

APPROVAL SHEET

Title of Thesis: Test-cost Sensitive Regression for Planner Runtime Prediction

Name of Candidate: Brandon Scott Wilson
Master of Science, 2008

Thesis and Abstract Approved: _____
Dr. Marie desJardins
Associate Professor
Department of Computer Science and
Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: Brandon Scott Wilson.

Permanent Address: 11103 Mexico Farms Road, Cumberland, Maryland 21502.

Degree and date to be conferred: Master of Science, December 2008.

Date of Birth: April 14, 1984.

Place of Birth: Cumberland, MD.

Secondary Education: Fort Hill High School, Cumberland, Maryland.

Collegiate institutions attended:

University of Maryland Baltimore County, Bachelor of Science, 2006.

Major: Computer Science.

Professional publications:

Mark Roberts, Adele Howe, Brandon Wilson and Marie desJardins, "What makes planners predictable?," *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, forthcoming.

Brandon Wilson and Marie desJardins, "Forming stable, overlapping coalitions in an open multi-agent system," *Working Notes of the AAAI Fall Symposium on "Regarding the 'Intelligence' in Distributed Intelligent Systems*," November 2007.

Professional positions held:

Research Assistant, CSEE Department, UMBC (June 2007 – August 2008).

Teaching Assistant, CSEE Department (August 2005 – June 2007).

Java Developer, Agnik LLC (June 2006 – August 2006).

ABSTRACT

Title of Thesis: Test-cost Sensitive Regression for Planner Runtime Prediction

Brandon Scott Wilson, Master of Science, 2008

Thesis directed by: Dr. Marie desJardins, Associate Professor
Department of Computer Science and
Electrical Engineering

In many domains, such as medical diagnosis, obtaining the complete set of feature values for a test instance is impractical due to the costs associated with the features. Test costs can arise in various forms depending on the problem domain. For instance, in the medical domain, the test costs are typically monetary. Traditional machine learning algorithms focus on the single objective of minimizing error, assuming all features are “free,” even though they often are not. Learning in the presence of test costs introduces a second objective for the learner to satisfy: minimizing cumulative test costs.

This thesis focuses on solving the dual-objective learning problem for regression using a greedy feature acquisition approach. The approach in this thesis sequentially acquires features of a test instance in the order that is expected to provide the greatest reduction in prediction error per unit cost. This method can be applied to any regression algorithm given a dataset and the test costs associated with each feature. With this approach, greater accuracy can be achieved by acquiring less than the complete set of features for a test instance. Experimental analysis compares the benefits of cost-sensitive attribute acquisition against that of random and cheapest-first selection.

Test-cost Sensitive Regression for Planner Runtime Prediction

by

Brandon Scott Wilson

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
2008

To my parents, Mel and Diana, for their support and encouragement in all of my endeavors.

ACKNOWLEDGMENTS

I would first like to thank my advisor, Marie desJardins, for her guidance and patience over the past two years. She encouraged me to explore multiple areas of interest, teaching me the proper method of conducting research along the way. Working with her has been a truly enjoyable experience.

I would like to thank Mark Roberts and Adele Howe for introducing me to the general problem of planner performance analysis. They also provided informative feedback throughout the progress of my research.

I would also like to thank the members of my thesis committee, Tim Oates and Fusun Yaman. I would especially like to thank Fusun Yaman for being an unofficial advisor to my research.

I would also like to thank all of the friends and family who have provided me with the support to complete this thesis. I would especially like to thank my parents, Mel and Diana, who have always supported and encouraged my academic endeavors. A final thank you to Anusha Natarajan for providing me with support during the toughest times while completing this thesis.

This research was supported by NSF ITR grant #0325329.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGMENTS	iii
LIST OF FIGURES	vi
LIST OF TABLES	vii
Chapter 1 INTRODUCTION	1
1.1 Active Feature-value Acquisition	2
1.2 Problem Definition and Motivation	2
1.3 Contribution	3
Chapter 2 BACKGROUND AND RELATED WORK	5
2.1 Notation	6
2.2 Predicting Planner Performance	7
2.3 Approaches to Feature Acquisition for Classification	8
2.3.1 Feature Acquisition During Training	8
2.3.2 Feature Acquisition During Testing	13
2.4 Regression with Non-uniform Prediction Costs	15

Chapter 3	THE TEST-COST SENSITIVE REGRESSION FRAMEWORK	17
3.1	Building a TCSR Model	17
3.2	Selecting Features and Making Predictions with a TCSR Model	19
3.3	Accounting for Multiple Models with TCSR	21
Chapter 4	RESULTS	25
4.1	Data	25
4.2	Methodology	26
4.3	Single-model Evaluation	29
4.3.1	Evaluation with True Test Costs	29
4.3.2	Evaluation with Random Test Costs	33
4.4	Multiple-model Evaluation	40
Chapter 5	DISCUSSION AND CONCLUSION	42
5.1	Advantages and Disadvantages of TCSR	42
5.2	Sources of “Bad” Gain	43
5.3	Future Work	44
5.4	Conclusion	46
Appendix A	SINGLE-MODEL PERFORMANCE GRAPHS	47
	REFERENCES	62

LIST OF FIGURES

3.1	A graphical representation of the gain, as utilized in single-model operation.	21
3.2	A graphical representation of the gain, as utilized in multiple-model operation. <i>Planner 2</i> is currently predicted as the fastest planner.	23
3.3	The selection and prediction algorithm for TCSR operating in single model mode. .	24
4.1	The performance of TCSR on datasets on planners that exhibit characteristics that are consistent with TCSR assumptions (using the true cost assignment).	31
4.2	The performance of TCSR on planners that exhibit characteristics that violate the assumptions of TCSR (using the true cost assignment).	34
4.3	Performance evaluation of TCSR on planner dataset that exhibit characteristics that are consistent with the TCSR assumptions (using random cost assignment).	36
4.4	Performance evaluation of TCSR on planner dataset that exhibit fail to uphold the assumption underlying the TCSR method (using random cost assignment).	38
4.5	The performance graph for TCSR operating in multiple-model mode (using the true cost assignment).	40
A.1	The single-model performance graphs for each both true and random cost assign- ments by planner.	47

LIST OF TABLES

2.1	The 27 planners utilized studied by Roberts and Howe and their corresponding best success and runtime learning algorithms. Table values courtesy of Mark Roberts.	9
2.2	Descriptions of the collected features, courtesy of Mark Roberts.	10
4.1	The subset of planners included in the experiments.	26
4.2	Costly features of the planner portfolio	27

Chapter 1

INTRODUCTION

A user with a planning problem must decide which planner, from the vast inventory of available AI planners, will best solve their problem. Typical users do not have the appropriate background to make this choice. Roberts and Howe (2006) proposed the use of an algorithm portfolio to automate this task. Using learned models of success and runtime, their portfolio selects a subset of planners that are likely to solve the problem efficiently. Unfortunately, not all of the features used by the learned models can be collected at equal cost (in terms of computational time). In some situations, it may even take longer to obtain all of the features than it would to run the planner and obtain the true values of success and runtime. The planner portfolio is a good example of the more general problem addressed in this thesis: predicting a continuous class variable (i.e., performing regression) while minimizing the costs accrued from feature acquisition at test time.

The Test-Cost Sensitive Regression (TCSR) approach proposed in this thesis is a general, greedy feature acquisition algorithm that selects a subset of features to acquire at “test time” based on the expected change in prediction arising from acquiring each feature. The result of applying TCSR to the planner portfolio is a set of models that maximize predictive accuracy while minimizing the cost of prediction.

1.1 Active Feature-value Acquisition

Active feature-value acquisition, in the context of machine learning, is the process of selectively obtaining unknown feature values in order to make a better prediction or build a better predictive model. Obtaining additional features is generally not free (in either the computational or monetary sense); therefore, a cost is associated with obtaining the value of an unknown feature. The objective of this modified learning problem is to maximize the accuracy of the learned model, while minimizing the cost accrued by acquiring features along the way.

Recent work on active feature acquisition has resulted in two types of acquisition methods that minimize acquisition cost for classification tasks (Melville *et al.* 2004; 2005; Yang *et al.* 2006; Chai *et al.* 2004). The two types correspond to the two phases of learning: training and testing. Each of these methods has been shown to significantly reduce the feature acquisition cost accrued during the phase for which they were designed.

Methods that are designed for the training phase are typically provided with a set of incomplete training instances and a set of complete test instances. The objective is to acquire features for the incomplete instances such that the predictive accuracy of the final model is maximized and the accrued acquisition cost is minimized. Conversely, methods for the testing phase accept a set of complete training instances and a set of incomplete test instances as input. The objective is still minimization of acquisition cost with maximal predictive accuracy, but this optimization is performed for individual test instances. Since the response time when encountering new instances is the most critical challenge for the planning portfolio, this thesis focuses on reducing cost during the testing phase.

1.2 Problem Definition and Motivation

Acquisition problems occur in regression domains, just as they do in classification domains. Unfortunately, existing methods for minimizing feature acquisition cost for classification are not

directly applicable to regression. To my knowledge, feature acquisition in the context of regression has not yet been investigated. Therefore, this thesis focuses on feature acquisition for regression.

Melville *et al.* (2005) introduced the concept of estimating the expected utility of a feature and making feature acquisition decisions based on the estimates. Their interpretation of utility is a measure of gain in classification accuracy per unit cost. Although they applied this concept to training classification models, the same approach can be taken to model acquisition costs for regression at test time. The TCSR approach proposed in this thesis is an adaptation of the estimated expected utility method for regression problems.

The choice of acquiring or not acquiring a feature at test time depends directly on the contribution of that feature to the overall model accuracy and the cost associated with acquiring the feature. The TCSR algorithm presented in this thesis assumes that knowledge of the true value of a feature will result in a better prediction than any estimated value. Based on this assumption, TCSR decides which features to acquire at test time by determining the expected change in prediction that knowledge of the unknown features would provide.

For some domains, the acquisition problem is not limited to its effects on a single model. For example, the planner portfolio consists of several models. When a new instance is encountered, the decision to acquire features is complicated, because acquiring the value of any individual feature can affect each model differently. Which feature should be acquired if each model would receive the greatest benefit from a distinctly different feature? This thesis presents a greedy solution to this question and incorporates it into the TCSR approach.

1.3 Contribution

This thesis proposes a general framework for test-cost sensitive regression that can be applied to any regression problem with non-uniform feature costs. The approach in this thesis is applicable to two general types of regression problems: those that make a prediction from a single model and

those that make a prediction from multiple models.

Both the single-model and multiple-model operation modes are evaluated in the domain of planner performance analysis. Empirical evaluation reveals that TCSR makes better acquisition decisions when the data and regression model fit a specific set of criteria. The thesis concludes with a discussion of the potential improvements to make TCSR more generally applicable and general guidelines on how to decide if a particular learning problem will benefit from the application of TCSR.

Chapter 2

BACKGROUND AND RELATED WORK

Cost-sensitive learning refers to supervised learning algorithms that attempt to minimize both accuracy and cost during model induction. Supervised learning methods have traditionally had the single objective of minimizing the error of the model, ignoring the cost incurred while building the model or making predictions. Unless otherwise stated, the term *cost* is meant in the most general sense; cost can occur in different forms and may be in various measurable units, such as monetary or temporal units. Turney (2000) presented various sources of cost during learning, including the type addressed by this thesis: test (acquisition) costs.

Traditional supervised learning techniques take a set of training instances, \mathcal{X} , and their corresponding class labels, \mathcal{Y} , as input and output a hypothesis, h , that predicts the most probable label of future instances. The features that comprise the individual instances can be continuous, integer-valued, or nominal (discrete). The task is referred to as *classification* when the label is discrete and *regression* when the label is continuous. The TCSR framework presented in this thesis is intended for the task of supervised regression where all of the input features are real-valued.

Earlier efforts to account for costs focused on the classification problem with misclassification costs (Fan *et al.* 1999; Knoll, Nakhaeizadeh, & Tausend 1994; Ting 1998). A traditional classifier assumes that *misclassification costs* – the cost of misclassifying an instance as one class when it is really another – are the same for all classification errors. Assuming uniformity of misclassification

costs can lead to poor results for many classification problems. For example, a medical diagnosis does not carry uniform costs for all errors. Consider one scenario in which the patient is incorrectly diagnosed as having a disease and another in which the patient is misdiagnosed as healthy when they actually have the disease. A standard classifier treats both cases as equal even though they are not: the latter scenario can result in the loss of life. A cost-sensitive classifier places a greater emphasis on minimizing more costly errors.

The following sections introduce the notation that is utilized in remainder of this thesis, as well as the background needed to understand the proposed approach. The related work on cost-sensitive learning in the fields of classification and regression is also introduced in this chapter.

2.1 Notation

This section introduces the notation that is used in the remainder of this thesis. The notation is loosely based on that of Chai *et al.* (2004).

$\mathcal{X} = \{x_1, x_2, \dots, x_n\}$, $x_i \in \mathbb{R}^k$, is a set of instances, where each x_i is a feature vector representing a specific instance and $x_{i,j}$ represents feature j of instance i . The set $\mathcal{Y} = \{y_1, y_2, \dots, y_n\}$, $y_i \in \mathbb{R}$, is the set of labels corresponding to the instances in the set \mathcal{X} .

$\mathcal{F} = \{f_1, f_2, \dots, f_m\}$ denotes the set of all input features for an instance. $\mathcal{K} \subseteq \mathcal{F}$ represents the set of features whose values are currently known. $\mathcal{U} = \mathcal{F} - \mathcal{K}$ is the set of remaining features that are in \mathcal{F} , but whose values are unknown. The value C_i represents the cost of acquiring feature f_i . In some situations, the cost of a subset of features is incurred as a single cost, and cannot be divided among the features. In this case, the meaning of f_i is relaxed to refer to the grouped subset of features, and C_i refers to the cost to acquire the whole subset. Also, when iterating over each possible value $v \in f_i$, v represents one possible combination of the values of the individual features of f_i .

2.2 Predicting Planner Performance

Recent work on planner performance analysis (Roberts & Howe 2006; 2007a; 2007b) has created accurate models of planner success and less accurate models of planner runtime. Roberts and Howe combined a set of planners into a portfolio, using the learned runtime and success models to direct the portfolio when solving a new problem. The idea of a planner portfolio that ranks planners and allocates computational time among them is suggested as a means of defeating the lack of generality faced by planners. For any large set of problems, a single planner will likely not be successful on all problems. Instead, one planner will work well on one set of problems while another planner will be efficient for a different set. Experimental results show the portfolio is able to solve more problems successfully than any of the constituent planners alone.

The performance models were constructed and evaluated using a set of 4727 problems from 678 domains on a collection of 27 publicly available planners (refer to Table 2.1 for a complete list of the planners). The problems were obtained from the first five International Planning Competitions (IPCs). The best learning algorithms for success and runtime were found using a brute-force search through the classifiers and regression methods available in the Weka toolkit (Witten & Frank 2005). Table 2.1 presents the best learning algorithm for success and runtime for each planner, as well as the corresponding accuracy. Success proved to be rather easy to predict using the domain and instance features, but the runtime models were significantly less accurate.

The portfolio operates in a two stage, round-robin fashion. Upon encountering a new problem, the planners are ranked in order of the probability that they will succeed given the problem features. Then, the portfolio enters the first stage of operation, in which it runs the first five planners for 10 seconds each. If a solution is found, then the process halts. Otherwise, the portfolio enters stage two and runs the top planners in order for 100 seconds each. After each pass through all of the planners, if a solution is not found, the allocated time is increased by 100 seconds and the process repeats. Using this algorithm, the portfolio proved to be more successful than any single planner.

Unfortunately, the runtime models were not reliable enough to be utilized in the portfolio. Had they been more accurate, a useful addition would be to allocate time proportional to the runtime predictions. The TCSR framework was designed to allow for more expensive features because they may be the key to better runtime models for some planners.

2.3 Approaches to Feature Acquisition for Classification

Feature acquisition algorithms have been developed for both the training and the testing phases of learning. Feature-value acquisition during training is oriented toward reducing the total cost accrued to build the most accurate model, whereas the acquisition methods for testing are primarily concerned with minimizing the cost of acquiring features for a test instance to obtain the best prediction.

2.3.1 Feature Acquisition During Training

Zheng *et al.* (2002) developed the Goal-Oriented Data Acquisition (GODA) algorithm for problems in which the features of the supplied training data can be divided into two categories: the local (i.e., base) set and the global (i.e., costly) set. They assume that the training instances are all missing the global features. The test set, however, is assumed to be complete. They also assume that the global features are obtainable as a complete set for a fixed price. Obtaining only a subset of the costly features, as required by the problem addressed in this thesis, is not permitted. GODA is designed to select a subset of the training instances for which to obtain global features with the goal of producing the most accurate model at the cheapest (acquisition) cost.

GODA sequentially selects K instances for which to acquire global information based on their overall contribution to the classifier accuracy. To measure the contribution of an individual instance, GODA imputes the missing features of the instance, temporarily adds the instance to the current set of complete instances, and evaluates the accuracy of the model built from this set of

Planner	Best Success Classifier	Classification Accuracy	Best Runtime Learner	RMSE (sec)
altalt-1.0	KStar	99.18%	KStar	3.55
blackbox-4.2	MLP	97.42%	MLP	84.79
cpt-1.0	MLP	99.38%	KStar	83.52
ff-2.3	KStar	93.40%	M5P	316.79
hsp-2.0	J48	92.78%	M5Rules	421.66
hsp-2.0-b-h1plus	KStar	96.70%	DT	333.09
hsp-2.0-b-H2Max	KStar	98.76%	KStar	368.75
ipp-4.0	NNge	96.08%	M5Rules	424.95
ipp-4.1	JRip	96.39%	M5P	383.94
lpg-1.1	PART	95.36%	M5Rules	251.64
lpg-1.2	KStar	94.12%	REPTree	269.97
lpg-td-1.0	IB1	94.95%	MLP	309.40
metric-ff2002	J48	93.81%	M5P	189.98
mips-3	NNge	95.46%	DT	373.75
optop-1.6.19	KStar	96.80%	KStar	49.04
prodigy-4.0	NNge	99.59%	KStar	170.44
r-1.1	LMT	98.97%	M5Rule	216.14
sapa-2.200406	log	99.07%	KStar	250.67
satplan-2006	IB1	94.64%	REPTree	191.16
sgp-1.0b	log	98.76%	KStar	253.23
sgp-1.0h	PART	98.87%	KStar	265.52
sgplan-2006	PART	90.62%	MLP	294.45
simplanner-2.0	PART	90.93%	KStar	145.23
snlp-1.0	MLP	99.59%	DT	134.45
stan-4	ADT	98.66%	KStar	128.98
ucpop-4.1	LMT	98.66%	DT	235.62
vhpop-2.2	MLP	99.28%	KStar	22.49

Table 2.1. The 27 planners utilized studied by Roberts and Howe and their corresponding best success and runtime learning algorithms. Table values courtesy of Mark Roberts.

Feature Name	Type	Short Description
time	numeric	Time until success or failure (range 0.0 – 1800.0 seconds)
num_predicates	numeric	Number of predicates in the domain
min_params_pred	numeric	Minimum of the number of atoms over all predicates
mu_params_pred	numeric	Average of the number of atoms over all predicates
max_params_pred	numeric	Maximum of the number of atoms over all predicates
num_invariant_pred	numeric	The number of disunified predicates
num_operators	numeric	Number of operators in the domain
num_toggle_oper	numeric	The number of pairs of clobbering actions
min_prec_oper	numeric	Minimum of the number of predicates in the precondition over all operators
max_prec_oper	numeric	Maximum of the number of predicates in the precondition over all operators
mu_prec_oper	numeric	Average of the number of predicates in the precondition over all operators
min_post_oper	numeric	The minimum number of predicates in effects
max_post_oper	numeric	The maximum number of predicates in effects
mu_post_oper	numeric	The average number of predicates in effects
num_neg_post_oper	numeric	The number of actions with negative effects
min_neg_post_oper	numeric	Minimum of the number of negative predicates in the postcondition over all operators
max_neg_post_oper	numeric	Maximum of the number of negative predicates in the postcondition over all operators
mu_neg_post_oper	numeric	Average of the number of negative predicates in the postcondition over all operators
ratio_neg_post_oper	numeric	The ratio of actions with negative effects
num_inits	numeric	Number of items in the starting state of the world
num_objects	numeric	Number of objects in the world
num_goals	numeric	Number of goals to achieve
min_tail_actions	numeric	The minimum of the number of operators that unify to goals
mu_tail_actions	numeric	The average of the number of operators that unify to goals
max_tail_actions	numeric	The maximum of the number of operators that unify to goals

Table 2.2. Descriptions of the collected features, courtesy of Mark Roberts.

instances. The global features are acquired for the instance that contributes the most to the accuracy of the model built from the training set and it is permanently added to the set of complete training instances. After acquiring the global features of K instances, GODA returns a model induced from the complete training instances only.

An obvious disadvantage to GODA is the complexity of inducing a model for each possible incomplete instance on every iteration. Motivated by the computational demands of GODA, Melville *et al.* (2004) developed two feature-value acquisition policies (uncertainty sampling (US) and error sampling (ES)) that are applicable to the same situations as GODA, but have lower computational requirements. Both uncertainty sampling and error sampling induce models from all training instances, whereas GODA only utilizes the complete instances. Uncertainty sampling acquires complete information for the incomplete instances that have the highest *uncertainty score*. The uncertainty score is the difference between the probability of the most probable class and the probability of the second most probable class. On every iteration, global information is obtained for the m instances with the lowest uncertainty score on every iteration until a stopping criterion is met. Error sampling is a special case of uncertainty sampling in which misclassified instances are assigned an uncertainty score of -1. Again, m instances are chosen for global information acquisition after each iteration until a stopping criterion is met.

US, ES, and GODA are only applicable in situations where the values of all unknown features of an incomplete instance must be acquired together, at a fixed price. They fail to accommodate the more general acquisition problem where individual features may be assigned a cost and a subset of features may be acquired. Sampled Expected Utility (SEU) estimation was proposed by Melville *et al.* (2005) as a solution to the more general acquisition task. SEU acquires the values for the missing features in the training data that are expected to provide the most improvement in model performance per unit cost (referred to as utility).

The general feature acquisition problem is a relaxed version of the feature acquisition problem studied in previous work (Melville *et al.* 2004; Zheng & Padmanabhan 2002), where the entire set

of missing features for an instance can be acquired at a fixed price. In the general feature acquisition problem, each missing feature may have an associated cost and be acquired individually. Both TCSR and SEU are designed to solve the general acquisition problem, but SEU reduces acquisition costs during training for classifier induction, while TCSR reduces the overall costs during the testing phase of regression.

SEU measures the estimated expected utility of a random sample of all missing features. It acquires a set of b features with the highest estimated values. The estimated utility of acquiring a feature $x_{i,j}$ is computed by:

$$E(x_{i,j}) = \sum_{k=1}^K P(x_{i,j} = v_k) U(x_{i,j} = v_k), \quad (2.1)$$

where feature $x_{i,j}$ is assumed to be nominal with K possible values. $P(x_{i,j})$ is the probability that the value of $x_{i,j}$ is v_k . Any classifier that produces probability estimates for class membership can be used to estimate these probabilities. The utility of knowing that the value of $x_{i,j}$ is V_k can be calculated by:

$$U(x_{i,j} = v_k) = \frac{A(\mathcal{X}, x_{i,j} = v_k) - A(\mathcal{X})}{c_{i,j}}, \quad (2.2)$$

where $A(\mathcal{X})$ represents the accuracy of the classifier induced from the current training set and $A(\mathcal{X}, x_{i,j})$ represents the accuracy of the classifier induced from the training data plus the known value of $x_{i,j}$.

Computing the estimated utility for each possible value of each feature requires that a new model be built for each value of every unknown feature of all instances in the training set. This restricts the applicability of the approach; therefore, the authors suggest random sampling of the missing features to make the utility estimation more practical. The random sampling allows the user to specify the number of expected utilities to be computed. At each iteration of the acquisition process, only the expected utilities of the random sample are computed. A batch of missing values with the highest expected utility are chosen for acquisition from the random sample.

SEU performed well in comparison to cheapest-first and random strategies on most datasets. SEU was often outperformed by cheapest-first when the cheapest features were the most highly correlated with the class. Likewise, if all features had similar costs and contributed to classification equally, then random tended to perform well and sometimes better than SEU.

2.3.2 Feature Acquisition During Testing

Chai *et al.* (2004) proposed cost-sensitive Naive Bayes (csNB), a probabilistic feature acquisition strategy for naive Bayes classification. The objective of csNB is to minimize both acquisition costs and misclassification costs; prior work (Turney 2000; Marlon Núñez 1991; Elkan 2001; Domingos 1999) focused on either misclassification costs or acquisition costs alone, but never both. The csNB classifier learns a model from the training instances in the same way as a traditional naive Bayes classifier, differing only during the classification of unseen instances. A test instance initially has unknown values for all of the features. The csNB classifier only acquires values for features that are expected to reduce misclassification cost by at least as much as the acquisition cost of the feature. Features are acquired sequentially based on their expected total cost reduction until there are no unknown features, or until the remaining unknown features' acquisition costs outweigh their savings in misclassification costs.

The utility of acquiring an unknown feature \mathcal{U}_i is calculated by:

$$Util(\mathcal{U}_i) = Gain(\mathcal{K}, \mathcal{U}_i) - C_{test}(\mathcal{U}_i), \quad (2.3)$$

where *Gain* (Equation 2.4) is the expected change in misclassification cost after acquiring the unknown attribute \mathcal{U}_i . Subtracting the acquisition cost of $C_{test}(\mathcal{U}_i)$ from the *Gain* gives the final *Utility*, or reduction in the total cost of obtaining \mathcal{U}_i . Gain is defined as:

$$Gain(\mathcal{K}, \mathcal{U}_i) = C_{mc}(\mathcal{K}) - C_{mc}(\mathcal{K} \cup \mathcal{U}_i). \quad (2.4)$$

C_{mc} represents the expected misclassification cost, computed as a weighted sum over all possible class values. To compute the expected misclassification cost with an unknown feature \mathcal{U}_i , the value is imputed as the weighted average over all possible values of \mathcal{U}_i .

Sheng *et al.* (2006) expanded upon the work on test-cost sensitive acquisition methods with their Sequential Batch Test (SBT) algorithm. SBT acquires features in sequential *batches*, whereas other algorithms acquire features one at a time. The motivation for batch processing of features is that in many situations, there is a delay associated with acquiring each feature; furthermore, these delays may overlap. Consider the scenario of a classifying a patient as having a disease or not having a disease. The features include tests such as x-rays, blood samples, and MRIs. SBT assumes that each test has an associated cost, as well as a delay. The cost in this situation is the monetary cost of performing the test; the delay is the processing time to obtain the result. SBT suggests that obtaining tests in batches is advantageous because processing the entire batch of tests requires only as much time as the most lengthy test in the batch. For example, if the doctor requests an MRI and a blood test, which require 5 hours and 72 hours, respectively, then the total cost is 72 hours, because they were requested in the same batch. On the other hand, requesting the second test only after waiting for the value of the first test would result in a total time of 77 hours.

SBT is designed as a shell that can convert any sequential test-cost sensitive acquisition algorithm into a test-cost sensitive acquisition algorithm that acquires *batches* of features. SBT assumes that there is a function that measures the cost reduction of acquiring an attribute. Using this function, SBT acquires all features with a positive cost reduction in the first batch. The process repeats until all features have been acquired or there are no features with a positive cost reduction. SBT performed well experimentally using a cost-sensitive decision tree as the base learner. Unfortunately, it cannot currently be applied to feature acquisition problems in regression because there are no sequential solutions to apply it to.

2.4 Regression with Non-uniform Prediction Costs

Regression-based research in the field of cost-sensitive learning is limited. Early efforts were made to differentiate between the costs of under-predictions and over-predictions (Crone, Lessmann, & Stahlbock 2005). Researchers have only recently begun to acknowledge the need for more work in this area. To date, no work in the field of cost-sensitive regression has investigated the feature acquisition problem as it is proposed in this thesis.

Torgo and Ribeiro (2007) introduced a utility-based metric for model evaluation as an alternative to traditional loss functions, which ignore costs. Their work is motivated by the uniform prediction-cost assumption of traditional model evaluation metrics. This assumption is not valid for many domains, such as the task of predicting the closing value of a stock. Predicting an extremely high or low closing value is far more costly when the prediction is incorrect than predicting an average value, because extreme value predictions can induce a flurry of selling and trading, resulting in an enormous loss of profit or investment. The utility-based metric will present a more meaningful understanding of the model performance in this domain, because it penalizes mispredictions based on their relevance.

The utility of a model is measured as the difference between the total *benefit* and the total *cost* of each prediction over all of the test instances. These two metrics assume that the user defines a continuous relevance function over the target variable to indicate which values are most relevant (i.e. high and low stock prices). This individual relevance function is used to construct a bi-variate relevance function that measures the relevance of the true and predicted values. The bivariate relevance distinguishes between false negatives (predicting an irrelevant value for a relevant instance) and false positives (predicting a relevant value for an irrelevant test instance). The final cost of an individual instance is directly proportional to loss and the bi-variate relevance of the true and predicted values. This ensures that the most inaccurate and relevant mispredictions are assigned the largest costs. Likewise, the benefits of an instance is directly proportional to the relevance of the

true value of the instance and inversely proportional to the loss. This assigns the greatest benefit to the most relevant and accurate predictions.

Torgo and Ribeiro's utility-based metric was used to evaluate several different models for a stock market forecasting task. It successfully distinguished which models were better trained to predict relevant (rare) instances. Since this is only an evaluation metric, it cannot be applied to the feature acquisition problem. The costs utilized by this metric also inhibit its application to the dual-objective learning problem in this thesis. In particular, the costs are assumed to be higher for more relevant instances, which is not always true for acquisition costs. In fact, the most expensive feature could be the *least* correlated with the continuous class.

Chapter 3

THE TEST-COST SENSITIVE REGRESSION FRAMEWORK

The TCSR framework proposed in this thesis is an adaptation of the sampled expected utility (SEU) approach to acquisition cost minimization during training for classification (Melville *et al.* 2005) (see Section 2.3.1). Given a dataset, a cost assignment for each feature, and a regression algorithm, TCSR attempts to minimize the total cost accrued from mispredictions and feature acquisitions during testing. TCSR is a general framework that can be applied with any regression algorithm.

3.1 Building a TCSR Model

Before discussing the operation of a TCSR model, I will present the individual components that make up a TCSR model and how they are constructed. A TCSR model has several components: a continuous regression model, an imputation model, and a conditional probability distribution for each costly feature. The construction of each component is independent of the others. The only shared information among them is the training information.

The first component of a TCSR model is the regression model. This model is built by a user-specified regression algorithm. It is assumed that the regression algorithm is capable of receiving a set of training instances and returning the regression function, $\hat{t}(\mathcal{F})$, that best fits the instances. It

is possible that TCSR would have a greater impact on some regression algorithms, but in general TCSR does not require the use of any specific regression algorithm.

The imputation model is the next component of a TCSR model. The imputation model estimates the values of missing features when making a prediction with the regression function, $\hat{t}(\mathcal{F})$, and a subset of the input features, \mathcal{U} , are unknown. Imputing these values makes it possible to have only a single regression model that uses all features instead of having $2^{|\mathcal{U}|}$ models, one for each possible combination of features, or restricting the regression algorithm choice to only algorithms that are capable of handling instances with missing values. The imputation model is very closely related to the probability distributions discussed later. In fact, the probability distributions can be used to impute features, in which case the imputed value for a feature would be the expected value that is computed using the probability distribution.

The *knn* imputation method is utilized in this thesis (Batista & Monard 2003). When imputing a missing value in a test instance, *knn* imputation finds the k nearest neighbors to the instance, from a set of training instances, and uses the average (continuous feature) or mode (nominal feature) of the feature from the nearest neighbors as the imputed value. The *knn* imputation method is robust because it can impute both continuous and discrete variables. Also, it is a lazy learner, so it does not require a separate model to be built for each feature like other predictive imputation methods.

The final component of a TCSR model is a set of conditional probability density functions for the costly features. The density function for each costly feature, u_i , provides probability estimates for that feature given any subset of the other input features, $\mathcal{F} - u_i$. This general process of estimating a density function from a set of observed instances is known as *density estimation*. Density estimation has been studied by statisticians for years, resulting in a variety of methods ranging in complexity and accuracy from simple histogram estimators to more complex kernel density estimators and wavelet density estimators (Scott 1992; Silverman 1986). Any general density estimation method can be used with TCSR; however, the

remainder of this thesis will use the Flexible Naive Bayes classifier to estimate probability distributions (John & Langley 1995). The Flexible Naive Bayes algorithm is an improvement over traditional Naive Bayes that uses kernel density estimators to learn probability density functions for continuous input features. Traditional naive Bayes would discretize continuous features, resulting in a decrease in accuracy. Flexible Naive Bayes was selected due to its proven efficiency and accuracy.

The probability density function for each costly feature is constructed by first discretizing the costly feature in the training instances using the equal-frequency binning method, which attempts to divide the target variable into n bins, each with the same number of instances. Discretizing the training instances in this way converts the problem into a classification problem so the instances can be used to build a Flexible Naive Bayes classifier (John & Langley 1995). When a test instance is encountered, the probability distribution over the n possible feature values, given the known features as evidence, can be obtained from the classifier. The probability that an instance belongs to a class, y , will be denoted using standard conditional probability notation. For example, $p_i(y|\mathcal{K})$, will represent the probability that the value of feature u_i of an instance is y , given the values of the known features.

3.2 Selecting Features and Making Predictions with a TCSR Model

This section discusses how the components of a TCSR model work together to acquire beneficial features and reduce the overall cost of testing an individual instance. Using the probability distributions, TCSR computes the expected change in prediction value resulting from acquiring an unknown feature. TCSR then uses this as a basis for selecting features to acquire.

Acquisition choices are based solely on their *utility*. The utility for an unknown feature is an estimation of the expected change in prediction per unit cost of collecting the feature. The utility is always positive; larger values represent more change due to feature acquisition. The utility from

acquiring an unknown feature $f_i \in \mathcal{U}$ is calculated by:

$$Utility(\mathcal{K} \cup f_i) = \frac{Gain(f_i, \mathcal{K})}{C_i}. \quad (3.1)$$

The *Gain* of a feature f_i represents the expected change in prediction value and is computed by:

$$Gain(f_i, \mathcal{K}) = \int_{f_i} p(f_i | \mathcal{K}) \left| \hat{t}(\mathcal{K} \cup f_i) - \hat{t}(\mathcal{K}) \right| d_{f_i}. \quad (3.2)$$

This equation can be simplified because the probability distribution is discrete. The resulting equation is:

$$Gain(f_i, \mathcal{K}) = \sum_{v \in f_i} p(f_i = v | \mathcal{K}) \left| \hat{t}(\mathcal{K} \cup f_i = v) - \hat{t}(\mathcal{K}) \right|, \quad (3.3)$$

where p is the estimated probability distribution and f_i is the set of values produced by discretizing the feature to create the probability distribution. Note that wherever $\hat{t}(F)$ is computed and not all features are known, then the values of the features in the set \mathcal{U} are imputed by the imputation model.

Figure 3.1 provides a graphical interpretation of the gain for an unknown feature over the entire range of possible values (in the planner runtime domain). The underlying assumption is that the gain from acquiring a feature value represents a beneficial change in the prediction. The gain is the shaded area, where knowing the true value of the unknown feature results in a deviation from the current prediction. Ideally, an irrelevant feature would not affect the prediction and the gain would be zero.

The process of selecting features to acquire is a greedy process. When a new test instance is encountered, the utility for each unknown attribute is computed and the feature with the largest positive utility is acquired. The process repeats until a user-specified budget is expended, or until all features are acquired. The algorithm described in this section is displayed in Figure 3.3.

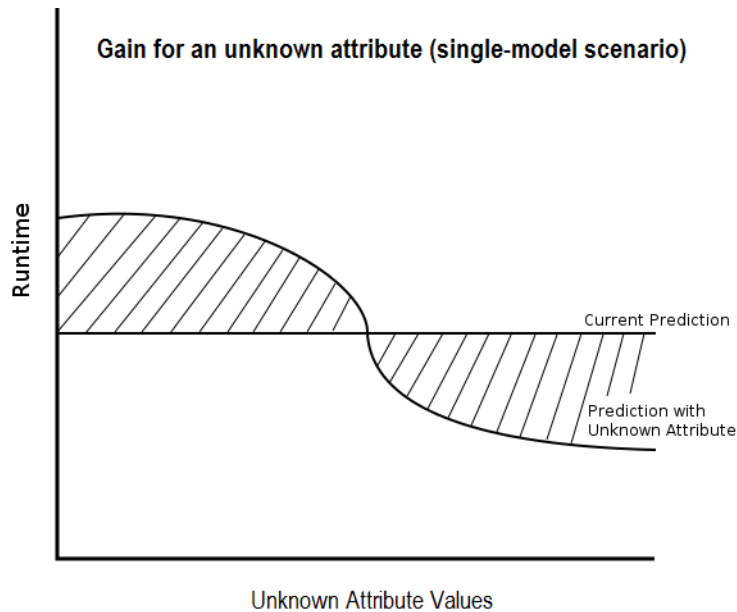


FIG. 3.1. A graphical representation of the gain, as utilized in single-model operation.

3.3 Accounting for Multiple Models with TCSR

The task for multiple-model operation is different from that of single-model operation. In single-model operation, the goal is to reduce the error of the prediction for the test instance while accruing the smallest cost. The goal of multiple-model operation, however, is to acquire attributes and accrue the smallest cost necessary to identify the model that makes the most “favorable” prediction. In multiple-model operation of the planning portfolio, the goal is to identify the planner that will solve a given problem in the shortest time. If the fastest planner is identified incorrectly, then the secondary goal is to minimize the difference between the true runtime of the predicted fastest planner and the true runtime of the actual fastest planner.

The TCSR framework, as described thus far, selects features for acquisition based solely on their expected impact on a single model. To apply the TCSR algorithm within the context of a

regression problem with multiple models, like the planner portfolio, it must acquire features based on their expected impact on multiple models. Generalizing the cost analysis of TCSR from a single model to multiple models requires a small adjustment to the utility calculation. The global utility for an unknown feature $f_i \in \mathcal{U}$ is computed by:

$$GlobalUtility(\mathcal{K} \cup f_i) = \sum_{v \in f_i} \frac{p(f_i = v | \mathcal{K}) (t_{best} - t_{pred})}{C_i}, \quad (3.4)$$

where t_{best} is the prediction of the best model with only the known features and t_{pred} is the prediction of the best model with the known features and f_i . This guides the acquisition of features toward the features that make the predictions more likely to correctly identify the best model. To adapt the single-model algorithm in Figure 3.3 to accommodate multiple models, line 9 is the only required change, where the value on the right side of the assignment is replaced by the global utility equation (Equation 3.4).

Figure 3.2 depicts the multiple-model gain in the planner performance domain. There are three planners in the graph and planner 2 is predicted to be the fastest planner without knowing the true value of the unknown attribute. Notice that knowing the value of the attribute makes planner 1 and planner 3 the predicted fastest planners over a portion of the attribute range. The shaded area represents the gain in the areas where planner 2 is no longer the predicted fastest planner when the unknown attribute value is known.

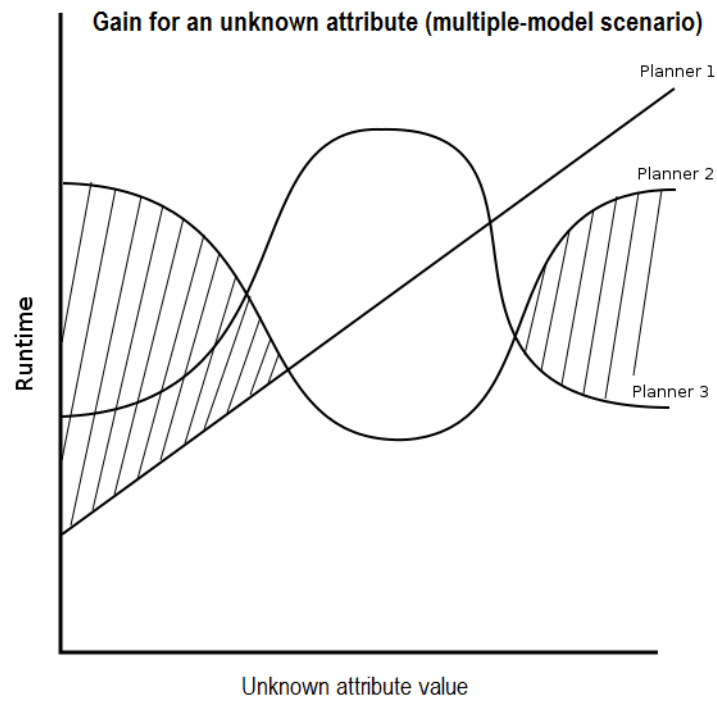


FIG. 3.2. A graphical representation of the gain, as utilized in multiple-model operation. *Planner 2* is currently predicted as the fastest planner.

```

input :  $x$  (the test instance),  $c$ ,  $\mathcal{K}$ ,  $\mathcal{F}$ ,  $b$  (the budget)
output: the prediction,  $y$ , for instance  $x$ 
1  $\mathcal{U} = \mathcal{F} - \mathcal{K}$ ;
   /* Acquire features until all values have been acquired or
   the budget is expended */
2 while  $\mathcal{U} \neq \emptyset$  and  $b > 0.0$  do
   /* Remove unknown features that exceed the budget */
3   for  $u_i \in \mathcal{U}$  do
4     if  $b - c_i < 0.0$  then
5        $\mathcal{U} = \mathcal{U} - u_i$ ;
6     end
7   end
   /* Compute the utility of each feature in  $\mathcal{U}$  */
8   for  $f_i \in \mathcal{U}$  do
9      $util[i] = \frac{Gain(f_i, \mathcal{K})}{c_i}$ ;
10  end
   /* Acquire the feature with maximum utility that does not
   exceed the budget -- Stop if all available feature
   exceed the budget */
11   $i = \arg \max_i \{util[i]\}$ ;
12   $\mathcal{U} = \mathcal{U} - f_i$ ;
13   $\mathcal{K} = \mathcal{K} + f_i$ ;
14   $b = b - c_i$ ;
15 end
   /* Impute the remaining unknown features */
16  $x = ImputeUnknown(\mathcal{U}, x)$ ;
17 return  $\hat{t}(x)$ ;

```

FIG. 3.3. The selection and prediction algorithm for TCSR operating in single model mode.

Chapter 4

RESULTS

Experiments were conducted to evaluate the effect of TCSR on the planner performance domain. The single-model operation was evaluated using a subset of the data collected by Roberts and Howe (2006) using the true cost assignment values as well as a random cost assignment. The multi-model operation was analyzed in the context of a simple portfolio strategy of determining the fastest planner.

4.1 Data

The data used in these experiments is a subset of the data collected by Roberts and Howe (2006; 2007a) during their studies of planner performance. The subset consists of the more challenging problems. A problem is considered to be challenging if at least one but no more than three planners can solve it, or if it has a median runtime of greater than one second. Using the more challenging subset of problems focuses the runtime prediction problem on the problems that are the most difficult to predict. The set of planners was also reduced to the most successful planners (those that were able to solve at least 5% of the challenging problems). Roberts and Howe justified the exclusion of the remaining planners because they are older and cannot solve newer problems. Therefore, they are subsumed by the newer planners. Table 4.1 shows the chosen subset of planners and the number of problems from the total set on which they were successful. The final

Planner	Number of Successful Runs
blackbox-4.2	87
ff-2.3	611
hsp-2.0	190
hsp-2.0-b-h1plus	69
ipp-4.0	197
ipp-4.1	207
lpg-1.1	154
lpg-1.2	209
lpg-td-1.0	583
metric-ff_2002	585
mips-3	104
optop-1.6.19	67
r-1_1	94
simplanner-2.0	197

Table 4.1. The subset of planners included in the experiments.

dataset consists of 1329 problems and 14 planners.

The features were partitioned into base and costly feature sets based on the time required to collect them. The base features are considered to be any feature that is computable in the same amount of time as is required to load the domain. There are five base features: *num_predicates*, *num_objects*, *num_inits*, *num_goals*, and *num_operators*. The remaining features are considered part of the costly set, even if their cost is minute. The costly features are divided into feature groups in Table 4.2. Features that are grouped together represent a set that must be acquired together at a fixed price, or not at all. Recall that a feature group refers to a group of features that can be obtained at a single cost.

4.2 Methodology

The experiments evaluate the performance of the TCSR framework in the single-model and multiple-model modes of operation. Within each mode, the TCSR acquisition choices are com-

Attribute 1	min_params_pred max_params_pred mu_params_pred
Attribute 2	min_prec_oper max_prec_oper mu_prec_oper
Attribute 3	min_post_oper max_post_oper mu_post_oper
Attribute 4	min_neg_post_oper mu_neg_post_oper max_neg_post_oper ratio_neg_post_oper num_neg_post_oper
Attribute 5	min_tail_actions max_tail_actions mu_tail_actions
Attribute 6	num_toggle_oper
Attribute 7	num_invariant_pred

Table 4.2. Costly features of the planner portfolio

pared with the choices made by two other selection heuristics: random and cheapest-first. Formally, random is defined as making a random selection from the list of unknown features (assuming a uniform distribution) and cheapest-first selects the unknown feature with the lowest test cost, with ties being broken via random selection.

TCSR was run on each planner dataset using the best regression algorithm as reported by Roberts and Howe (2006) (see Table 2.1). The implementations for the regression algorithms were used directly from the Weka machine learning toolkit (Witten & Frank 2005). During the training and testing phases, only successful instances are considered, because predicting the runtime of an unsuccessful run is not useful for a planner portfolio. A planner portfolio should not run a planner that is predicted to fail while there are planners predicted to succeed. Discarding the unsuccessful instances emulates the filtering of the instances through the success models.

The performance of TCSR on each dataset is captured by two metrics: the average error and the average cumulative acquisition cost. Each data point represents an average over all of the successful instances in the dataset. The data points on these graphs and for all experiments in this thesis were averaged over five trials of 10-fold cross validation. The error bars represent one standard deviation on either side of each point.

Many of the problems that were successfully solved were also the problems with low acquisition costs, which led to many of the “costly” features actually being nearly free. Applying TCSR in this environment made it difficult to evaluate the cost savings. To provide another perspective on TCSR, the true costs were replaced with random costs in the range of [0,100] and TCSR was re-run on each planner dataset. The random cost assignment was evaluated using the same metrics as the true cost assignment.

Learning curves were also generated for the multi-model operation mode, but a different metric was used to evaluate the results. In multi-model mode, the predicted runtime is less significant than correctly identifying the fastest planner. A simple portfolio strategy obtains runtime predictions and runs the planner with the fast predicted runtime. Even if the predicted fastest planner is

not correct, it is not necessarily a problem. For example, if the true runtime for planner *A* is 80 seconds and the true runtime for planner *B* is 85 seconds, then mistakenly identifying planner *B* as the fastest is acceptable. However, if the actual runtime of planner *B* is 300 seconds, then a significant amount of time can be wasted. Therefore, multiple-model mode is evaluated by the difference between the actual runtime of the true fastest planner and the actual runtime of the predicted fastest planner.

4.3 Single-model Evaluation

This section evaluates the performance of the single-model operation mode of TCSR using a true cost assignment and a random cost assignment. Since the true costs were close to zero for many problems, the average error and cumulative acquisition cost had to be displayed separately to be readable. For the random cost assignment, the graphs were merged to illustrate the relationship between cost and error reduction.

4.3.1 Evaluation with True Test Costs

Figure 4.1 depicts the performance of TCSR on the subset of all planners for which TCSR performed well. The performance graphs show that given a dataset that satisfies the assumption (acquiring the true value of a feature results in a better prediction) of TCSR, then it can perform well. TCSR performed consistently better than random and cheapest-first on this set of planners.

TCSR outperforms random and cheapest-first across the entire range of attribute acquisitions for *simplanner* and *lpg-td-1.0*. As the first attributes are acquired, the average error for TCSR decreases much more quickly than random or cheapest-first. TCSR not only performs better in terms of error, it achieves these results with only a slightly larger incurred cost than that of the cheapest-first heuristic. These datasets are examples of the type of dataset that TCSR performs well on: the cheapest features do not necessarily have the best correlation with the class variable

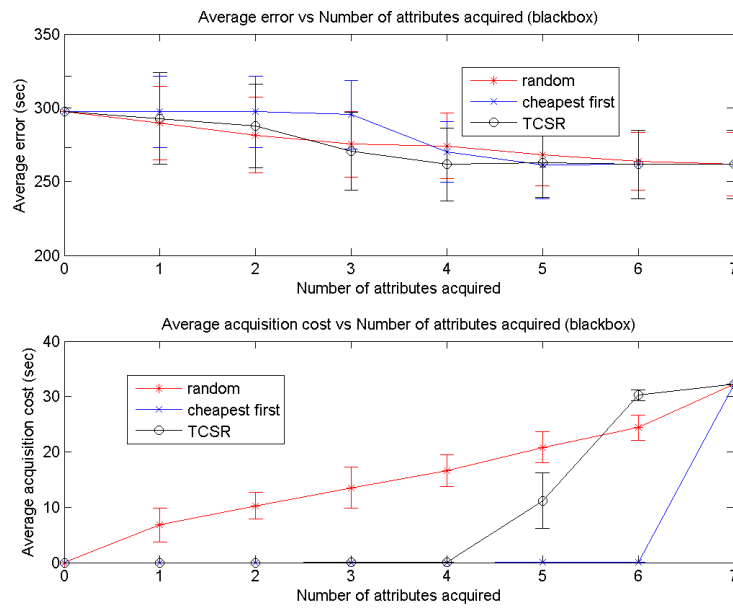
and the acquisition of a true feature value generally improves the quality of the predictions.

As one of the few planners to solve any problems with significant feature costs, *lpg-td-1.0* (Figure 4.1b) serves as an excellent example of the capability of TCSR to manage both tasks of reducing error and cumulative acquisition costs. Cheapest-first is able to achieve the same performance as TCSR, but only after acquiring several irrelevant features. Random incurs a much higher cost and must acquire nearly all of the features to achieve a performance value comparable to that of TCSR.

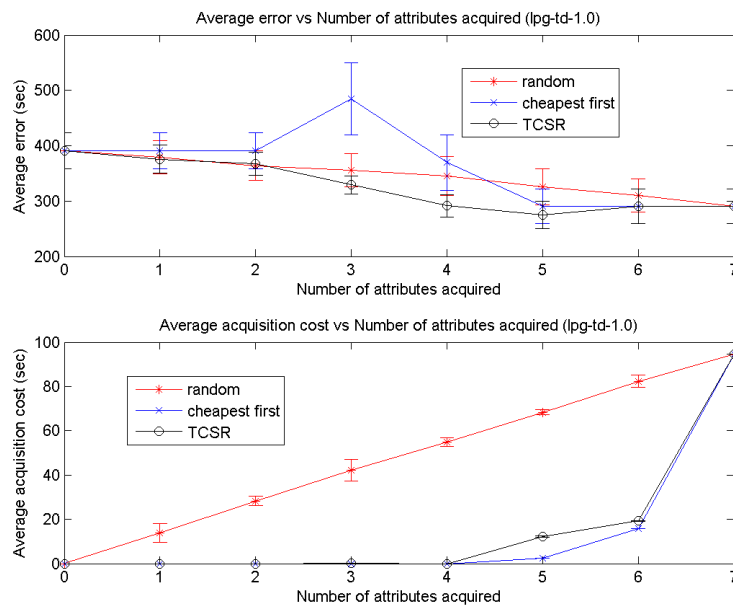
The graph for *optop* demonstrates the claim that TCSR correctly identifies which attribute is likely to produce the greatest change in prediction. Notice that the cumulative acquisition cost never deviates from zero. This means that the set of problems that *optop* solved only included features that were very cheap to compute. In this special case, cost is not a factor and TCSR acquires attributes based only on their expected change in prediction. It selects attributes that converge on the minimum error much sooner than either of the other strategies.

Other planners, such as *hsp-2.0-b-h1plus*, *r-1_1*, *metric_ff*, and *mips-3*, demonstrate characteristics of a dataset that cause TCSR to perform poorly (see Figure 4.2). TCSR behaves unpredictably when the regression model actually makes *worse* predictions with the true values than with the estimated values. This is a property of the regression model, not TCSR itself. Incorrectly estimating the probability distribution yields poor utility estimates that ultimately can cause TCSR to make poor acquisition choices.

The performance graphs for *mips-3* depict a situation in which all of the costly features seem to be irrelevant to the runtime prediction and acquiring their true value can actually cause the regression model to make *poorer* predictions. Sometimes TCSR can recover from acquiring a bad feature, as it does for *r-1_1*, but TCSR is often misled when several misleading features are present. The root of this problem stems from the fundamental assumption that acquiring true values of features will not hurt the prediction. Obviously this assumption is violated by *hsp-2.0-b-h1plus*, *r-1_1*, *metric_ff*, and *mips-3*, because the average initial error, with all costly features imputed, is

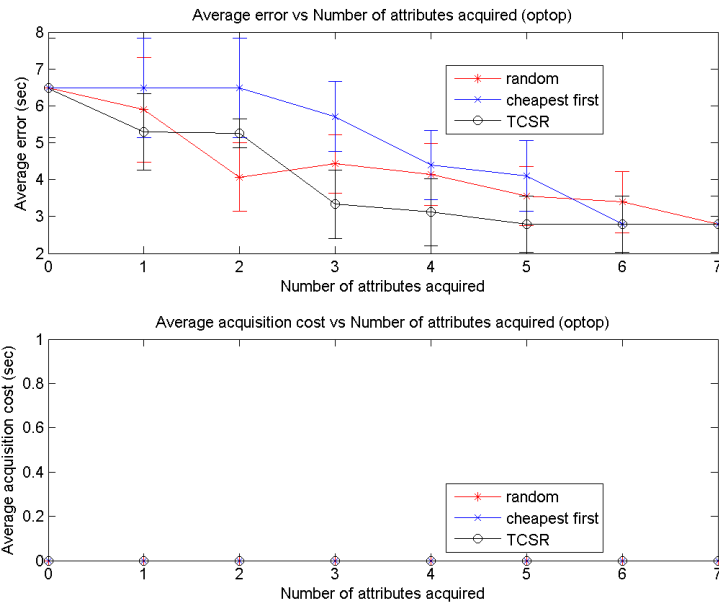


(a) blackbox-4.2

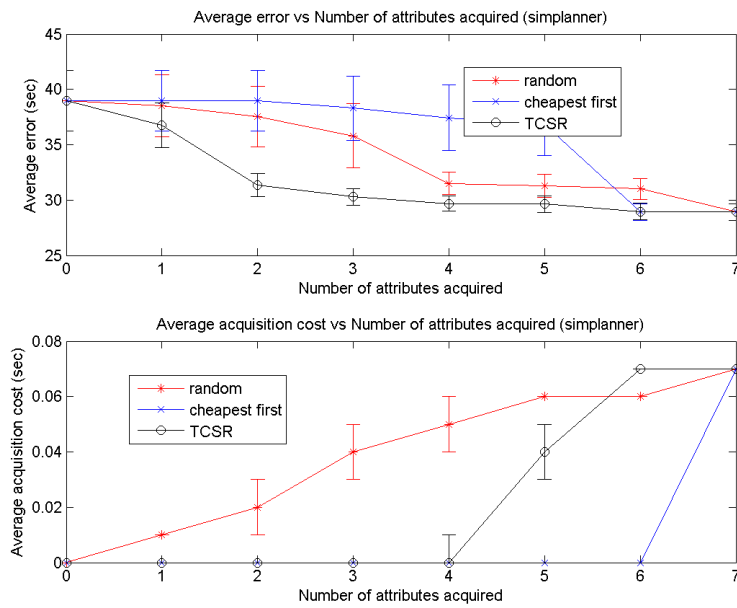


(b) lpg-td-1.0

FIG. 4.1. The performance of TCSR on datasets on planners that exhibit characteristics that are consistent with TCSR assumptions (using the true cost assignment).



(c) optop



(d) simplanner

Figure 4.1 (continued)

less than that with all feature values acquired.

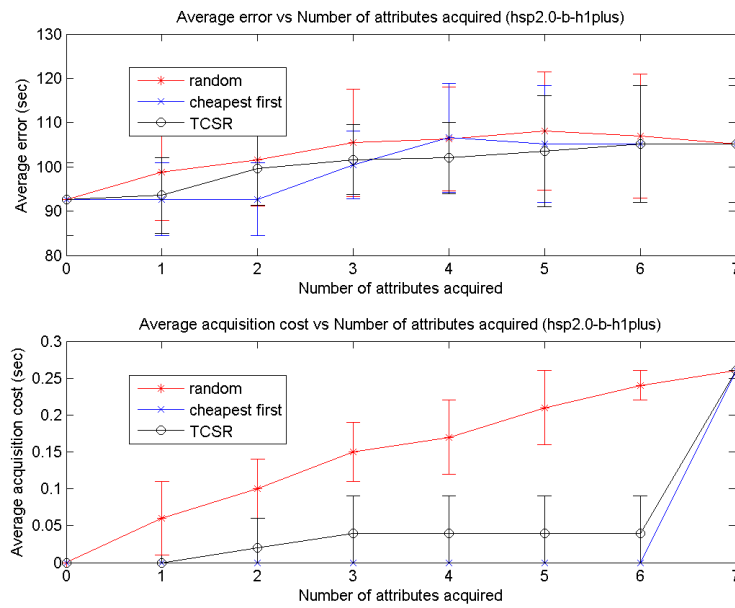
Examining each of the performance graphs shows that nine of the fourteen planner datasets comply with the general gain assumption; that is, the average error with all costly features imputed is greater than the average error with all costly features acquired. Of the nine datasets, TCSR performs *better* than random and cheapest-first on six (67%) of them (better is loosely defined in this situation to indicate that TCSR achieves a better average error per unit cost). The three planners on which TCSR does not exhibit superior performance still have individual features that cause the average error to increase, even though the average error will decrease when all features have been acquired. This indicates that although the general gain assumption is a reasonable guideline for predicting TCSR performance, features that cause average error to increase sharply before decreasing may still be present.

4.3.2 Evaluation with Random Test Costs

Applying a random cost assignment in place of the true cost structure provides a more complete understanding of how well TCSR minimizes both error and cumulative acquisition costs. The results once again show that TCSR performs well as long as the data and regression model do not violate its underlying assumptions.

For the planners in Figure 4.3, TCSR maintains a better average error than both cheapest-first and random over the majority of the features acquired (and sometimes over the entire range). In these situations, TCSR satisfies both learning objectives by achieving a better error rate on a smaller budget.

The planners that exhibited poor behavior with the true costs (i.e., increasing error as more features are acquired) continue to exhibit this behavior with the random costs (see Figure 4.4). This is not surprising because the problem is correlated with the regression model and the algorithm from which it was induced, not the cost values or TCSR.



(a) hsp-2.0-b-h1plus

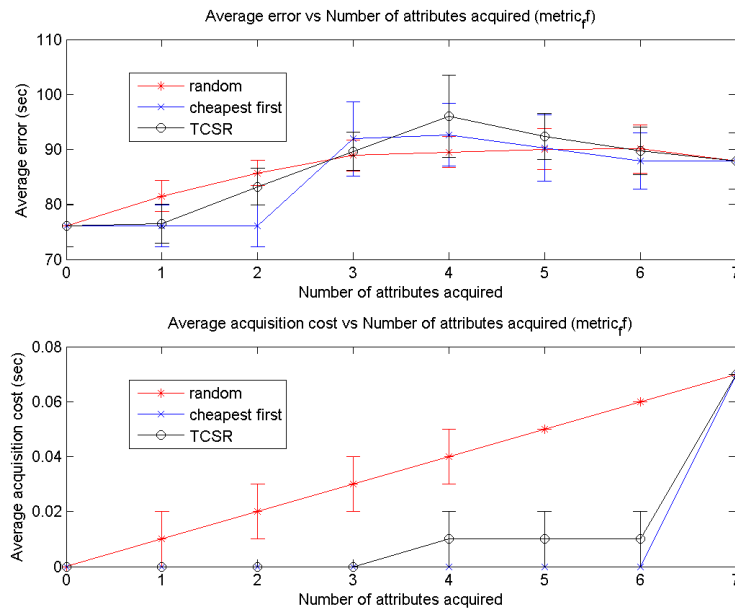
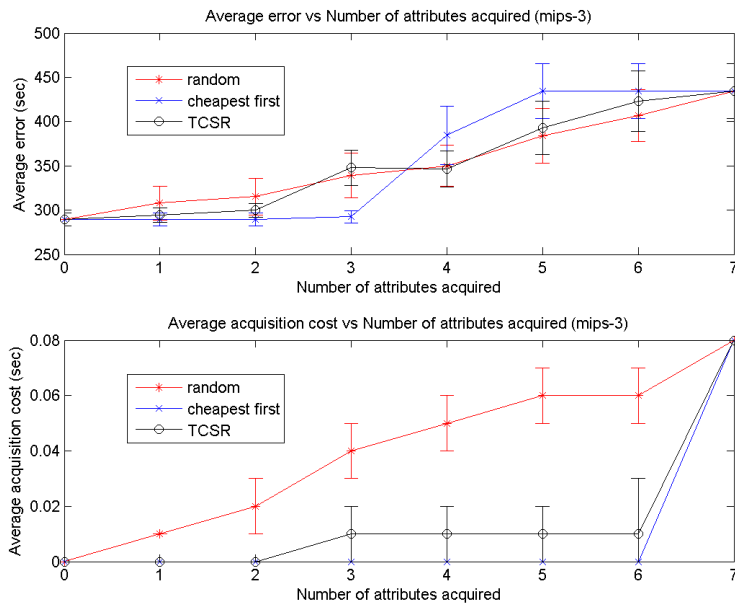
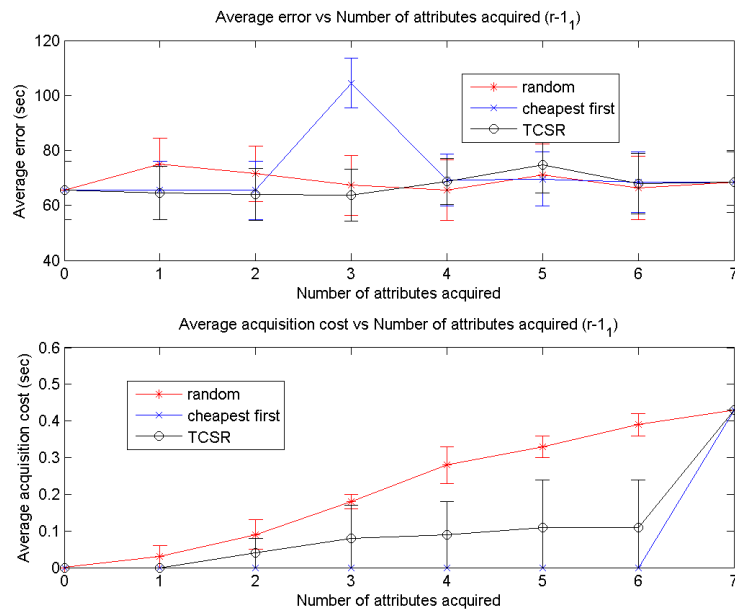
(b) metric_{ff}

FIG. 4.2. The performance of TCSR on planners that exhibit characteristics that violate the assumptions of TCSR (using the true cost assignment).

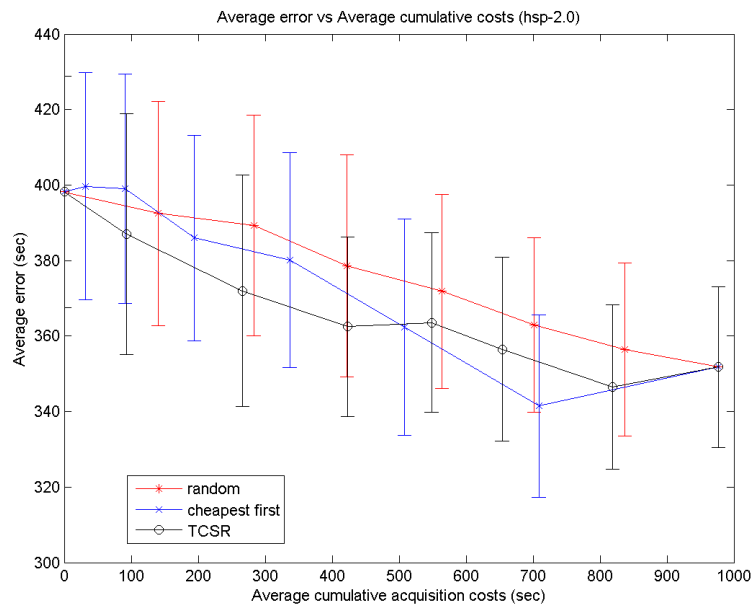


(c) mips-3

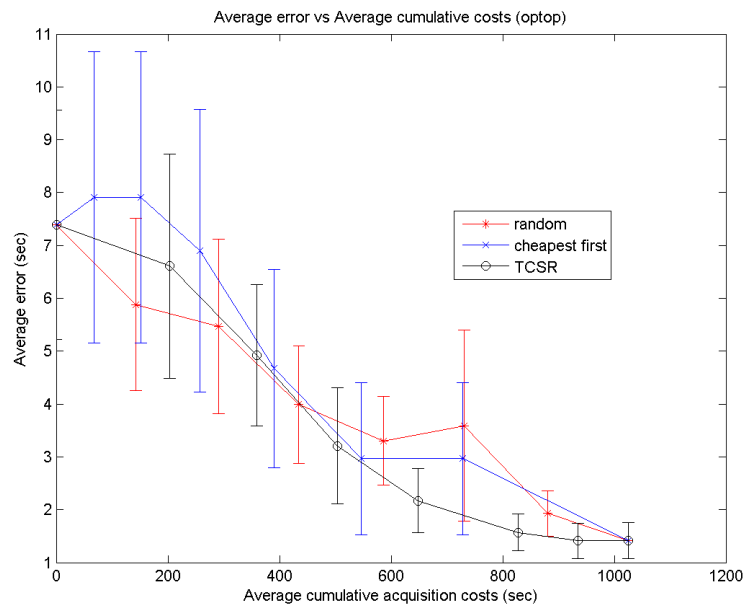


(d) r-1_1

Figure 4.2 (continued)

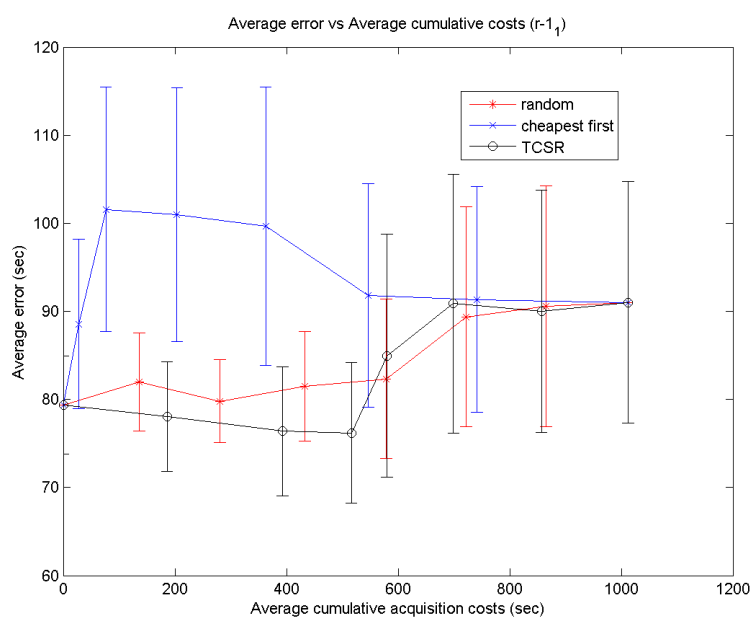


(a) hsp-2.0



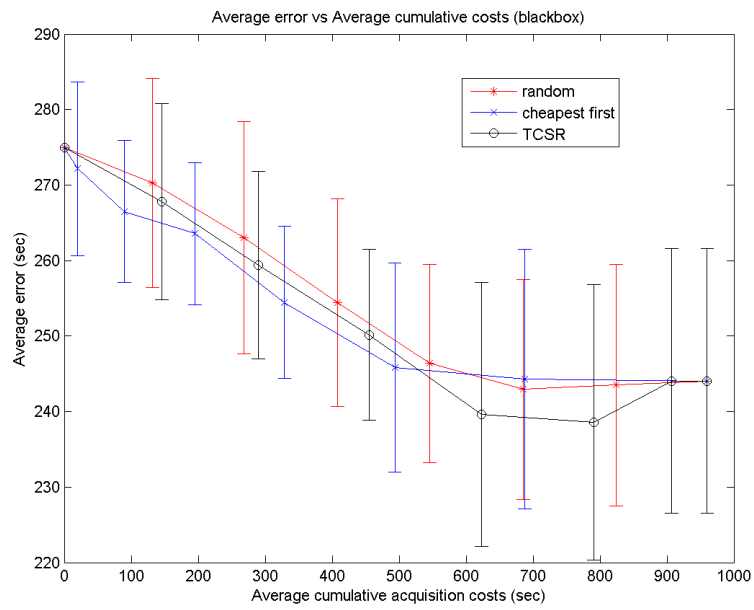
(b) optop

FIG. 4.3. Performance evaluation of TCSR on planner dataset that exhibit characteristics that are consistent with the TCSR assumptions (using random cost assignment).

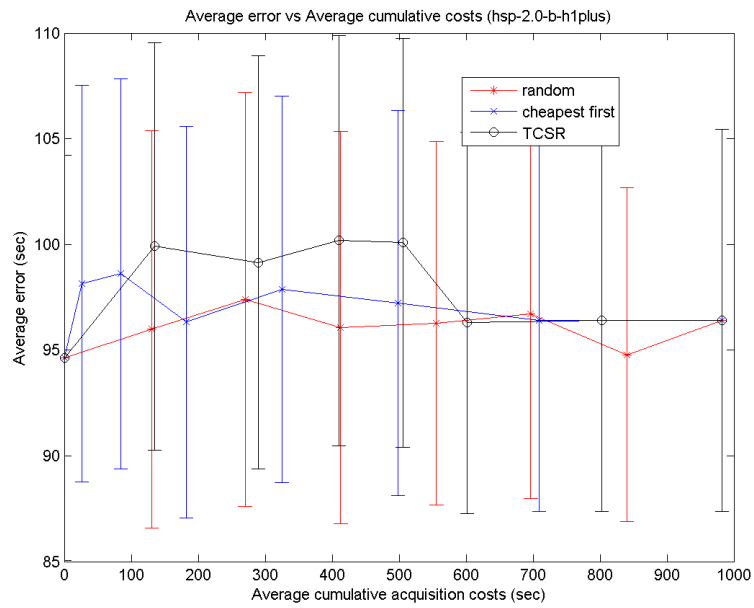


(c) r-1_1

Figure 4.3 (continued)

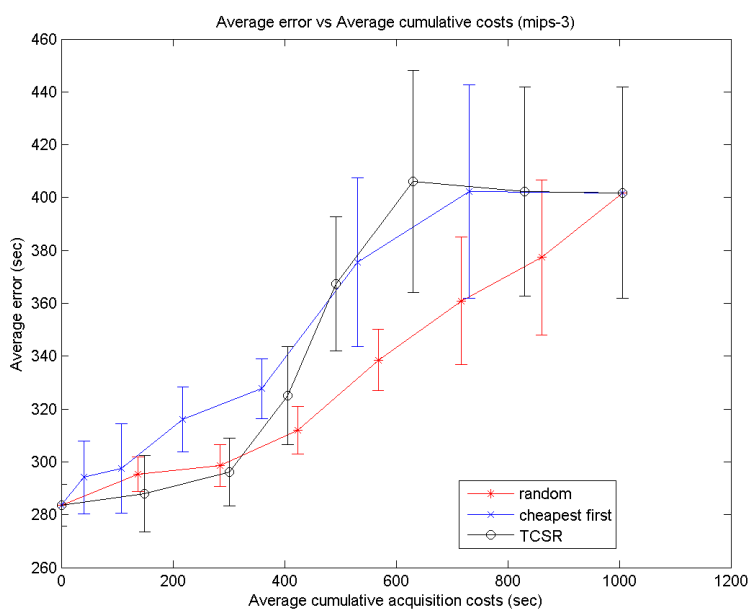


(a) blackbox-4.2



(b) hsp-2.0-b-h1plus

FIG. 4.4. Performance evaluation of TCSR on planner dataset that exhibit fail to uphold the assumption underlying the TCSR method (using random cost assignment).



(c) mips-3

Figure 4.4 (continued)

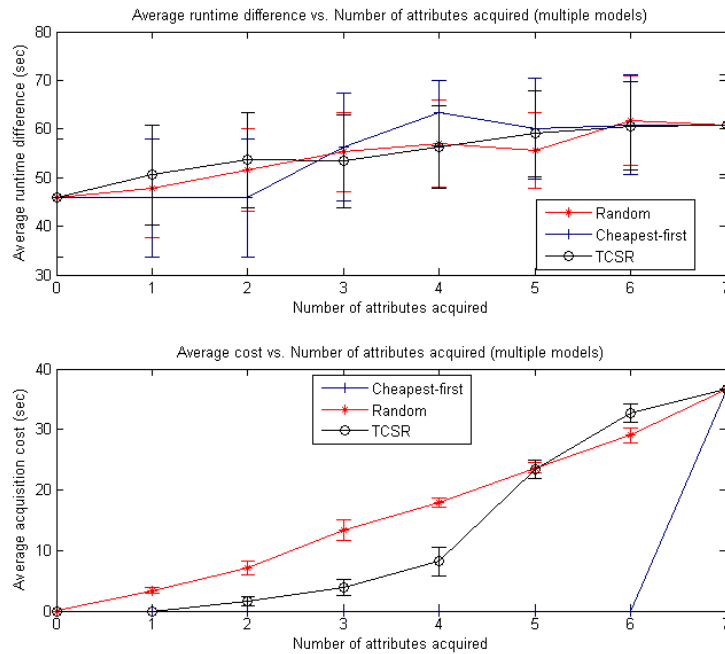


FIG. 4.5. The performance graph for TCSR operating in multiple-model mode (using the true cost assignment).

4.4 Multiple-model Evaluation

The evaluation of the multiple-model operation of TCSR focused on the subset of planners that TCSR proved to work well with in the single-model mode: *blackbox-4.3*, *hsp-2.0*, *lpg-td-1.0*, *optop-1.6.19*, *r-1.1*, and *simplanner*. Although all successful instances were still utilized during training, only the instances that were solved by two or more planners were tested and used to create the performance graphs. There are a total of 251 instances that fit this criterion.

Figure 4.5 depicts the performance of TCSR in multiple-model mode. All three selection methods perform poorly. The average difference between the true runtime of the fastest planner and the true runtime of predicted fastest planner continually increased as feature values were acquired.

Since the models utilized in this experiment were all models that performed reasonably well in single-model mode, it can be assumed that the problems that contributed to that performance were excluded from this performance metric (either only one planner solved them, or because the fastest planner was correctly predicted).

Although the experimental results were poor for the multiple-model operation, it cannot be concluded that the multiple-model operation does not work. The predictive gain assumption does not hold for a majority of the test instances for one or more of the models, causing the poor performance.

Chapter 5

DISCUSSION AND CONCLUSION

Empirical evaluation of TCSR has shown that it is superior to random and cheapest-first acquisition in the presence of certain data and regression model characteristics. On the other hand, data and models that deviate from these characteristics were shown to cause TCSR to behave unreliably, often causing performance to deteriorate. This section analyzes the aspects of a problem that affect the performance of TCSR and provides insight into determining when TCSR is applicable.

5.1 Advantages and Disadvantages of TCSR

The seemingly erratic behavior of TCSR is actually the result of the assumption that any change in prediction resulting from acquiring the true value of a feature will bring the prediction closer to the true value (referred to as the *predictive gain assumption*). This assumption is at the core of the TCSR approach. As expected, compared to random and cheapest-first, TCSR performs quite well when this assumption is upheld during the testing of all, or a large proportion of, the test instances. Random accrues a much larger cost than TCSR to achieve the same error reduction, while cheapest first minimizes the accrued cost, but tends to acquire many irrelevant features that do not contribute to the predictive accuracy.

When the predictive gain assumption is violated, TCSR behaves unpredictably. TCSR greedily acquires the feature that is expected to provide the greatest change in the predicted value. It is

unable to determine whether the change will actually bring the prediction closer to the true value or move it further away from the true value. This causes TCSR to fluctuate between acquiring features that improve the prediction and those that degrade the prediction instead of converging on the best accuracy.

There are two scenarios that can cause the gain assumption not to hold: (1) when the “true” value of an unknown feature is not actually the value of the feature (noisy domains), and (2) when the learned model is inaccurate, which means that the imputed value actually can cause the prediction to be better than the true feature value. Noise was not a problem in the planner performance domain, but it would be in many others where noise-introducing agents, such as noisy sensors, are present. An inaccurate regression model is the result of a learning algorithm that learns the incorrect function for a set of instances. As a result, the model can sometimes make better predictions with estimated values of unknown features. The inaccuracies of the regression model are then propagated to TCSR, degrading the performance.

Short of selecting a different underlying regression method, several other steps can be taken to limit the loss in performance that results from an inaccurate regression model. When working with a high-dimensional dataset, as with the planner performance data, more irrelevant features are likely to be present. Using a feature selection algorithm to pre-process the data will prevent the regression algorithm from ever incorporating the irrelevant features, so it is less likely to incorrectly weight them. The effect of dimensionality on TCSR performance is left as future work.

5.2 Sources of “Bad” Gain

The performance of TCSR has been shown to be poor when the gain assumption does not hold. “Bad” gain can be a result of several factors: noise, high target feature variance over the range of values of an unknown feature, and a poor regression model. Each of these sources can exist independently, but when more than one is present, the effects can be more noticeable.

If the true feature values contain noise then it is very likely that an imputed value could produce a better estimate than the true value. That is because the noise can cause the true value to be incorrect and mislead the regression model more than the estimated value would. Estimating the noise in the dataset would provide more insight into the expected TCSR performance.

An irrelevant feature with high target feature variance over the unknown feature range can also mislead TCSR during the gain calculation. In this situation, the large fluctuations in the target value over the unknown feature values indicate a large gain, even though the feature is actually irrelevant to the target value. When TCSR acquires this feature, instead of achieving a better prediction and reducing error as the gain value would suggest, the error may not change, or may even increase.

The final source of “bad” gain is the regression model itself. If the learned regression model is not exactly correct, then a value other than the true feature value may result in a better prediction. Consider a simple example where the training data is a random sampling from the function, $y = x^2 + 4$. If the regression algorithm learns the function, $y = x + 8$, then regression model will produce better predictions with values other than the true values. For example, given the true value of x is 4, the regression model will predict $y = 12$, an error of 8. Using the value of 12 for x would have given a better prediction of $y = 20$, an error of 0. Obviously, in this situation, an estimated value of x results in a better prediction because of the incorrect regression model.

5.3 Future Work

The experimental results revealed that the performance of TCSR is highly sensitive to various conditions. The predictive gain assumption has a devastating effect on the performance of TCSR when it does not hold. Also, although TCSR generalizes to allow groups of features that can be acquired at a single fixed price, the computational demand of TCSR grows exponentially in the size of the largest individual group. Finally, it appears that for some planners, there are features

that improve the prediction only after the value of some other features are known. Acquiring these features simultaneously, as a batch, would improve performance more quickly. Each of these issues will be examined as future work.

The presence of irrelevant features can negatively affect the performance of a regression model; this in turn causes TCSR to make poor utility estimates. Reducing the dimensionality of the dataset, using a feature selection or feature extraction technique, prior to applying TCSR would eliminate many of the irrelevant features.

An improved stopping criterion, for ceasing further acquisitions, would improve TCSR performance by determining when the error begins increasing. For several planners, TCSR initially chose features that decreased the error until a local minimum was reached. After this minimum, further acquisitions resulted in increases in error. A stopping criterion that determines when it has achieved this local minimum would greatly reduce the incurred cost on acquiring bad features.

Computing the expected utility of a group of features f_i requires the predicted value for each possible combination of the features in that group ($|f|^k$ combinations, assuming k possible values for each feature). For many regression algorithms, especially lazy learners, this is not practical. A sampling method, like that utilized by SEU, that randomly selects a representative set of features from the group would reduce complexity, but might have unpredictable effects on the acquisition decisions. Random sampling as well as more sophisticated sampling methods were effective for SEU and therefore should be investigated in the future work.

For some learning problems, the contribution of certain features is highly correlated with the values of other features; when acquired individually, the features may be irrelevant or worse. To remedy this scenario, the future work should also focus on acquiring features in batches. Applying the SBT algorithm (see Section 2.3.2) would not only allow for batches, but it would also incorporate the potential for delays associated with the features.

The true value of feature costs are not always known. In this thesis the costs were assumed to be known, but in actuality, the cost of acquiring a feature in the planner portfolio is not known

until it has been acquired. The future work will account for problems where the true costs are not known and instead there may be a distribution of acquisition costs. The cost estimation must be factored into the feature-acquisition decision making process as well.

5.4 Conclusion

This thesis examined the problem of test-time feature-value acquisition in the context of the numeric prediction problem (regression). The approach examined in this thesis is a greedy feature acquisition algorithm that acquires the values of unknown features based on their estimated expected utility. Within the domain of planner runtime prediction, TCSR successfully minimizes the prediction error per unit cost faster than both random selection and cheapest-first.

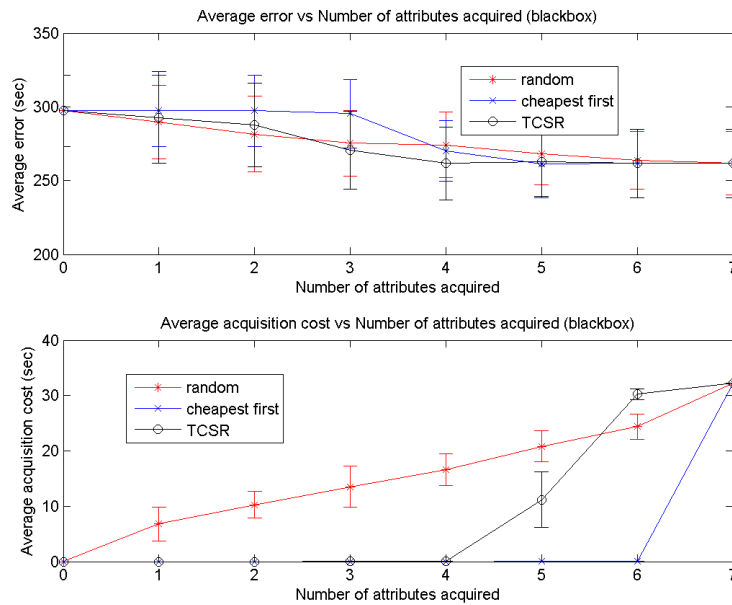
TCSR performance degrades rapidly when the acquisition of missing values causes predictions to worsen. TCSR acquires the feature that is expected to change the prediction the most. Since the acquisition is at test-time, whether the change is for the better or for the worse cannot be determined. The presence of many irrelevant features can increase the chances that this happens. Reducing the dimensionality of the data prior to applying TCSR is suggested as a possible remedy.

The results of this thesis support further investigation into the feature-value acquisition problem for regression, specifically the probabilistic approach utilized by TCSR. For cost-sensitive domains with many features, TCSR can reduce the costs incurred from obtaining feature values of expensive, but relevant, features; current regression techniques either guess these values or require them all to be acquired.

Appendix A

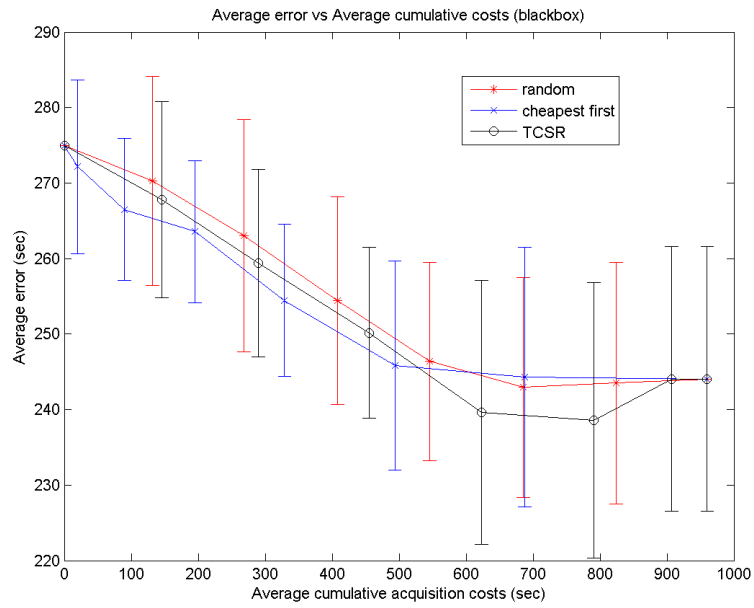
SINGLE-MODEL PERFORMANCE GRAPHS

This Appendix contains the performance graphs for each of the planners used in the experimental analysis using both the true and random cost assignment structures.

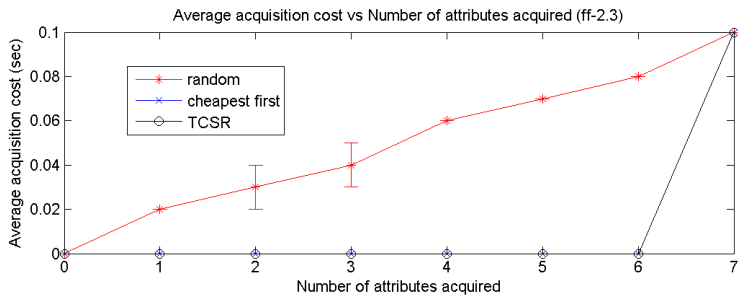
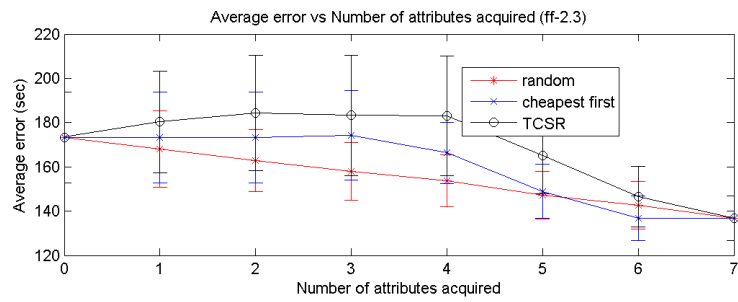


(a) blackbox-4.2 - true costs

FIG. A.1. The single-model performance graphs for each both true and random cost assignments by planner.

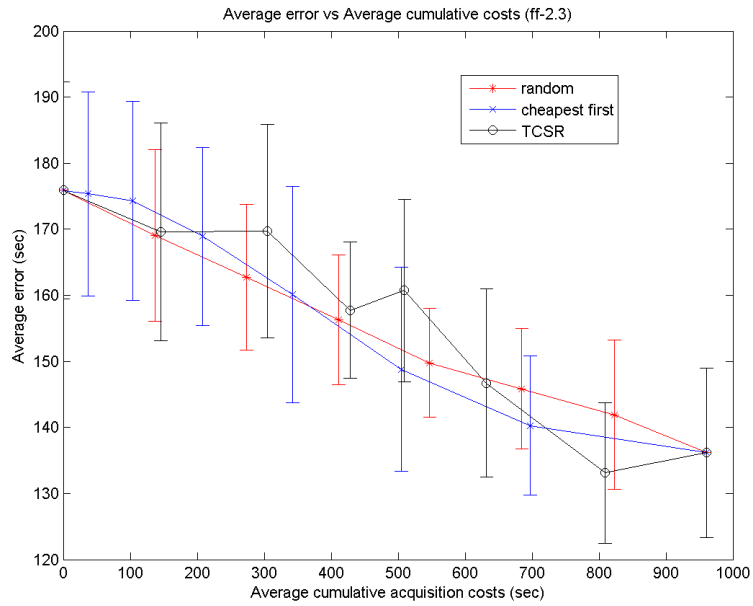


(a) blackbox-4.2 - random costs

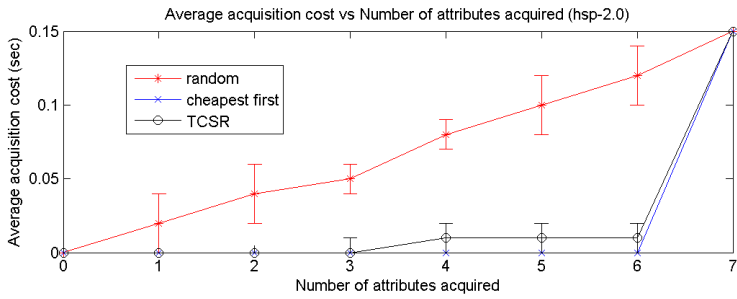
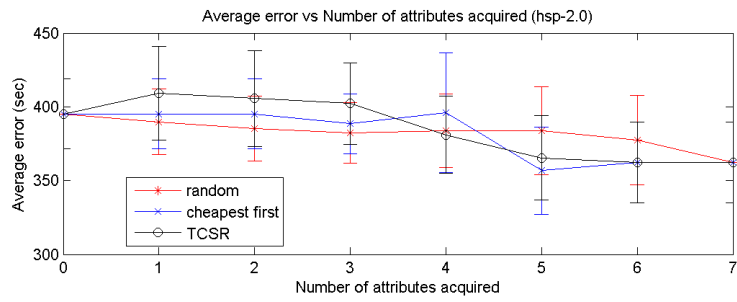


(b) ff-2.3 - true costs

Figure A.1 (continued)

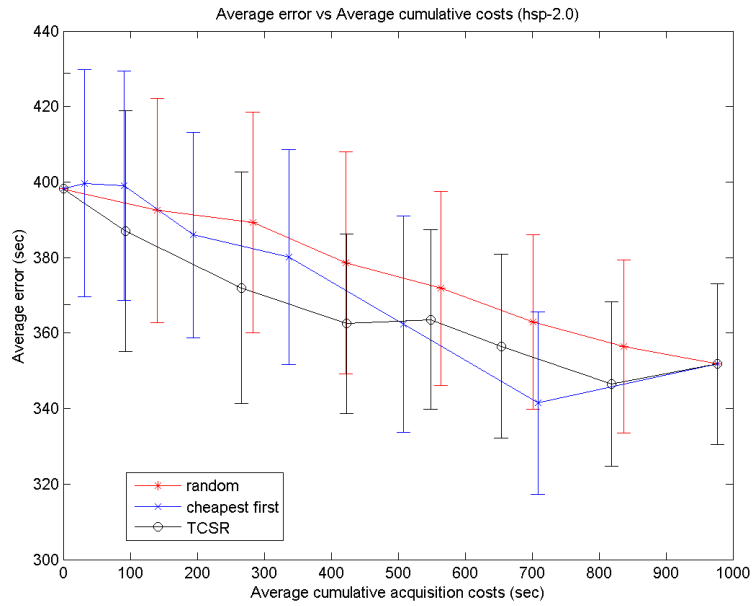


(b) ff-2.3 - random costs

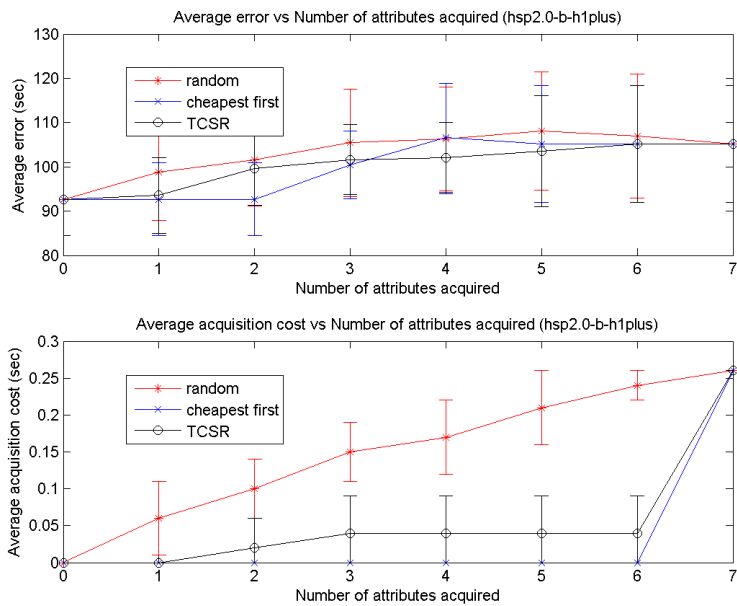


(c) hsp-2.0 - true costs

Figure A.1 (continued)

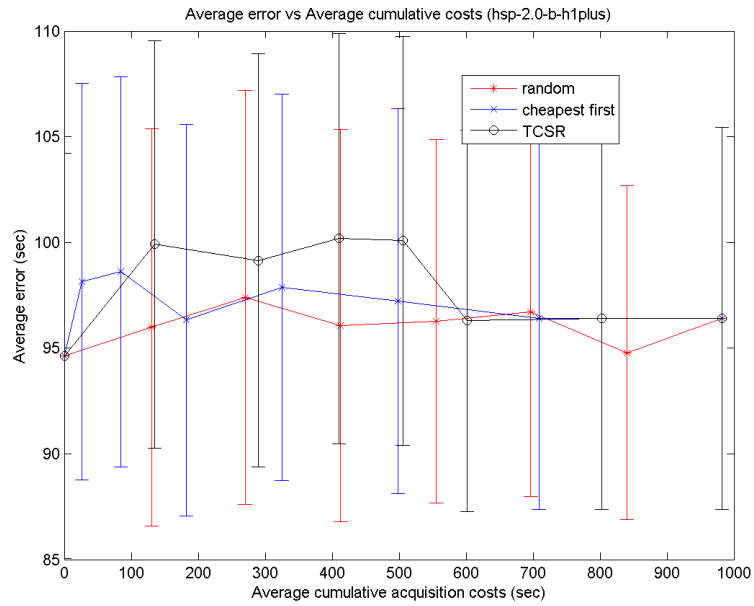


(c) hsp-2.0 - random costs

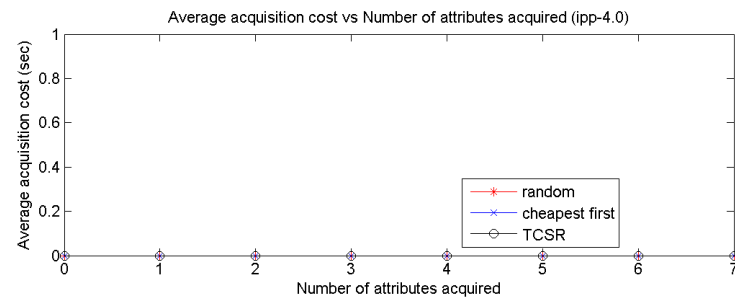
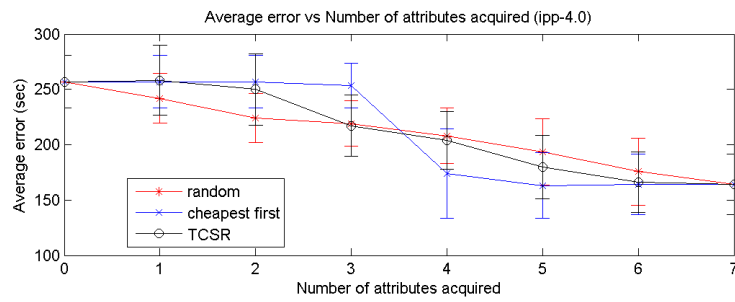


(d) hsp-2.0-b-h1plus - true costs

Figure A.1 (continued)

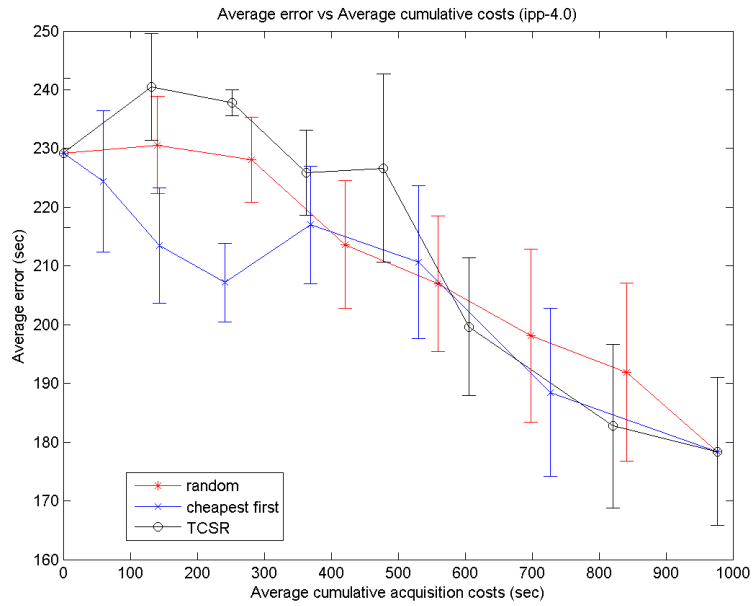


(d) hsp-2.0-b-h1plus - random costs

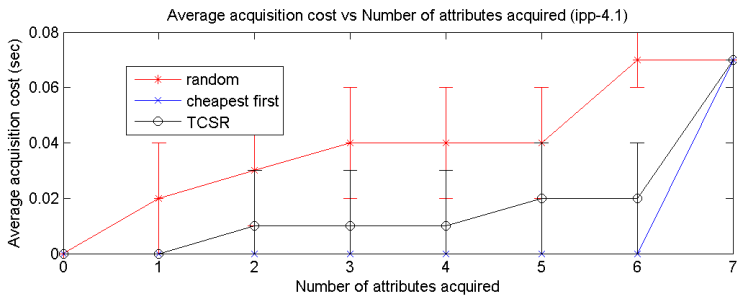
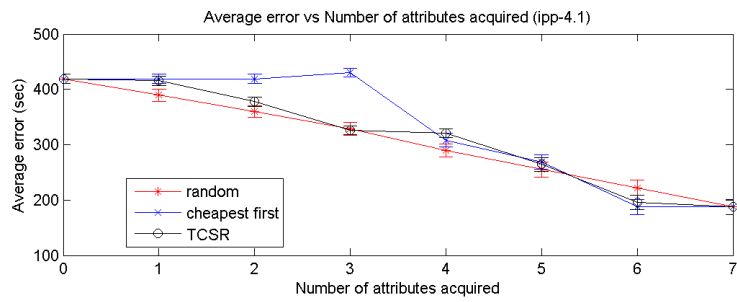


(e) ipp-4.0 - true costs

Figure A.1 (continued)

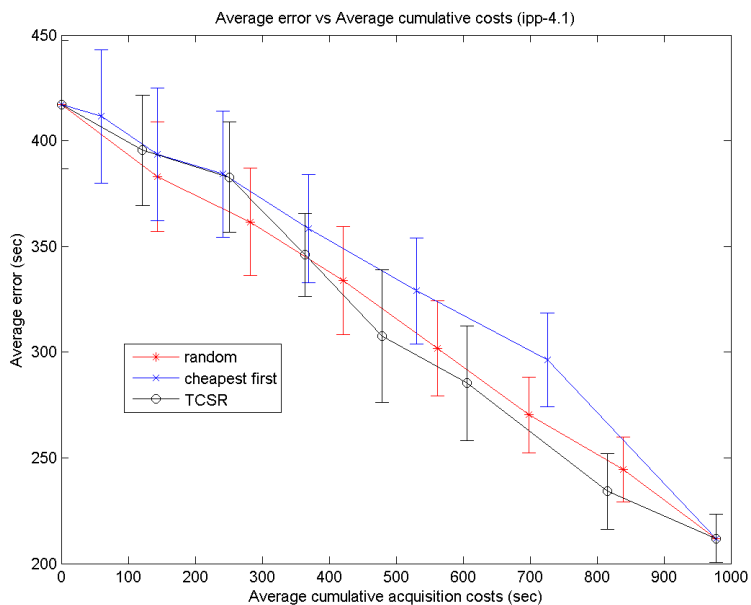


(e) ipp-4.0 - random costs

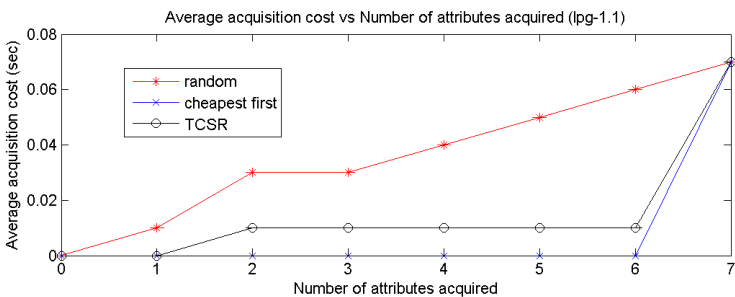
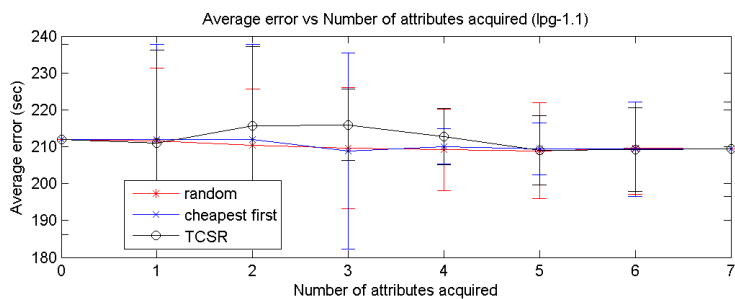


(f) ipp-4.1 - true costs

Figure A.1 (continued)

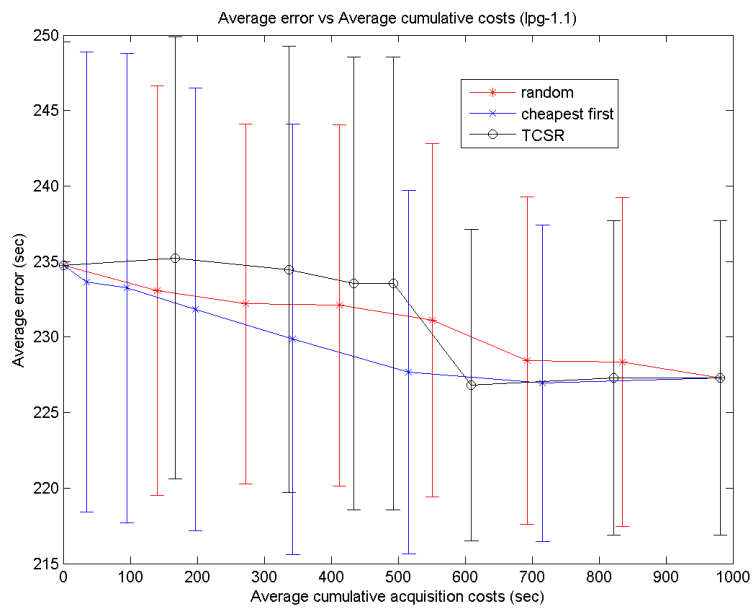


(f) ipp-4.1 - random costs

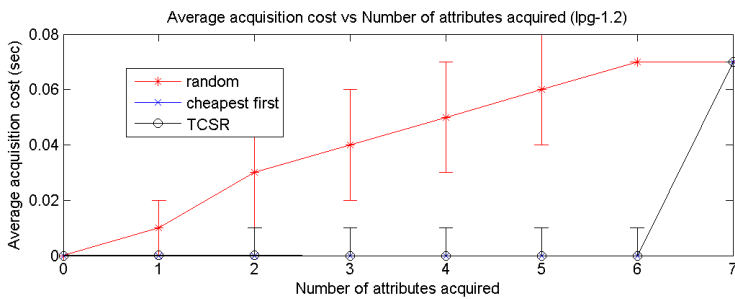
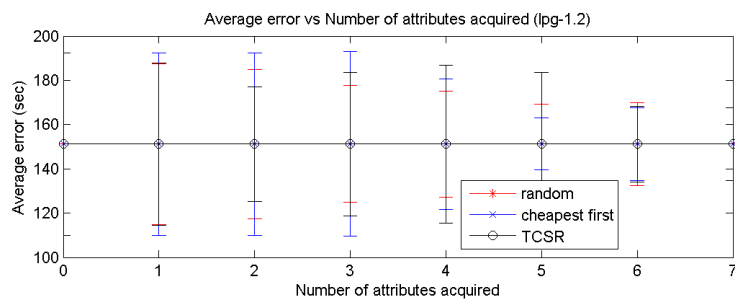


(g) lpg-1.1 - true costs

Figure A.1 (continued)

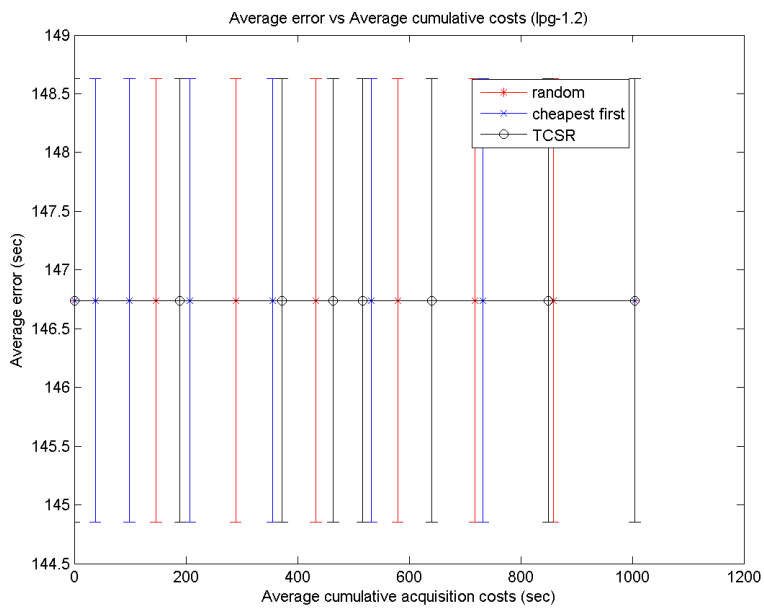


(g) lp-g-1.1 - random costs

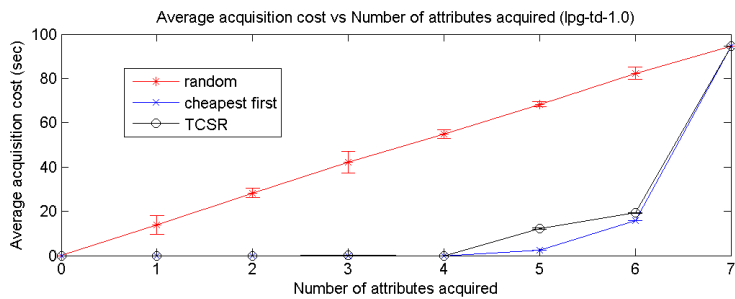
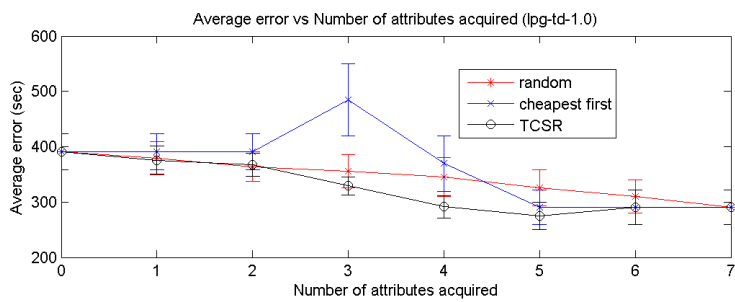


(h) lp-g-1.2 - true costs

Figure A.1 (continued)

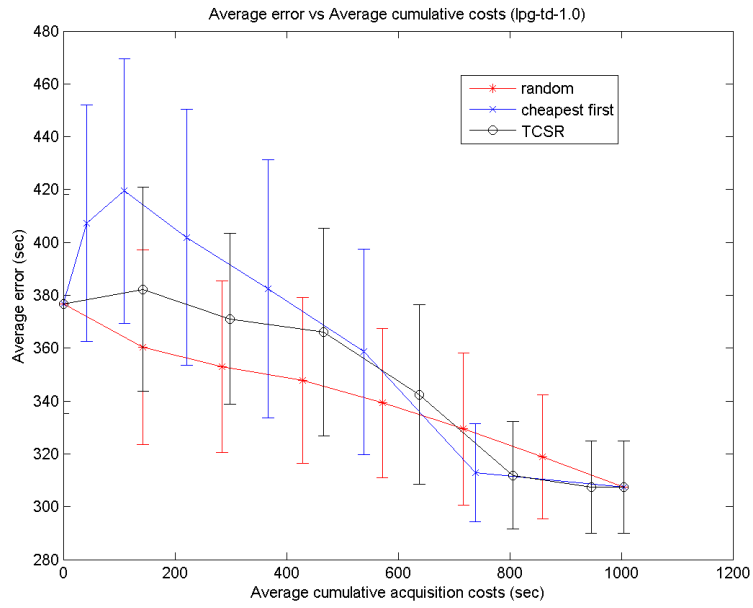


(h) lp-g-1.2 - random costs

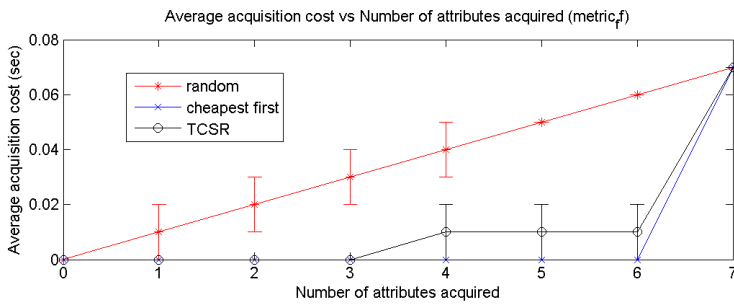
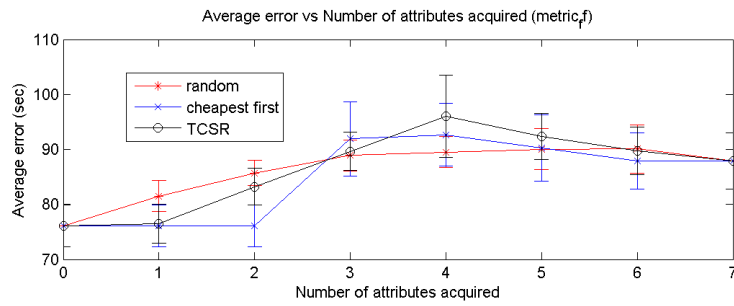


(i) lp-g-td-1.0 - true costs

Figure A.1 (continued)

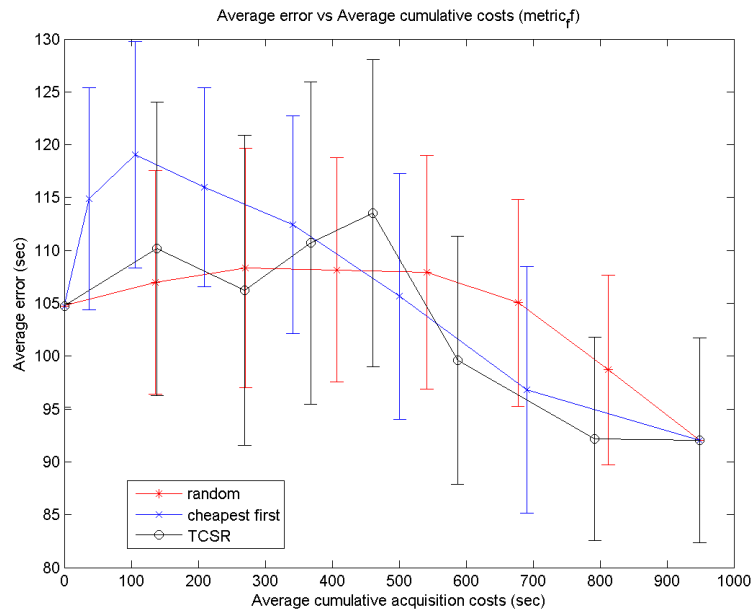


(i) lpg-td-1.0 - random costs

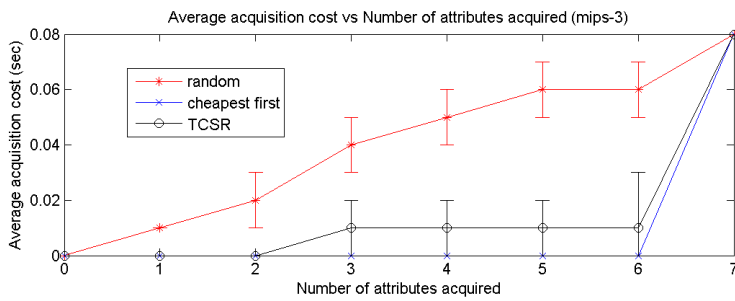
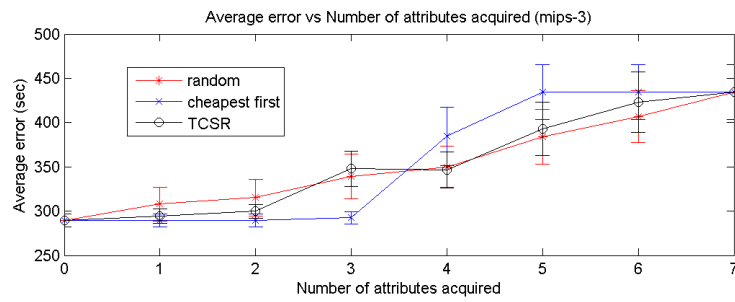


(j) metric_ff - true costs

Figure A.1 (continued)

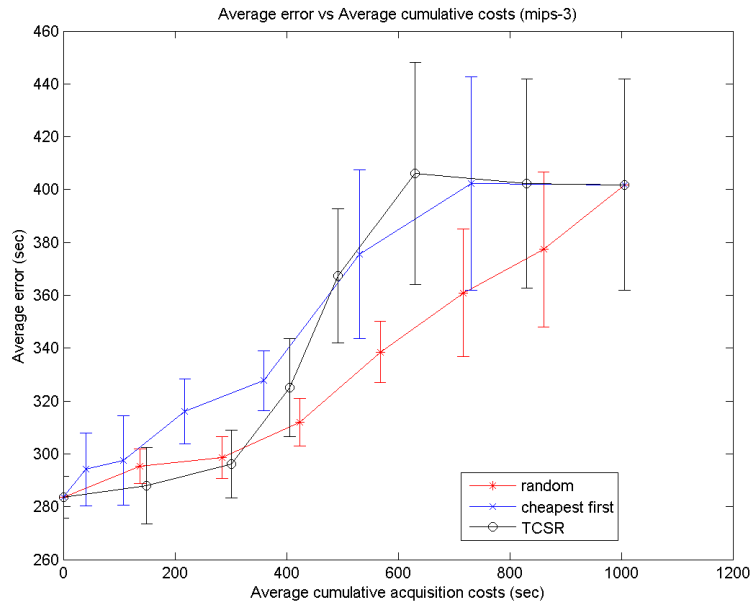


(j) metric_ff - random costs

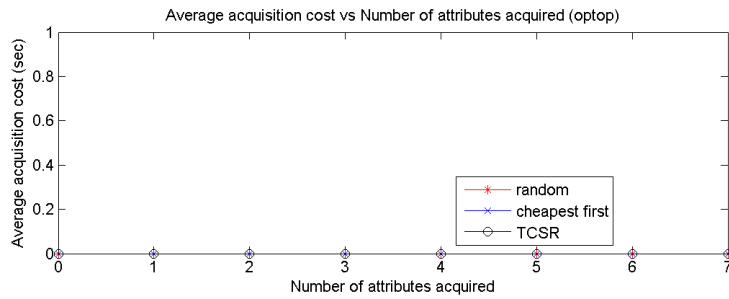
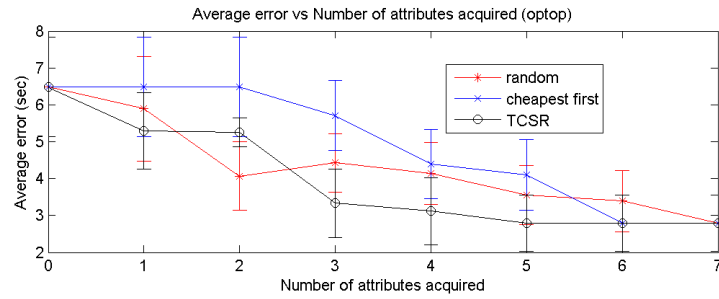


(k) mips-3 - true costs

Figure A.1 (continued)

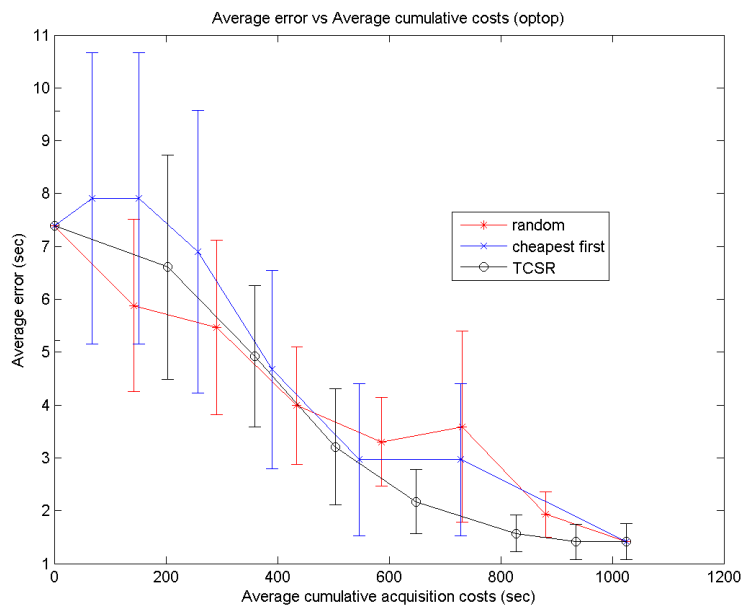


(k) mips-3 - random costs

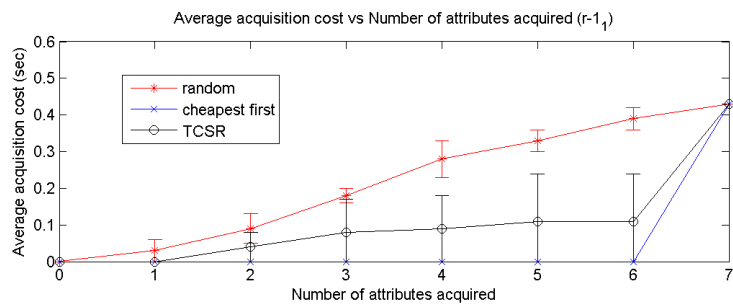
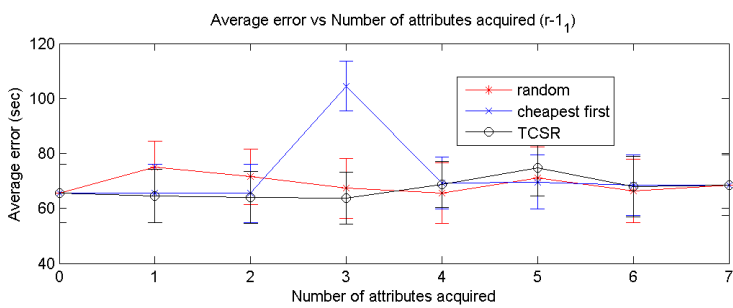


(l) optop - true costs

Figure A.1 (continued)

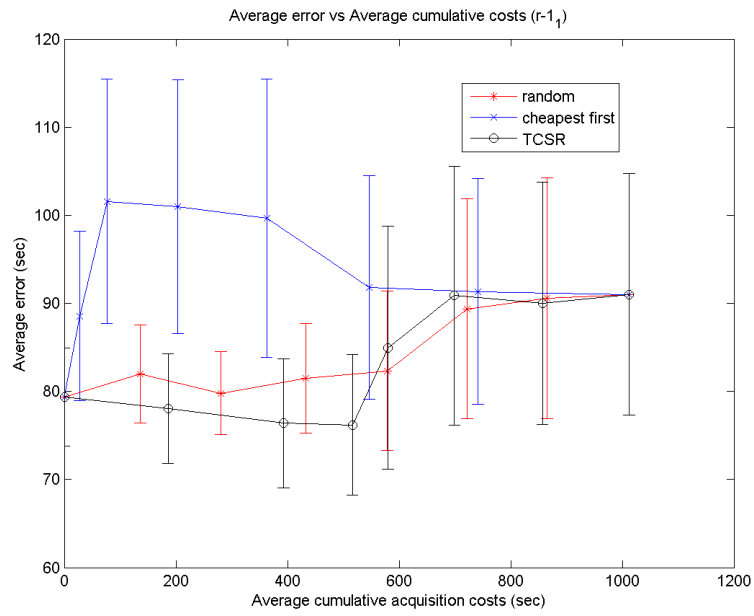


(l) optop - random costs

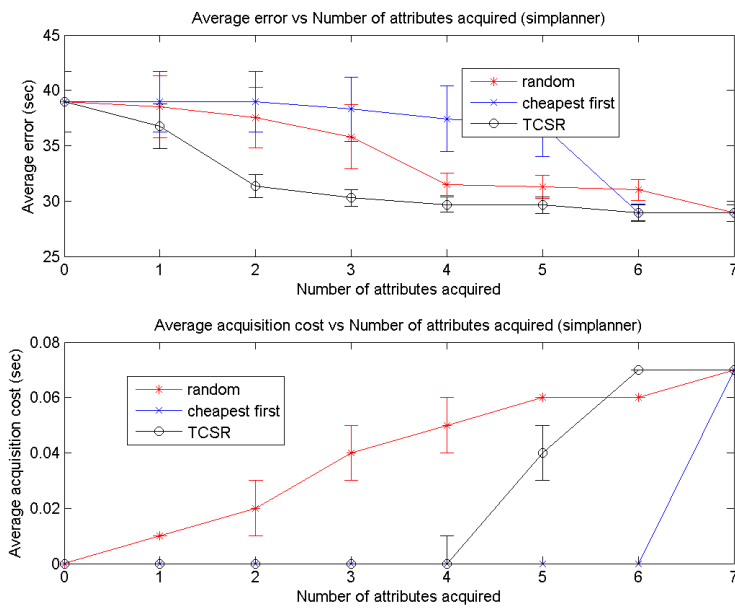


(m) r-1_1 - true costs

Figure A.1 (continued)

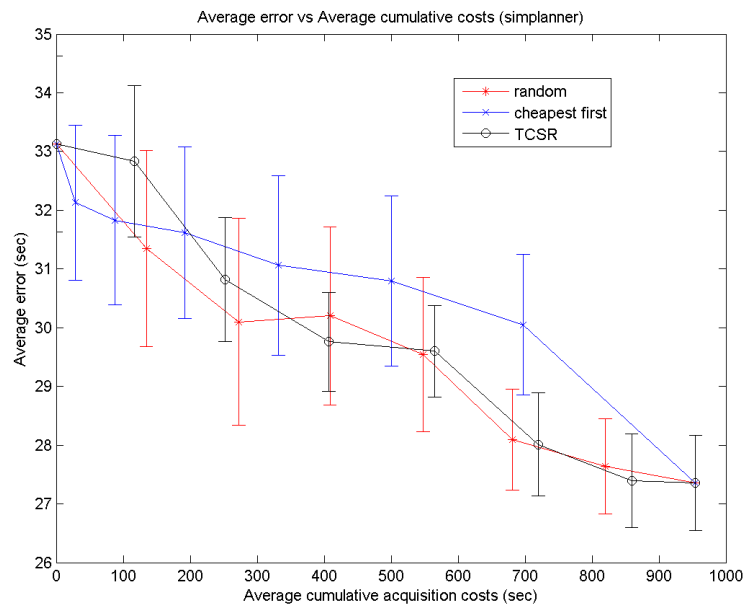


(m) r-1_1 - random costs



(n) simplanner - true costs

Figure A.1 (continued)



(n) simplanner - random costs

Figure A.1 (continued)

REFERENCES

- [1] Batista, G. E., and Monard, M. C. 2003. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence* 17(5):519–533.
- [2] Chai, X.; Deng, L.; Yang, Q.; and Ling, C. X. 2004. Test-cost sensitive naive Bayes classification. *Proceedings of the Fourth IEEE International Conference on Data Mining* 51–58.
- [3] Crone, S. F.; Lessmann, S.; and Stahlbock, R. 2005. Utility based data mining for time series analysis: cost-sensitive learning for neural network predictors. In *UBDM '05: Proceedings of the 1st International Workshop on Utility-Based Data Mining*, 59–68. New York, NY, USA: ACM.
- [4] Domingos, P. 1999. MetaCost: A general method for making classifiers cost-sensitive. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 155–164.
- [5] Elkan, C. 2001. The foundations of cost-sensitive learning. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* 973–978.
- [6] Fan, W.; Stolfo, S. J.; Zhang, J.; and Chan, P. K. 1999. AdaCost: Misclassification cost-sensitive boosting. *Proceedings of the Sixteenth International Conference on Machine Learning* 97–105.
- [7] John, G. H., and Langley, P. 1995. Estimating continuous distributions in Bayesian classifiers. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* 338–345.
- [8] Knoll, U.; Nakhaeizadeh, G.; and Tausend, B. 1994. Cost-sensitive pruning of decision trees. *Proceedings of the Eighth European Conference on Machine Learning* 94:383–386.

- [9] Marlon Núñez, M. 1991. The use of background knowledge in decision tree induction. *Machine Learning* 6(3):231–250.
- [10] Melville, P.; Saar-Tsechansky, M.; Provost, F.; and Mooney, R. 2004. Active feature-value acquisition for classifier induction. *Proceedings of the Fourth IEEE International Conference on Data Mining* 483–486.
- [11] Melville, P.; Provost, F.; Saar-Tsechansky, M.; and Mooney, R. 2005. Economical active feature-value acquisition through expected utility estimation. *Proceedings of the First International Workshop on Utility-based Data Mining* 10–16.
- [12] Roberts, M., and Howe, A. 2006. Directing a portfolio with learning. *Proceedings of the Workshop on Learning for Search at the Twenty-first National Conference on Artificial Intelligence*.
- [13] Roberts, M., and Howe, A. 2007a. Learned models of performance for many planners. In *Working Notes of the International Conference on Automated Planning and Scheduling*.
- [14] Roberts, M., and Howe, A. 2007b. Local search topology: implications for planner performance. *Working Notes of the International Conference on Automated Planning and Scheduling*.
- [15] Scott, D. W. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley-Interscience.
- [16] Sheng, V. S., and Ling, C. X. 2006. Feature value acquisition in testing: a sequential batch test algorithm. In *Proceedings of the Twenty Third International Conference on Machine learning*, 809–816. New York, NY, USA: ACM.
- [17] Silverman, B. 1986. *Density estimation for statistics and data analysis*. Chapman & Hall/CRC.

- [18] Ting, K. M. 1998. Inducing cost-sensitive Ttees via instance weighting. *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery* 23–26.
- [19] Torgo, L., and Ribeiro, R. 2007. Utility-based regression. *Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases* 4702:597–604.
- [20] Turney, P. 2000. Types of cost in inductive concept learning. In *Proceedings of the Workshop on Cost-Sensitive Learning at the Seventeenth Annual International Conference on Machine Learning*, 15–21.
- [21] Witten, I. H., and Frank, E. 2005. Data mining: Practical machine learning tools and techniques.
- [22] Yang, Q.; Ling, C.; Chai, X.; and Pan, R. 2006. Test-cost sensitive classification on data with missing values. *IEEE Transactions on Knowledge and Data Engineering* 18(5):626–638.
- [23] Zheng, Z., and Padmanabhan, B. 2002. On active learning for data acquisition. *Proceedings of the IEEE International Conference on Data Mining* 562–569.

