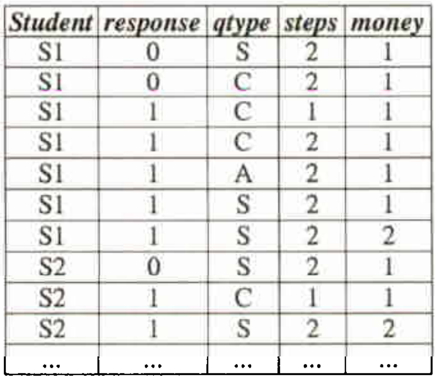
5 Evaluation of a Transfer Model using Logistic Regression

We can better explain in this section how a transfer model is evaluated using logistic regression. More specifically we will show what the input and outputs to a logistic regression are. Table 6 shows a sample transfer model with two skills. Table 7 shows a sample Log+Factors dataset with three factors.



problem	Steps=1	Money=2
p0	1	0
p1	0	1

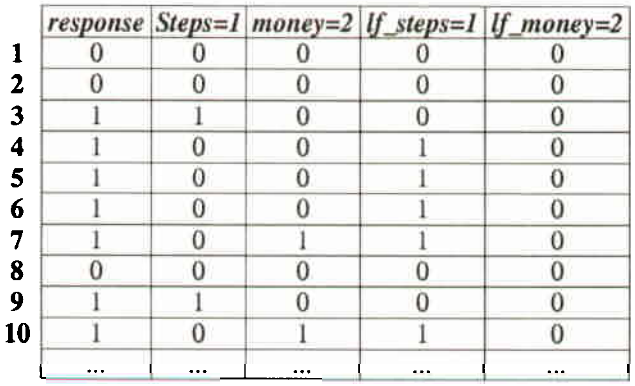
Table 6: Example transfer model with two skills



Student	*response*	*qtype*	*steps*	*money*
S1	0	S	2	1
S1	0	C	2	1
S1	1	C	1	1
S1	1	C	2	1
S1	1	A	2	1
S1	1	S	2	1
S1	1	S	2	2
S2	0	S	2	1
S2	1	C	1	1
S2	1	S	2	2
...	...	...	...	...

Table 7: Sample Log+Factors dataset from Table 5

To create the input for a logistic regression a transfer model is mapped to the Log+Factors dataset creating yet another data file that we call the Log+Factors+TransferModel dataset. The Log+Factors+TransferModel dataset is the input to the logistic regression. Mapping a transfer model to a Log+Factors dataset is similar to how a factor table is mapped to a Log dataset. Instead of mapping based on problem name as with factor tables, transfer models are mapped based on factor values. The number of skills in a transfer model determines how many columns the Log+Factors+TransferModel dataset for the logistic regression has. When a transfer model is mapped to a Log+Factors dataset, an additional column for each skill is created. These additional columns are called the learning factors. Each learning factor for a student is incremented every time the student gets a problem correct that contains the skill associated with the learning factor. Table 8 shows the resulting input file (i.e. the Log+Factors+TransferModel dataset) for logistic regression when the transfer model in Table 6 is mapped to the sample Log+Factors dataset in Table 7.



	response	*Steps=1*	*money=2*	*lf_steps=1*	*lf_money=2*
1	0	0	0	0	0
2	0	0	0	0	0
3	1	1	0	0	0
4	1	0	0	1	0
5	1	0	0	1	0
6	1	0	0	1	0
7	1	0	1	1	0
8	0	0	0	0	0
9	1	1	0	0	0
10	1	0	1	1	0
	...	...	...	...	...

Table 8: Example input file for logistic regression obtained by mapping the transfer model in Table 6 to the Log+Factors dataset in Table 7

Rows one and two in Table 8 represent the first two questions the first student was asked. Since neither of those questions were one step questions or involved money, there are zeros in the steps=1 and money=2 columns for these questions. At row three in Table 8 the student gets a question that is a one stepper and the student answered it right, so a one is placed in the lf_steps=1 column in row four. Lf_steps again stands for the learning factor for the steps=1 skill. Learning factors keep track of how many times a student has successfully demonstrated a particular skill. A learning factor can also be interpreted as the amount of experience a student has had with a certain skill.

The student does not get any more one step problems after the third question and therefore the value in the lf_steps=1 column is not incremented. Row eight in Table 8 represents the beginning of a new student session. The learning factors are reset when a new student session is started. Thus, although the student answered the question in row 7 of Table 8 correctly, a one is not placed in the lf_money=2 column of row eight because row eight is the beginning of data for a different student.

For the logistic regression, *response* is the dependent variable. The other columns in the Log+Factors+TransferModel dataset are used to create a regression equation that predicts response. The output from a logistic regression looks something like the following.

```
                 Coef      tvalue
Intercept        -0.41     -3.66
steps=1          0.70      3.47
money=2          0.42      1.06
lf_steps=1       0.23      2.79
lf_money=2       0.77      0.58

BIC = 2402.172
```

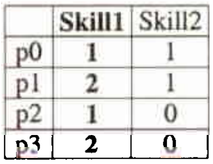
The tvalues for the skills and learning factors represent how significant those parameters are for the logistic regression. The BIC (Bayesian Information Criterion) number is used to compare how good one model is to another. Since it is a logistic regression, the coefficients for the regression are logged odds. Thus, to find the logged odds for response, the equation is (where RLODDS stands for response logged odds):

```
RLODDS = -0.41
         + 0.70*steps=1
         + 0.42*money=2
         + 0.23*lf_steps=1
         + 0.77*lf_money=2
```

For row seven in Table 8, RLODDS is equal to -0.41 + 0.70*0 + 0.42*1 + 0.23*1 + 0.77*0 = 0.23. The equation to convert a logged odd to a probability is (e^logged_odd)/(1+e^logged_odd)). Thus, the predicted probability that the student would get the question in row seven of Table 8 correct is (e^.23)/(1+e^.23) = 0.56. Thus the student had a 56 percent probability of answering the question in row seven of Table 8 correctly. If the probability is greater than 50 percent we predict that the student will answer the question correctly. If the predicted probability for a correct answer is less than 50 percent then we predict the student will not answer the question correctly.

6 Search for a Fit Transfer Model: Blind Search

Finding a good transfer model that accurately predicts whether or not a student will answer a given question correctly is not easy. As a result, some work has already been done to create a search process to find suitable transfer models for a dataset [OH03]. The search process uses a brute force method to find transfer models. The search process implemented a depth first search algorithm and used four operations to create transfer models. Three of the operations- *map*, *add* and *split*- were explained in the Background section on transfer models. The fourth operation that the blind search method used was called *filter*. A filter operation removes a specific value for a skill in a transfer model. Table 9 shows a transfer model before a filter operation and Table 10 shows the same transfer model where the twos for skill1 have been filtered out.



	Skill1	Skill2
p0	1	1
pl	2	1
p2	1	0
p3	2	0

Table 9: Table before filter

	Skill1	Skill2
p0	1	1
p1	0	1
p2	1	0
p3	0	0

Table 10: Table after filter of 2 from skill1

The blind search method is good because it considers all the possible transfer models that can be constructed for a dataset. However, the number of possible transfer models for a dataset can be quite large. The main bottleneck of the search process is evaluating each model with logistic regression. The other problem with the blind search method is the large amounts of memory that it consumes.

The number of transfer models that can be generated for a dataset with just a few factors can be quite large. For example a dataset with three factors, *steps*, *question type* and *money* can have thousands of transfer models with only three skills. The number of transfer models with four or five skills for such a dataset would be in the tens of thousands. Yet another disadvantage of the blind search method is that it does not use any heuristic about which transfer model in the search space to expand next. Thus, the blind search method can spend a lot of timing going down paths that do not yield good transfer models before stumbling onto a path that results in a good transfer model.

The blind search method starts with an initial transfer model that is empty (contains no skills). Only adds and maps can be performed on an initial model. The blind search method will create a transfer model for every factor value pair that exists in the dataset. Thus, if a dataset has three factors where each factor has two values then the blind search method creates six transfer models for the first level, provided all the factors are categorical. Quantitative factors can be mapped as well as added, so one extra transfer model is created for each quantitative factor.

Each transfer model in the first level of the search tree contains only one skill. These transfer models are used to create the next level of transfer models of size two. Transfer models in the second level of the search tree can be created by adding another skill to one of the first level transfer models, splitting the skill in one of the first level transfer models or by filtering out one of the values for the skill in one of the first level transfer models. This process repeats for every level after the first. Figure 5 shows a partial example of what the search tree for the blind search method might look with a maximum depth limit of three. The boxes Figure 5 represent transfer models and the numbers inside the boxes represent the order in which the blind search method evaluates the transfer models.

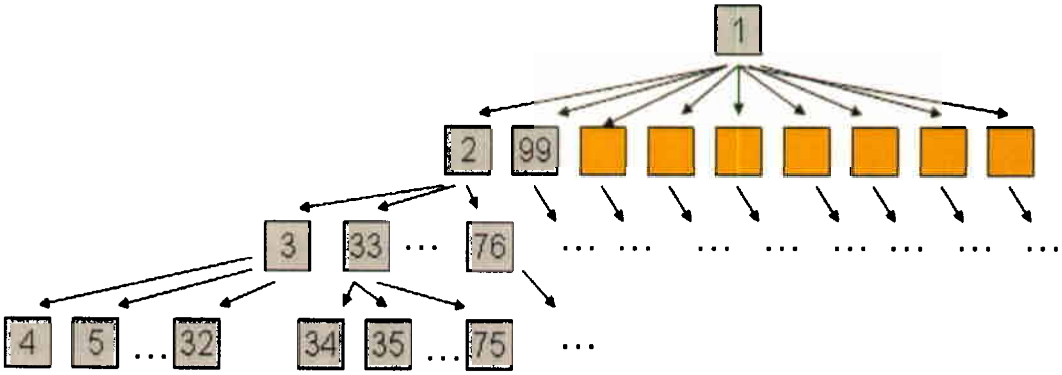


Figure 5: Example search tree for the blind search method

Thus, if one is looking for a transfer model with at most six or seven skills, the search process has a lot of possible transfer models to consider. The search method introduced in this thesis work greatly reduces the search tree for transfer models and allows good transfer models to be found in considerably less time.

7 Our Search Algorithm Using Association Rules

For this project, we used association rules to guide a search process in finding transfer models that predict student's success. The transfer models themselves provide information as to what skills are required to solve a particular type of problem. The search process starts with a list of factors, a Log+Factors dataset and an empty transfer model as the root node. The Log+Factors dataset is a table where the rows represent a question Ms. Lindquist gave to a student and the columns are the factors for each question. There is one additional column in the Log+Factors dataset that indicates whether the student answered the question correctly or not.

Each node in the search tree represents a new transfer model. To generate new transfer model nodes from a current transfer model node, the search algorithm performs one of three operations on the current transfer model. The three operations are map, add and split (the filter operation was not used in this search process because a suitable association rule for the operation could not be determined). Only one operation is performed on a transfer model at a time and each operation includes only one new skill in the current transfer model, creating a new transfer model node. The new transfer model node becomes a child of the transfer model node it was created from.

Since the root node contains an empty transfer model, only add or map operations can be performed on the root node to create the first level of transfer models. A list of possible skills to include in the transfer model is generated by mining the Log+Factors dataset for factors associated with a student's response being correct or incorrect. The initial set of mined rules have the form 'factor=value ==> response=(0|1) [Conf: #, Sup: #, Lift: #, Chi-square: #]' where *factor* is the name of a factor in the Log+Factors dataset (e.g. steps, question type), *value* is one of the possible values the factor can have (e.g. steps can have a value of either 1 or 2), *response=0* means students' response is incorrect and *response=1* means student's response is correct. A hypothetical set of association rules obtained from mining the Log+Factors dataset is shown in Figure 6.

```
steps=1 ==> response=1
        [conf: 0.9, sup: 0 .6, lift: 1.02, Chi-Square: 30]
qtype=2 ==> response=0
        [conf: 0.9, sup: 0.4, lift: 1.25, Chi-Square: 40]
qtype=3 ==> response=1
        [conf: 0.9, sup: 0.5, lift: 1.04, Chi-Square: 20]
```

Figure 6: Hypothetical set of association rules obtained from mining the Log+Factors dataset

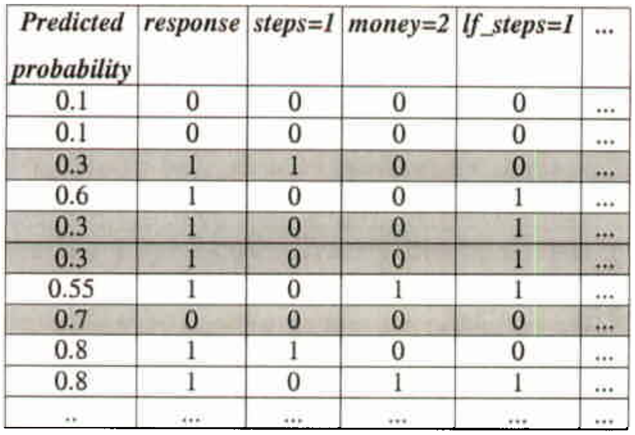
The first rule in Figure 6 is interpreted by the search process as potentially adding a skill for one step problems to the initial model. Since *steps* happens to be a quantitative factor it can be mapped into the initial transfer model as well. The second rule suggests adding a skill for questions of type two, and the third rule suggests adding a skill for questions of type three.

Thus, in this hypothetical situation the search process would create four child transfer model nodes from the root node. The first child node would represent the initial transfer model with a skill for one steps problems added. The second would represent the initial transfer model with the factor steps mapped into the initial transfer model. The third child node would represent the initial transfer model with a skill for questions of type two added. The fourth child node would represent the initial transfer model with a skill for questions of type three added.

Thus, in this hypothetical situation at level one in the search process there are four nodes which have been created and added to the search tree as children of the initial transfer model. The nodes in the first level contain transfer models with only one skill. These first level nodes can be referred to as the base models. It is from these transfer models that all other transfer models are built from.

Transfer models built from the base models are constructed a little differently than how the base models are constructed. First the base models are mapped to the Log+Factors dataset, creating a Log+Factors+TransferModel dataset, and then a logistic regression is run over the Log+Factors+TransferModel dataset (as was described in section 5). The instances that the logistic regression predicted incorrectly are then mined for association rules using the AprioriSetsAndSequences algorithm [P04].

Thus, we are looking for commonalities in the instances in the Log+Factors+TransferModel dataset predicted incorrectly by the logistic regression. An instance is determined to be predicted incorrectly if the predicted probability for the student's response being one is less than 50 percent and the student's actual response is correct, or if the predicted probability is greater than or equal to 50 percent and the student's actual response is incorrect. Put another way, instances for which abs(response – predicted_probability) > 0.5 are considered to be predicted incorrectly. Table 11 shows a sample Log+Factors+TransferModel dataset with an additional column containing the predicted probability for each instance. The highlighted instances in Table 11 are instances that are predicted incorrectly. It is these incorrectly predicted instances that are mined for association rules that will determine the next operation to perform on a transfer model.



Predicted probability	*response*	*steps=1*	*money=2*	*lf_steps=1*	...
0.1	0	0	0	0	...
0.1	0	0	0	0	...
0.3	1	1	0	0	...
0.6	1	0	0	1	...
0.3	1	0	0	1	...
0.3	1	0	0	1	...
0.55	1	0	1	1	...
0.7	0	0	0	0	...
0.8	1	1	0	0	...
0.8	1	0	1	1	...
..	...	...	...	...	...

Table 11: Data file used for logistic regression with incorrectly predicted instances highlighted

The actual data file that is mined for association rules is the merge of the Log+Factors+TransferModel dataset which is the input to the logistic regression and the Log+Factors dataset from which the Log+Factors+TransferModel dataset was created. This allows association rules to be obtained that relate to skills in the transfer model as well as factors that have not been included in the transfer model yet. The association rules found from the instances that the logistic regression predicts incorrectly are used to determine the new transfer models to be created from the base models.

For the initial transfer model only the operations add and map can be used. For all transfer models created from the base models the split operation can be used as well. Association rules of the form 'factor=value ==> response=(1|0)' are interpreted as potential add or map operations to be performed on the base model. Association rules of the form 'skill=value && factor=value ==> response=(0|1)' are interpreted as a potential split operation to be performed on the base model where *skill* in the transfer model is split by *factor* = *value*.

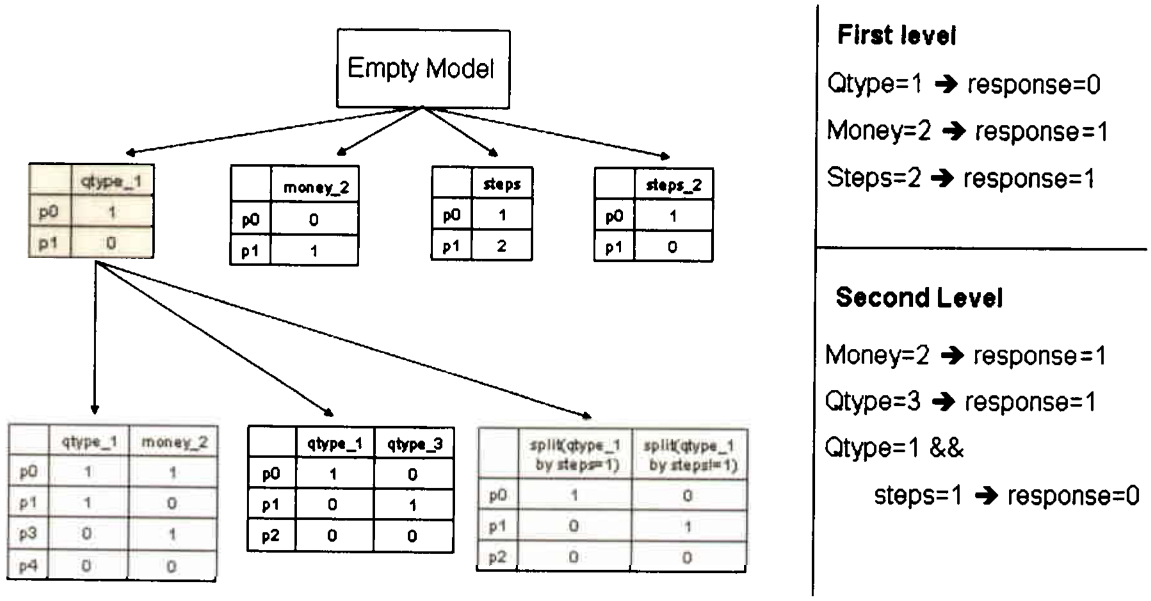
Figure 7 shows a hypothetical set of association rules that could be obtained by evaluating a base transfer model with one skill for one step problems and mining the instances that the model predicts incorrectly.

```
steps=2 ==> response=1
        [Conf: 0.9, Sup: 0.5, Lift: 1.10, Chi: 10]
steps=1 && qtype=2 ==> response=0
        [Conf: 0.8, Sup: 0.7, Lift: 1.20, Chi: 14]
```

Figure 7: Hypothetical set of association rules mined from base model

The first rule in Figure 7 suggests creating a new transfer model with an additional skill for two step problems. For the second rule in Figure 7, since there is already a skill for one step problems in the parent transfer model, this rule suggests splitting that skill by the factor question type when it has a value of two.

The same process for generating transfer models from the base models is repeated for every child node of the base models. The transfer model is expanded by mapping it to the Log+Factors dataset, creating a Log+Factors+TransferModel dataset which is then inputted into a logistic regression. Next, mining for association rules in the instances in the Log+Factors+TransferModel dataset the logistic regression predicted incorrectly is done and new transfer models are created from the current transfer model based on the association rules obtained. Figure 8 shows an example of how the association rule guided search works.



	qtype_1
p0	1
p1	0

	money_2
p0	0
p1	1

	steps
p0	1
p1	2

	steps_2
p0	1
p1	0

	qtype_1	money_2
p0	1	1
p1	1	0
p3	0	1
p4	0	0

	qtype_1	qtype_3
p0	1	0
p1	0	1
p2	0	0

	split(qtype_1 by steps=1)	split(qtype_1 by steps!=1)
p0	1	0
p1	0	1
p2	0	0

First level

Qtype=1 ➔ response=0

Money=2 ➔ response=1

Steps=2 ➔ response=1

Second Level

Money=2 ➔ response=1

Qtype=3 ➔ response=1

Qtype=1 &&
steps=1 ➔ response=0

Figure 8: Example of association rule guided search

For every transfer model that is evaluated, a BIC value for the model is obtained. In addition, the overall accuracy for a model is calculated. Overall accuracy for a model is the number of correctly predicted instances (i.e. the model predicted a correct response and the actual response was correct or the model predicted an incorrect response and the actual response was incorrect) divided by the total number of instances. These two numbers help determine how good a transfer model is and are used in guiding and terminating the search process.

A path in the search space is explored until no new association rules can be found in the instances the transfer model, through a logistic regression, predicts incorrectly; if there are no instances predicted incorrectly; if no new skill can be added; or if no significant improvement is achieved from the transfer model's parent model (i.e. overall accuracy). In addition, the BIC for a transfer model is used to determine when to stop exploring one path and start exploring another. Low BIC values are desired, so when a transfer model's branch reaches a specified threshold (BIC values for transfer models differ from one dataset to another) further searching on that path will stop.

7.1 Determining Which Child Node to Expand First

Since evaluating a transfer model is the bottleneck in the search process it is desired to evaluate a minimum amount of transfer models as possible. Several methods for guiding the search process have been considered in this thesis work. The methods that have been considered are heuristic methods that assign a score to each association rule and thus have a way to sort the transfer model nodes available for expansion. The heuristics rank each transfer model node available for expansion and the best ranked node is expanded first.

The first heuristic method the search process can use in determining which transfer model to expand next is to sum up the 4 metrics for the association rule used to create the model. The four metrics for an association rule are support, confidence, lift and chi-square. For all four metrics the higher the value the better. Thus in a set of rules, the association rule with the highest sum of these four metrics could be considered the best and the transfer model created from that rule would be expanded first. This heuristic will be called the *equal-weight* heuristic.

Another way of determining which transfer model in the search space should be expanded first is to try to predict the BIC for a transfer model. The BIC for a transfer model can be potentially predicted using the four metrics (confidence, support, lift, chi-square) for the association rule used to create the model along with the transfer model's parent BIC and overall accuracy. These seven parameters (confidence, support, lift, chi-square, parent BIC, parent overall accuracy) can be used as input to a linear regression or a neural network and the output should be a prediction as to what the transfer model's BIC is. The search process for this heuristic would want to expand the transfer model that has the lowest predicted BIC in the search space. Using the linear regression to predict a transfer models BIC will be referred to as the *regression heuristic* and using an artificial neural network will be referred to as the *ANN heuristic*.

These three heuristics, *equal-weight*, *regression*, *ANN*, are explored in determining how effective they are in guiding the search process. The three heuristics are compared against each other to determine if any heuristic is better than the others.

8 Implementation

Implementation of our search process depended on two main components, S-Plus and AprioriSetsAndSequences. A considerable amount of work was done integrating these two components into our search process. The code for our search process was built off of the original code written for the blind search method. The majority of the work for integrating S-Plus with the search process had already been done for the blind search method. This work added functionality to the blind search codebase that aloud the search process to use AprioriSetsAndSequences. In addition, this work developed and implemented all the logic for creating transfer models based on the association rules obtained from AprioriSetsAndSequences as well as all the code for the heuristic searches. The ability to add new factors to be considered in the creation of transfer models was also implemented in this work. Figure 9 shows an overview of our search process. The shaded boxes in Figure 9 represent the components that this work implemented.

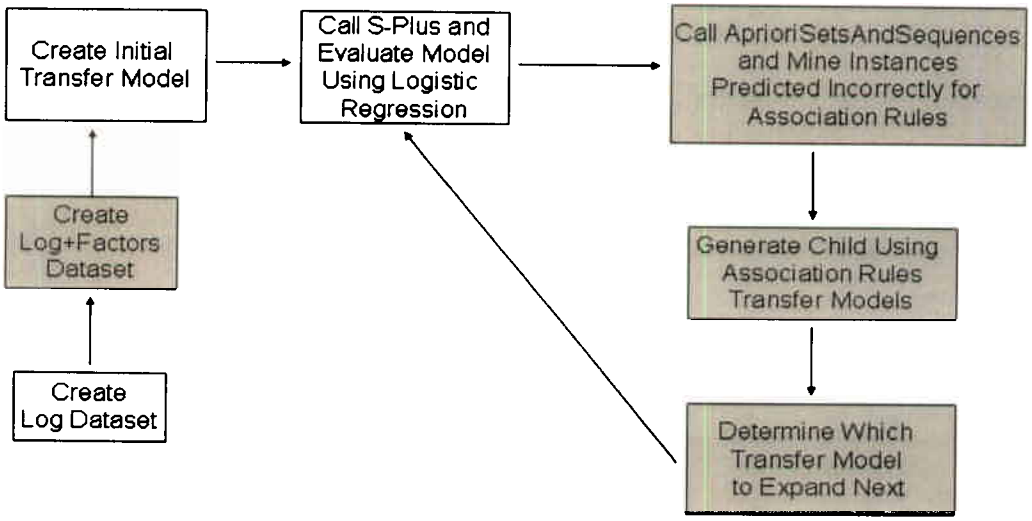

Create Initial Transfer Model
Call S-Plus and Evaluate Model Using Logistic Regression
Call AprioriSetsAndSequences and Mine Instances Predicted Incorrectly for Association Rules
Create Log+Factors Dataset
Generate Child Using Association Rules Transfer Models
Create Log Dataset
Determine Which Transfer Model to Expand Next

Figure 9: Overview of the implementation of our search process, the shaded boxes represent the components that this work created.

9 Experiments and Results

In evaluating the search process, the transfer models generated from the search process are scrutinized by domain experts to assess how logical they are. The transfer models are rated on how applicable they are to a larger set of students. Whether the transfer model illuminates any new information about why students struggle with certain problems and have an easy time with others is an important factor as well. The objective evaluation criterion for the transfer model generated is how parsimonious the model is. Models with few skills are preferred. How cognitively plausible the model is is also a criterion. If there are valid reasons for the skills being in the transfer model, then this model is preferred over other models that have high accuracy but no clear connection between the skills and problems. A transfer model's overall accuracy and BIC are also criteria used to determine how good a transfer model is.

Comparing the performance of the original blind search method to the search guided by association rules will also be done to assess how worthwhile it is to use association rules. Determining whether the search process using association rules is better requires comparing how long it takes to find a good model using association rules with how long it takes to find a good model through the brute for method. If the association method can find comparably good models in less time than the blind search method, then using association rules is beneficial.

In addition, the different heuristic ways of guiding the search process are compared. Each method is rated on how quickly and effectively it can find a good transfer model. The total number of nodes needed to be expanded by each search process before a good transfer model is found along with the total amount of time taken to find a good model are the main factors in determining which method if any is better at guiding the search process. Also, since an additional factor *money* was found to be in 40 percent of the problems asked by Ms. Lindquist, experiments will be conducted to determine how beneficial incorporating the factor *money* into transfer models is. In order to determine if taken in to consideration the factor *money* allows for better prediction as to whether a student will answer a problem correctly, transfer models containing the factor money will be compared to models that do not. If models containing the factor money are better at predicting a student's response then this suggest that money is an important factor in determining whether a student gets a problem correct.

9.1 Datasets

These experiments and results are based on a datasets obtained from Ms. Lindquist. The dataset referred to here and in the following sections are Log+Factors datasets. The datasets only contain questions from Ms. Lindquest's section 2 and contain only three factors, number of steps, question type and whether the question deals with money or not.

A question can have one or two steps and there are four different types of questions in the dataset. The four types of questions are *symbolize* (QSYM), *articulate* (QEXP), *generalize* (QGEN) and *compute* (QCOMP). The question types have been assigned the corresponding category numbers 1, 2, 3 and 4 respectively. The primary dataset used to run the majority of the experiments contains 751 instances representing data for about 42 students. Three other datasets were used each containing approximately 2000 instances representing data for approximately 60 students.

9.2 Test platform

All tests were conducted on a Windows machine. The machine has a 700MHz Intel Pentium III processor and 128 megabytes of RAM.

9.3 Simple Reference Model

As a reference for all models that are found by the search processes used in these experiments a simple transfer model where all problems are considered to be the same has a BIC value of 2383.256 and an overall accuracy of 0.554. Table 12 shows what the simple transfer model looks like.

	Skill1
p0	1

Table 12: Simple transfer model where there is only one type of problem

It is also worth noting that the majority class for response is incorrect, that is more students answered questions incorrectly (51%) than correctly (49%) in the dataset. Incorrectly answered questions only out number correctly answered questions by 25 instances though. There are 388 incorrect instances and 363 correct instances in the dataset.

9.4 Comparison between Blind Search Method and Our Search Algorithm

It is import to know that our search process can find good transfer models in the search space. The only way to really know if our search process is finding good models is to evaluate all possible models in the search space. This of course is exactly what the blind search algorithm does. The blind method uses a depth first function for traversing the search space to save on memory. Since it is estimated that there are over 1000 different transfer models at level 3 in the search space, the blind search was restricted to a depth of 3. The search process was run over the dataset for three days and after three days only 475 of the estimated 1,000+ models had been evaluated. Due to time constraints the blind search algorithm was stopped after three days. It would have most likely run for several more days until the entire search space at depth three was explored. Although the entire search space was not explored some good transfer models were found. Figure 10 and Figure 11 show the two best transfer models found by the blind search algorithm. What makes a transfer model good is primarily determined by the transfer model's BIC. The lower the BIC the better. In addition, the number of skills a model has is important. If two models have BIC values that are very close, then if one model has fewer skills than the other, the one with the fewer skills is considered to be better. Also, the complexity of the skills in a transfer model is used to determine how good a model is. Models with simple, easy to understand skills are preferred over models with complex skills.

	split(steps by qtype=3)	split(split(steps by qtype!=3) by money=2)	split(split(steps by qtype!=3) by money!=2)
p0	0	0	2
p1	0	0	1
p2	1	0	0
p3	0	2	0
p4	0	1	0
p5	2	0	0

BIC = 2364.106
Overall Accuracy: 0.679

Figure 10: Blind search- model 1

	Steps_1	qtype_2	money_1
p0	0	0	1
p1	1	1	1
p2	1	0	1
p3	0	1	1
p4	0	0	0
p5	1	0	0
p6	1	1	0
p7	0	1	0

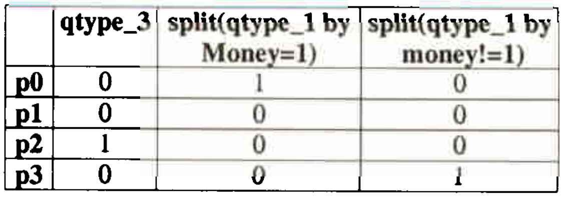
BIC = 2362.095
Overall Accuracy = 0.676

Figure 11: Blind search- model 2

Both Models found by the blind search have a low BIC and a high overall accuracy which is good. The second model can be considered to be a little better than the first model since the skills in model 2 are simpler than those in model 1. The first skill in model 1 is read as *steps* split by factor *qtype* (question type) when *qtype* is equal to three (*generalize*). The second skill is *steps* split by the factor *qtype* when *qtype* is not equal to three split by the factor *money* when *money* is equal to two. In model 2, all the skills in the model are simple skills (i.e. no splits involved). The first skill for example indicates which problems are one step problems.

Our search process using association rules to prune the search space was run on the same data as the blind search. To compare our association search method with the blind search method, our association search method was run using a depth first algorithm. For this experiment no heuristic was used to guide our depth first search algorithm. Here we were just trying to see if our search algorithm could find good transfer models in a more reasonable amount of time then the blind search. Our results show that our search algorithm does. Here we should note that when mining association rules one can specify how many rules the apriori algorithm should return. Thus, when using association rules to guide the search process one can control the maximum number of child nodes a node can have (branching factor) by specifying the number of rules the apriori algorithm returns. For this experiment we did two runs of our search algorithm. One where we set the number of rules to be returned to five and the other to ten. We will refer to the first run as AR prune (b=5) and the second run as AR prune (b=10). Both runs had a maximum depth limit of three.

For the AR prune (b=5) run there were only 28 nodes in the entire search tree. This is a considerable reduction compared to the 1000+ nodes estimated to be in the blind search tree at depth three. The AR prune (b=5) run took a total of 720 seconds (12 minutes) to evaluate all the models in the search tree. The two best models found in the AR prune (b=5) run are shown in Figure 12 and Figure 13.



	qtype_3	split(qtype_1 by Money=1)	split(qtype_1 by money!=1)
p0	0	1	0
p1	0	0	0
p2	1	0	0
p3	0	0	1

BIC = 2374.6
Overall Accuracy: 0.680

Figure 12: AR prune (b=5) model 1

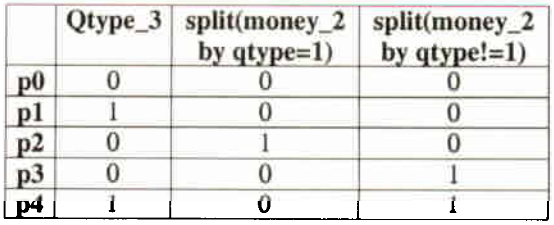
	Money_1	Qtype_3
p0	1	0
p1	1	1
p2	0	0
p3	0	1

BIC = 2366.548
Overall Accuracy = 0.662

Figure 13: AR prune (b=5) model 2

For the AR prune (b=5) run we did not get the same exact models obtained by the blind run. However, the models that we obtained are just as good if not better. The model in Figure 12 has a slightly higher overall accuracy then the two best models obtained from the blind search (Figure 10 and Figure 11). The transfer model shown in Figure 13 has a BIC value and overall accuracy very close to that of the transfer model shown in Figure 11.

For the AR prune (b=10) run we were able find an additional good transfer model that is slightly better than the two transfer models found in the AR prune (b=5) run which is shown in Figure 14. Overall there were 62 nodes in the AR prune (b=10) search tree and it took 30 minutes to evaluate all these nodes. The slightly better transfer model shown in Figure 14 is very similar to the one shown in Figure 12 except that it has a much lower BIC.



	Qtype_3	split(money_2 by qtype=1)	split(money_2 by qtype!=1)
p0	0	0	0
p1	1	0	0
p2	0	1	0
p3	0	0	1
p4	1	0	1

BIC = 2366.949
Overall Accuracy = 0.672

Figure 14: AR prune (b=10) model

Thus, from this experiment one can conclude that by using our search algorithm which uses association rules to prune the search space, one can find very good transfer models in significantly less time than using the blind search.

9.5 Comparison of Heuristic Ways to Guide Our Search Process

Three different ways were proposed as a means of guiding our search process i.e. determining which node to expand next. The first method was to add the metrics for the association rules used to create each transfer model (*equal-weight heuristic*). The second method was to use a linear regression to predict a transfer model's BIC (*regression heuristic*). The third method was to use a neural net to predict a transfer model's BIC (*ANN heuristic*). The inputs for both the linear regression and the neural net used to predict a transfer model's BIC were the metrics for the association rule used to create the model, support, confidence, lift and chi-square along with the parent transfer model's BIC and overall accuracy. Both the linear regression and the neural net were trained on data obtained from running our search algorithm using association rules with no heuristic with a maximum depth limit of four. The three heuristic were used with a depth first algorithm. Figure 15 shows an example of how the depth first search algorithm using the heuristics traverses the search tree.

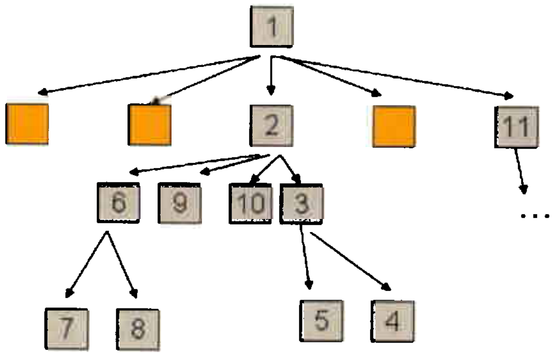


Figure 15: Example how the depth first search algorithm using heuristics traverses the search tree

All three of heuristics were tested and compared to our search algorithm running without any guidance along with the blind search method. For this experiment the maximum depth limit was set to four. Figure 16 shows a comparison of how each search method performed. Figure 17 shows how just the heuristic search method performed in comparison to each other.

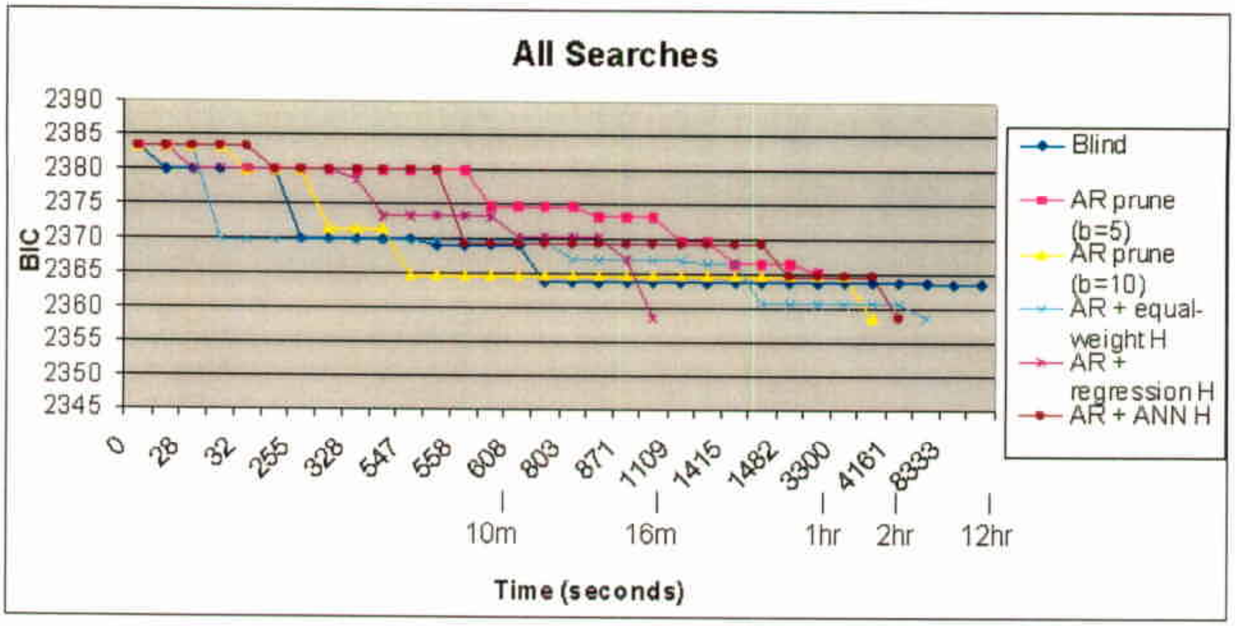


Figure 16: Comparison of heuristics on dataset1

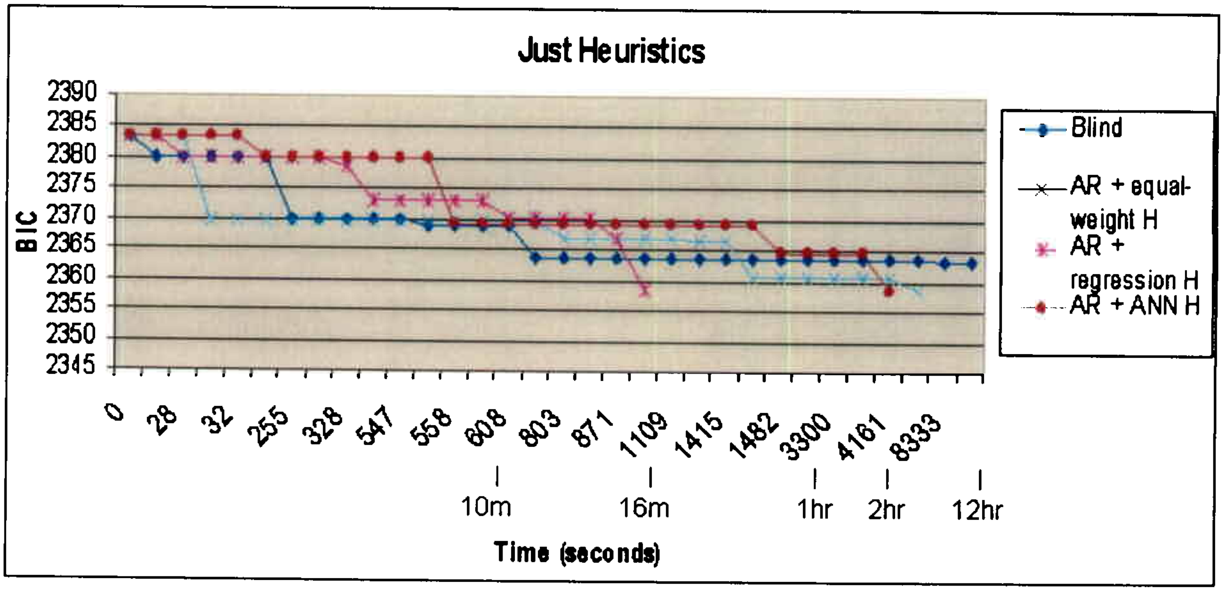
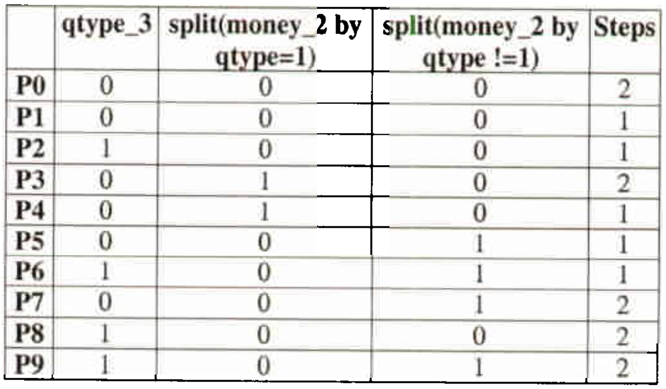


Figure 17: Comparison of just the heuristic searches on dataset1

Figure 16 and Figure 17 are graphs of BIC value versus time. Each search that was run kept track of the best transfer model it had seen so far. Thus, Figure 16 and Figure 17 show how quickly each search method was able find the transfer model with the lowest BIC in the search tree.

As can be seen in Figure 16 using a linear regression as a heuristic for our search process using association rules performed the best. With a linear regression guiding our search process the process was able to find the best transfer model in the search space in only 985 seconds (16 minutes). The best model in the search tree is shown below in Figure 18. The other two heuristics, *equal-weight* and *ANN* did not perform nearly as well. Both of these two methods of guiding the search took roughly two hours to find the best BIC in the search tree. Since these two heuristic performed relatively the same to the AR prune (b=10) run which did not use any heuristics it can be determined that neural network and adding the association rule metrics are not good heuristics for guiding the search. However, all search methods performed better than the blind search method. The blind search method was unable to find the transfer model with the lowest BIC even after

12 hours. The other search methods also have steeper slopes and their plateaus are shorter than the blind search method. That is, they move faster than the blind search towards good transfer models.



	qtype_3	split(money_2 by qtype=1)	split(money_2 by qtype !=1)	Steps
P0	0	0	0	2
P1	0	0	0	1
P2	1	0	0	1
P3	0	1	0	2
P4	0	1	0	1
P5	0	0	1	1
P6	1	0	1	1
P7	0	0	1	2
P8	1	0	0	2
P9	1	0	1	2

BIC = 2358.484
Overall Accuracy = 0.694

Figure 18: Best model found with maximum depth limit set to 4

9.6 Heuristics Performance in a Greedy Search

The performance of the three heuristic for guiding the search were tested using a greedy search algorithm as well. The same dataset was used for the greedy experiment as was used for the depth first experiment. The maximum depth limit was set to four again. Figure 19 shows an example of how the greedy search algorithm works. Figure 20 shows a graph of the performance of the three heuristics using a greedy search algorithm.

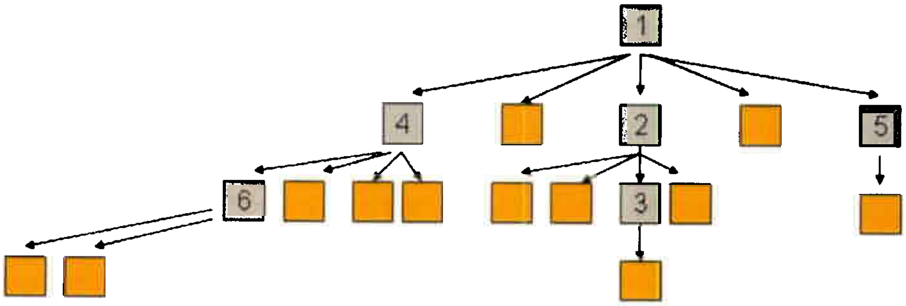


Figure 19: Example of search process using greedy algorithm

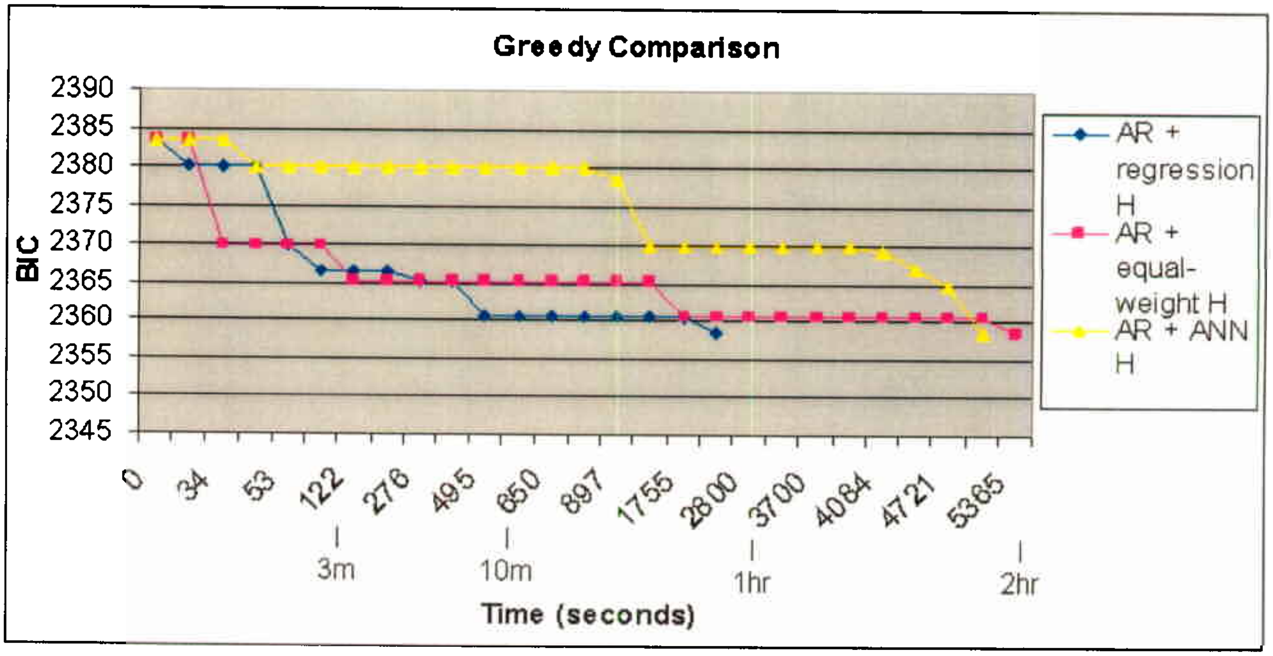


Figure 20: Comparison of heuristic methods using a greedy search algorithm on dataset1

As can be seen in looking at Figure 20, the heuristic of using a linear regression to predict a transfer models BIC and expanding the transfer model with the lowest BIC first performs well using a greedy search algorithm. The linear regression heuristic was able to reach a transfer model with a BIC of around 2360 within 8 minutes. The other heuristic methods did not perform as well. Both have very long plateaus where they are stuck on one BIC value before finding a transfer model with a lower BIC value. Here it can be concluded that the AR + regression heuristic in combination with a greedy search algorithm is the best search out of all the different searches that were run.

9.7 Evaluating Our Search Algorithm using Different Datasets

To better evaluate our search algorithm, the performance of our algorithm was compared to that of the blind search using different datasets. Three datasets were used. Each dataset contained approximately 2000 instances represent data for about 60 students. Since the AR + regression heuristic search performed the best in the previous experiments only this search was run against the three different datasets. The AR + regression heuristic was run with the greedy search algorithm. For both the blind search and the AR + regression heuristic search the maximum depth limit was set to 4 and the maximum amount of time the searches were run for was 2 hours. For the AR + regression heuristic search, branching was set to 5 and the same weights for the regression that were learned from the dataset with 751 instances (used in previous experiments) were used. Figure 21, Figure 22, and Figure 23 show the performance of the AR + regression heuristic search compared to the performance of the blind search for each of the three datasets.

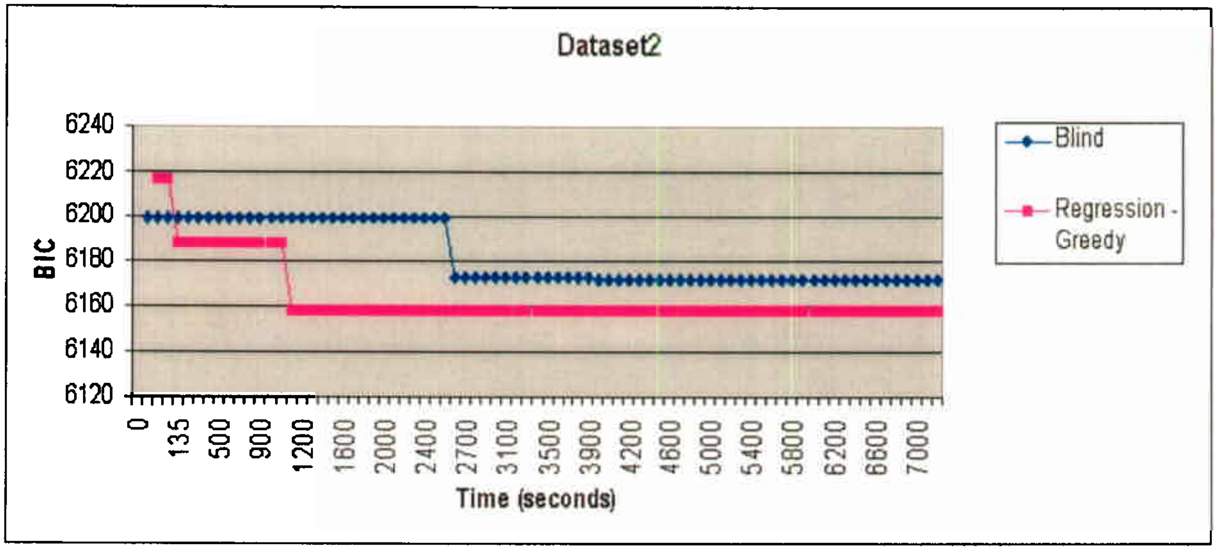


Figure 21: Comparison of AR + regression heuristic search and blind search performance on dataset 2

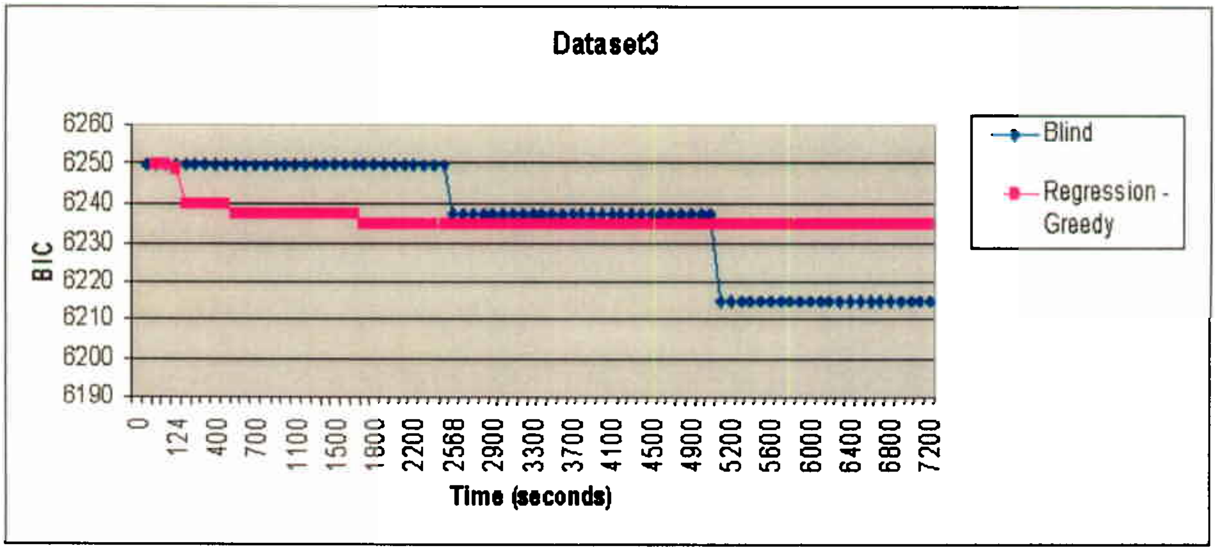


Figure 22: Comparison of AR + regression heuristic search and blind search performance on dataset 3

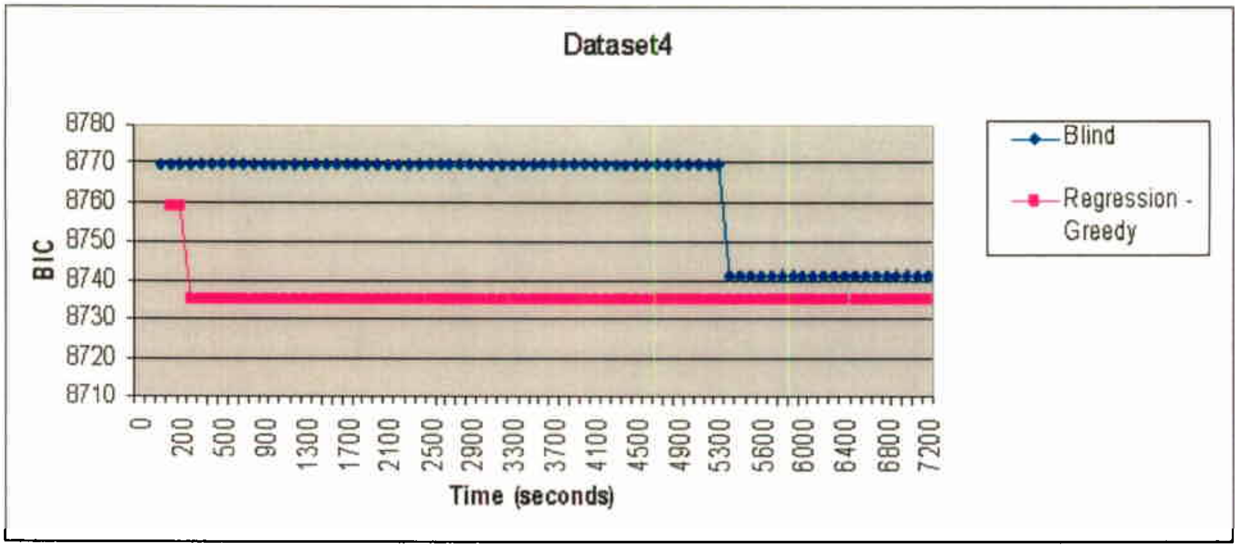


Figure 23: Comparison of AR + regression heuristic search and blind search performance on dataset 4

As can be seen in looking at Figure 21, Figure 22, and Figure 23 our search process using association rules to prune the search tree, and a greedy search based on a heuristic given by a linear regression formula to determine the order in which search tree nodes are explored, performs considerably better than the blind search method. In Figure 22, the AR + regression heuristic search starts out better than the blind search method. However, around an hour and a half into the search against the second dataset, the blind search method finds a significantly better model. It is worth noting that this model is an outlier compared to the other models in the search tree. Possible reasons that our search process did not find this significantly better model in the second dataset might be because the branching was set too low or the weights for the linear regression used to the guide the search were not trained properly for this dataset. Nonetheless our search process performed significantly better than the blind search method for 3 out of 4 datasets.

9.8 Evaluating Money as a Factor worth Using

For all the models that this work found which were considered the best models, each of them contained a skill that involved *money*. Since transfer models with a skill for *money* out performed models without a skill for *money*, this suggests that the presence of some reference to money in a problem is an important factor when determining whether a student will answer the problem correctly or not. In examining the regression equations outputted by S-Plus, it is unclear though, if the presence or the absence of a reference to money in a question increases the likelihood that the student will get the question correct.

10 Conclusions

In conclusion the search algorithm that this work has created allows for good transfer models to be found quickly. If one wants to find *the* best transfer model in the search space and one is willing to wait several days or even weeks before it is found then the blind search method is sufficient. However, if one would rather wait a couple hours at most for a reasonable transfer model then our search algorithm is the method of choice. By using association rules to prune the search space and using a heuristic such as linear regression, good transfer models can be found in a reasonable amount of time.

Since all the top transfer models that we found while conducting our searches included a skill for money in them it can be determined that *money* is an import factor for predicting which problems a student will have a hard time answering correctly and which problems a student will have an easy time answering correctly. This work did not look into whether problems that involve money are easier or harder to solve than problems that do not involve money, but since a skill for money does appear in our transfer models this suggests that students will have an easier time with one of these types of problems.

10.1 Future work

Future work for this project would be to test our search process against a greater variety of datasets. This would help determine how our search process works overall. In addition, there are plenty of other heuristics that could potentially improve the performance or our search algorithm. More work in determining what heuristics other than linear regression, neural networks and weighted sum could be explored. Also there are different search algorithms to consider such as A-star, least discrepancy, and many more. In addition, different ways of evaluating and comparing one transfer model to another could be investigated.

Work on optimizing the code for the search algorithm is something that could also be done. Right now one of the main problems with the code is that it consumes a considerable amount of memory. If the amount of memory that the search process uses could be reduced then the search process could be run for longer periods of time or explore deeper levels of the search tree.

11 References

[AIS93] R. Agrawal, T. Imielinski and A. Swami: Mining Association Rules between Sets of Items in Large Databases, *Proc. of the 20th VLDB Conference*, pp. 207-216, 1993.

[AS94] R. Agrawal and R. Srikant: Fast Algorithms for Mining Association Rules, Proc. of the *ACM SIGMOD Conference on Management of Data*, pp. 487-499, 1994.

[B84] B. Bloom : The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring, *Educational Researcher*, 13, pp. 4-16, 1984.

[CA95] A. Corbett, J. Anderson: *Knowledge tracing: Modeling the acquisition of procedural knowledge*, Kluwer Academic Publishers, 1995.

[HC03] N. Heffernan, E. Croteau: A Methodology for Evaluating Predictions of Transfer and an Empirical Application to Data from a Web-Based Intelligent Tutoring System: How to Improve Knowledge Tracing in Dialog Based Tutors, tech report <WPI-CS-TR-03-27> Dept. of Computer Science,WPI, 2003.

[HK03a] N. Heffernan: *Intelligent Tutoring Systems are Missing the Tutor: Building a More Strategic Dialog-Based Tutor*, phd thesis, CMU, 2001.

[HK03b] N. Heffernan, K. Koedinger: A Developmental Model for Algebra Symbolization: The Results of a Difficulty Factors Assessment, *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*, pp. 484-489, Hillsdal, NJ, Erlbaum, 1998.

[JKT00] B. Junker, K. Koedinger, M. Trottini: Finding improvements in student models for intelligent tutoring systems via variable selection for a linear logistic test model. Presented at *Annual North American Meeting of Psychometric Society*, Vancouver, BC, Canada, 2000.

[KLEG92] S. Katz, A. Lesgold, G. Eggan, and M. Gordin: Modeling the Student in Sherlock II, *Artificial Intelligence in Education*,1992 3(4), 495-518.

[KAHM97] K. Koedinger, J. Anderson, W. Hadley and M. Mark : Intelligent Tutoring Goes To School in the Big City, *International Journal of Artificial Intelligence in Education*, 8, pp. 30-43, 1997.

[L04] P. Laxminarayan : *Exploratory Analysis of Human Sleep Data*, Master's Thesis, Dept. of Computer Science, WPI, 2004.

[LWMG93] M. Lepper, M. Woolverton, D. Mumme and J. Gurtner : *Motivational Techniques of Expert Human Tutors: Lessons for the Design of Computer-Based Tutors*, Lawrence Erlbaum Associates, pp 75-105, 1993.

[OH03] L. Orlova and N. Heffernan: Learning Factors Talk, presentation, Artificial Intelligence Research Group, Dept. of Computer Science, WPI, 2003.

[P04] K. Pray: *Mining Association Rules from Time Sequence Attributes*, Master's Thesis, Dept of Computer Science, WPI, 2004.

[WF00] I. Witten and E. Frank: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations,* Morgan Kaufmann Publishers, 2000.