Granul. Comput. (2016) 1:199–211
DOI 10.1007/s41066-016-0017-2

ORIGINAL PAPER

Semi-greedy heuristics for feature selection with test cost constraints

Fan Min[1] · **Juan Xu**[1]

Received: 9 September 2015 / Accepted: 12 February 2016 / Published online: 29 February 2016

Abstract In real-world applications, the test cost of data collection should not exceed a given budget. The problem of selecting an informative feature subset under this budget is referred to as feature selection with test cost constraints. Greedy heuristics are a natural and efficient method for this kind of combinatorial optimization problem. However, the recursive selection of locally optimal choices means that the global optimum is often missed. In this paper, we present a three-step semi-greedy heuristic method that directly forms a population of candidate solutions to obtain better results. In the first step, we design the heuristic function. The second step involves the random selection of a feature from the current best k features at each iteration. This is the major difference from conventional greedy heuristics. In the third step, we obtain p candidate solutions and select the best one. Through a series of experiments on four datasets, we compare our algorithm with a classic greedy heuristic approach and an information gain-based λ-weighted greedy heuristic method. The results show that the new approach is more likely to obtain optimal solutions.

Keywords Feature selection · Granular computing · Semi-greedy · Test cost constraint

1 Introduction

Granular computing (Yao 2000; Lin 1998; Yao et al. 2013) is an emerging conceptual and computing paradigm for the processing of information (Pedrycz 2001; Bargiela and Pedrycz 2012). It is concerned with the construction and processing of data at the level of information granules (Pedrycz 2001, 2013). The term granular computing also covers any theories, methodologies, techniques, and tools that make use of information granules to solve complex problems (Yao et al. 2013). A wide range of applications have benefited from granular computing, such as image-processing (Wang and Zhang 2009), multimedia data semantic modeling (Al-Khatib et al. 1999), multi-scale decision making (Wu and Leung 2011; Gu and Wu 2013), fuzzy decision tree induction (Wang et al. 2008), document ranking (Qin et al. 2012) and recommendation systems (He et al. 2013).

An important task of granular computing is feature selection, whereby the resulting feature subset becomes the focus of our attention (Kohavi and John 1997). In this way, patterns are made more observable, and classifiers can be built more efficiently. Many feature selection problems have been identified in the rough set community (Pawlak 1982a), where the term "attribute reduction" is more often used. Such problems deal with information systems (Skowron and Rauszer 1992), decision systems (Słowiński 1992), coverings (Zhu and Wang 2003; Chen et al. 2007), fuzzy systems (Hu et al. 2007; Zhai et al. 2013), three-way decision data (Yao and Zhao 2008), and high-dimensional data (Janusz and Ślęzak 2014). There are a number of different definitions of a reduct based on, e.g., the positive region (Modrzejewski 1993) or information entropy (Ślęzak 2002). Numerous feature selection algorithms have also been developed, including discernibility-based methods (Zhang et al. 2005), greedy methods (Wang et al. 2002a; Yao et al. 2006), margin-based methods (Wei et al. 2014), and bionic methods (Ślęzak and Wróblewski 2003; Cai et al. 2014; Boehm et al. 2011). Accelerators based on redundancy removal (Qian et al. 2010) or modern parallel techniques (Qian et al. 2014) are also available.

✉ Fan Min
minfanphd@163.com

[1] School of Computer Science, Southwest Petroleum University, Chengdu 610500, China

Cost-sensitive feature selection has attracted considerable research interest in recent years (Min et al. 2011; Yang et al. 2013; Jia et al. 2013; Li et al. 2014a; Fan et al. 2015). Specifically, the test cost (Min and Liu 2009; Turney 1995, 2000; Yang et al. 2013) of collecting data items is frequently considered. A number of cost-sensitive feature selection problems have been addressed, including the minimal test cost attribute reduction problem (Min et al. 2011), feature selection with test cost constraint problem (Min et al. 2014), feature selection with positive region problem (Liu et al. 2014), and feature selection with variable costs problem (Zhao and Zhu 2014). Because the test cost is being considered, there are fewer optimal feature subsets than the classical definition of minimal size reducts. Hence, it is more challenging to find an optimal feature subset. To deal with this issue, bionic algorithms (Ślęzak and Wróblewski 2003; Cai et al. 2014; Jensen and Shen 2003) and random algorithms (Li et al. 2014b) have been employed to produce a population of candidate feature subsets. The design of a heuristic function in Min et al. (2011) enables parameter adjustments to be made in order to obtain candidate feature subsets. However, the runtime of conventional approaches is often uncontrollable, or the parameters need to be adjusted for different datasets. This poses the question of whether it is possible to design a simple mechanism to balance the trade-off between the quality of the feature subset and the runtime.

In this paper, we present a semi-greedy feature selection framework, and apply it to the problem of feature selection with test cost constraints (FSTC) (Min et al. 2014). Three techniques are employed, namely the design of the heuristic function, a random selection strategy, and a competition strategy. The design of the heuristic function is the same as for greedy heuristics, and is problem dependent. Our function considers the information gain of the current feature and the test cost of the feature. The random selection strategy is the major difference between our semi-greedy heuristics and greedy heuristics. Greedy heuristics always select the local optimum, i.e., the best feature in the current round. In contrast, semi-greedy heuristics choose one feature at random from the top k features. In this way, different feature subsets can be obtained. After running the feature selection algorithm p times, we have p candidate feature subsets. Finally, the competition strategy obtains the feature subset with the best quality from among these p candidates.

Semi-greedy heuristics have a runtime that is proportional to p. Thus, the runtime can be controlled by adjusting the population p. The major advantage of semi-greedy heuristics is that they can avoid local optima, thus improving the quality of the feature subset. Many researchers have turned to the study of heuristics with randomization. Kirkpatrick (1984) explored a general framework of simulated annealing in 1984, and the semi-greedy concept was first proposed by Hart (1987) for a vehicle routing problem in 1987. Feo and Resende (1995) later developed a theoretical framework of greedy randomized adaptive search procedures (GRASP). These approaches are applicable to most combinatorial optimization problems (Papadimitriou and Steiglitz 1998; Aarts and Korst 1988), but there is little in the literature on their application to feature selection problems, let alone cost-sensitive ones.

We conducted experiments using our semi-greedy heuristic method to determine the optimal settings of the parameters p and k, and compared its performance with that of a classic greedy heuristic algorithm (GHA) and an information gain-based λ-weighted greedy heuristic algorithm (λ-GHA) (Min et al. 2014). The experiments tested the algorithms using four well-known UCI datasets (Blake and Merz 1998). As no test costs are provided for these datasets, we generated them according to three representative distributions, namely Uniform, Normal, and Pareto. To evaluate the performance of the algorithms, we defined three metrics, the Finding optimal factor (FOF), minimal consistency fraction (MCF), and average consistency fraction (ACF).

The experimental results indicate that: (1) $k = 2$ is a good setting for the different datasets under all three test cost distributions; (2) it is unnecessary to set $p > 50$; (3) our algorithm outperforms GHA in terms of the quality of results; and (4) for the same population, our algorithm generally outperforms λ-GHA in terms of the quality of results with approximately the same runtime.

The remainder of this paper is organized as follows. Section 2 revisits the definition of the FSTC problem, and introduces some preliminary knowledge regarding test-cost-independent decision systems and positive regions. The three evaluation metrics are also discussed. Section 3 describes our proposed algorithm. Section 4 presents experimental results from four UCI datasets with three test cost distributions. The parameter settings are discussed in detail in this section. Finally, we give our concluding remarks in Sect. 5.

2 Problem definition

In this section, we briefly introduce some preliminary knowledge, including the test-cost-sensitive systems and the FSTC problem. Three metrics for evaluating the quality of the results are also defined.

2.1 Test-cost-sensitive decision systems

Decision systems are fundamental data models in data mining.

Definition 1 (Yao 2004) A decision system (DS) S is the 5-tuple:

$$S = (U, C, d, V = \{V_a | a \in C \cup \{d\}, I = \{I_a | a \in C \cup \{d\}), \quad (1)$$

where U is a finite set of objects called the universe, C is the set of features, d is the decision class, V_a is the set of values for each $a \in C \cup \{d\}$, and I_a: $U \rightarrow V_a$ is an information function for each $a \in C \cup \{d\}$.

Table 1 gives an example of a DS, where $U = \{x_1, x_2, x_3, x_4, x_5\}$, C = {Sneeze, Cough, Fever, Dizziness}, and d = Influenza.

There are many test-cost-sensitive DSs. A hierarchy of cost-sensitive DSs is given in Min and Liu (2009) from a test cost perspective. In this context, the model of a test-cost-independent DS is used. This is the simplest but most widely used DS.

Definition 2 (Min et al. 2011) A test-cost-independent (TCI) DS S is the 6-tuple:

$$S = (U, C, d, V, I, c), \quad (2)$$

where U, C, d, V, and I have the same meanings as in Definition 1, and $c : C \rightarrow \mathbb{R}^+ \cup \{0\}$ is the test cost function.

The simplest representation of c is to employ a vector c = [$c(a_1)$, $c(a_2)$, ..., $c(a_{|c|})$]. For any $A \subseteq C$, there is

$$c(A) = \sum_{a \in A} c(a). \quad (3)$$

Equation (3) indicates that the test costs are independent of each other. If all elements of the vector c are equal, the TCI-DS degenerates into a DS. Consequently, in this paper, we assume that there are no free cost tests. Table 2 presents a cost vector corresponding to Table 1.

The positive region (Pawlak 1982b) is often employed to judge the quality of a reduct. Any $\emptyset \neq B \subseteq C \cup \{d\}$ determines an indiscernibility relation $I(B)$ on U. A partition determined by B is denoted by U / $I(B)$, or U / B for brevity. Let $\underline{B}(X)$ denote the B-lower approximation of X.

Table 1 An exemplary decision system

Patient	Sneeze	Cough	Fever	Dizziness	Influenza
x_1	Yes	Yes	Yes	Yes	Yes
x_2	No	Yes	No	No	No
x_3	Yes	No	No	No	No
x_4	No	Yes	Yes	Yes	Yes
x_5	Yes	No	Yes	No	No

The positive region of d with respect to $B \subseteq C$ is defined as $POS_B(d) = \bigcup_{X \in U/\{d\}} \underline{B}(X)$.

2.2 Feature selection with test cost constraint

In rough set theory (Pawlak 1982b; Lin and Cercone 1996), feature selection (Zhao et al. 2013) is often referred to as attribute reduction. Different reduct problems have extended from the study of rough sets, such as decision theoretic rough sets (Yao and Zhao 2008), fuzzy rough sets (Zhang et al. 2012; Zhai et al. 2013), and neighborhood rough sets (Hu et al. 2008).

In real-world applications, we are sometimes given limited test costs to obtain the feature values. In these cases, the classification accuracy must be sacrificed to ensure the test cost remains within budget. This type of feature selection problem is defined as the feature selection with test cost constraint described in Problem 1. Compared with classical feature selection, the FSTC problem is more general.

Problem 1 (Min et al. 2014) The feature selection with test cost constraint (FSTC) problem:

Input: $S = (U, C, d, V, I, c)$;

Output: $B \subseteq C$;

Constraint: $c(B) \leq m$;

Optimization objective: (1) max $|POS_B(d)|$; and (2) min $c(B)$.

The output B is the optimal feature subset that meets the test cost constraint. There are two optimization objectives, which are typically not of equal importance. The primary objective is to keep the feature subset optimal from the viewpoint of the positive region. If more than one feature subset satisfies the primary objective, we select the best one according to the secondary objective. The output of Problem 1 never contains redundant features. We can define this type of feature subset as follows.

Definition 3 (Min and Zhu 2011) Let $S = (U, C, d, V, I)$ be a DS and $R \subseteq C$.

R is a sub-reduct if and only if, $\forall a \in R$, $POS_{R-\{a\}}(d) \subset POS_R(d)$.

Consequently, we use the sub-reduct instead of the feature subset in the following context.

Table 2 An exemplary cost vector

a	Sneeze	Cough	Fever	Dizziness
$c(a)$	\$2	\$25	\$10	\$40

2.3 Evaluation metrics

To demonstrate the performance of the algorithms for the FSTC problem, we introduce three statistical evaluation metrics. First, we would like to know whether the results are optimal. We can evaluate the performance of the algorithm in terms of its accuracy from a statistical viewpoint. The finding optimal factor (FOF) (Min et al. 2011) is defined as

$$\text{op} = k/K, \tag{4}$$

where K is the number of experiments and k is the number of successful searches of an optimal sub-reduct.

Second, even if the sub-reduct is not optimal, the quality can be evaluated by some quantitative metrics.

Definition 4 Let $S = (U, C, d, V, I, c)$ be a TCI-DS, and R be a sub-reduct. The consistency of R is

$$cs(R) = |POS_R(d)|. \tag{5}$$

Generally, the better the sub-reduct, the greater the value of $cs(R)$. For a dataset with a particular test cost setting, let $Sred(S)$ be the set of all sub-reducts obtained by an exhaustive approach, such as that proposed in Min and Zhu (2011), and R' be the optimal sub-reduct. Then,

$$R' = \arg \max_{R \in Sred(S)} cs(R). \tag{6}$$

Let the number of experiments be K. In the i-th experiment $(1 \le i \le K)$, assume that the sub-reduct computed by the algorithm is R_i and the optimal reduct is R'_i. The minimal consistency fraction (MCF) is defined as

$$\min_{1 \le i \le K} \frac{cs(R_i)}{cs(R'_i)}. \tag{7}$$

This describes the worst-case performance of the algorithm. The bigger the gap between the obtained sub-reduct and the optimal sub-reduct, the smaller the MCF value. Naturally, if all K sub-reducts are optimal, MCF = 1.

Finally, the overall performance of the algorithm is always important. The average consistency fraction (ACF) is defined as

$$\frac{1}{K} \sum_{i=1}^{K} \frac{cs(R_i)}{cs(R'_i)}. \tag{8}$$

3 Algorithm

In this section, a semi-greedy heuristic for feature selection under test cost constraints (SFTC) is described in detail. Our SFTC has the general form stated in Algorithm 1. There are two techniques, one for constructing a candidate sub-reduct and another for obtaining the best result through competition. Next, we present a high-level description of these two techniques.

Algorithm 1 Framework of SFTC

Input: $S = (U, C, d, V, I, c), m, k, p$.
Output: *bestSolution*.
Method: SFTC.

```
1: bestSolution = ∅; // initialize the output
2: for (i = 1 to p) do
3:     s = ConstructCandidate(S, m, k); //Construction
       //Competition
4:     if (s is better than bestSolution) then
5:         bestSolution = s;
6:     end if
7: end for
8: return bestSolution;
```

3.1 Semi-greedy selection

A heuristic function is essential for candidate construction. SFTC selects one feature according to the benefit measured by the heuristic function. Therefore, we first introduce the heuristic function.

Information gain is a feasible heuristic for the feature selection problem. Following the approaches of Wang et al. (2002b) and Min et al. (2011), we designed a heuristic function based on information gain. To describe this function, a number of basic concepts of information gain are needed. We let $S = (U, C, d, V, I, c)$ be a TCI-DS, as defined in Definition 2. Let $P, Q \subseteq C \cup \{d\}$, and the partitions of the universe U introduced by P and Q be $X = \{X_1, X_2, \ldots, X_n\}$ and $Y = \{Y_1, Y_2, \ldots, Y_m\}$, respectively. P can be viewed as a random variable defined on the universe U, and the probability distribution can be determined as follows.

Definition 5 (Wang et al. 2002b) The σ-algebra distributions of P and Q on U are

$$[X : p] = \begin{bmatrix} X_1 & X_2 & \ldots & X_n \\ p(X_1) & p(X_2) & \ldots & p(X_n) \end{bmatrix} \tag{9}$$

and

$$[Y : p] = \begin{bmatrix} Y_1 & Y_2 & \ldots & Y_m \\ p(Y_1) & p(Y_2) & \ldots & p(Y_m) \end{bmatrix}, \tag{10}$$

respectively, where $p(X_i) = \frac{|X_i|}{|U|}$, $i = 1, 2, \ldots, n$, and $p(Y_j) = \frac{|Y_j|}{|U|}$, $j = 1, 2, \ldots, m$.

Definition 6 (Wang et al. 2002b) The information entropy of P is

$$H(P) = -\sum_{i=1}^{n} p(X_i) \log(p(X_i)). \tag{11}$$

Definition 7 (Wang et al. 2002b) The conditional information entropy of Q with respect to P is

$$H(Q|P) = -\sum_{i=1}^{n} p(X_i) \sum_{j=1}^{m} p(Y_j|X_i) \log(p(Y_j|X_i)), \quad (12)$$

where
$p(Y_j|X_i) = |Y_j \cap X_i|/|X_i|, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m.$

Definition 8 (Min et al. 2011) Let $B \subset C$, $a \in C$ and $a \neq B$. The information gain of a with respect to B is

$$f_e(B, a) = H(\{d\}|B) - H(\{d\}|B \cup \{a\}). \quad (13)$$

As described in Problem 1, the primary objective is to maximize the positive region, and the secondary is to minimize the test cost. We now propose the SFTC heuristic function:

$$f(B, a) = \frac{f_e(B, a)}{c(a)}, \quad (14)$$

where $c(a)$ is the test cost of a. Generally, we seek features with larger values of $f(B, a)$.

Algorithm 2 describes the framework of the candidate construction process.

To construct a candidate sub-reduct, SFTC follows the well-known addition-deletion strategy (Yao et al. 2006; Min and Liu 2009), which has two main steps.

Step 1 Continuously add a feature a to B (which is initially the empty set). Let BK be the set of the current k-best features, where k is a user-specified constant to control the randomness of the algorithm. Feature a is selected from BK according to the heuristic function and the roulette wheel selection method (Burke et al. 1995). The probability of feature a being selected is given by

$$p(B, a) = \frac{f(B, a)}{\sum_{a' \in B} f(B, a')}. \quad (15)$$

This step allows for diverse sub-reducts to be obtained. In this step, we also remove features that do not satisfy the test cost constraint to avoid redundant information gain computations. The code is listed in lines 13–17.

Step 2 Remove redundant features, if any exist from the viewpoint of the positive region, from B. Once these two steps have finished, a candidate sub-reduct recorded in B has been successfully constructed.

Algorithm 2 Candidate construction

Input: $S = (U, C, d, V, I, c)$, m.
Output: $B \subseteq C$.
Method: ConstructSolution.

1: $B = \emptyset$; //initialize the output
2: $CA = C$; //unprocessed features
3: $c_l = m$; //available test cost
4: $BK = \emptyset$; //store the best k features
//Construct a sub-reduct through the semi-greedy selection;
5: **while** $(CA \neq \emptyset)$ **do**
6: For any $a \in CA$ where $c(a) \leq c_l$, compute $f(B, a, c)$;
//Addition
7: $BK =$ the set of k-best features based on f;
8: For any $a \in BK$, compute the selection probability $p(B, a)$;
9: Select feature a' from BK according to $p(B, a)$;
10: $B = B \cup \{a'\}$;
11: $CA = CA - \{a'\}$;
12: $c_l = c_l - c(a')$;
//Remove features not satisfying the constraint to accelerate the algorithm
13: **for** (each $a \in CA$) **do**
14: **if** $(c_a > c_l)$ **then**
15: $CA = CA - \{a\}$;
16: **end if**
17: **end for**
18: **end while**
//Remove redundant features from the viewpoint of the positive region
19: **for** (each $a \in B$) **do**
20: **if** $(POS_{B-\{a\}} = POS_B\{d\})$ **then**
21: $B = B - \{a\}$;//a is redundant
22: **end if**
23: **end for**
24: return B;

We illustrate one step of the feature selection process with the following example.

Example 1 Let $C = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and $B = \{a_3\}$. That is, a_3 is already included in the reduct. Our current task is to choose one more feature. Suppose that $hi = [f(B, a_1, c), \ldots, f(B, a_6, c)] = [50, 20, 0, 30, 15, 18]$, and let $k = 3$. $BK = \{a_1, a_4, a_2\}$, since a_1, a_4, and a_2 are the top three features. Accordingly, we compute the probabilities that these features are selected as $sp = [0.5, 0.3, 0.2]$. Therefore, the probabilities of selecting a_1, a_4, and a_2 in the current round will be 0.5, 0.3, and 0.2, respectively.

To compute the conditional entropy of a feature a, we split the data into $|V_a|$ subsets. This takes $O(|V_a||U|)$ time. The conditional entropy can then be computed according to Eq. (12). To add the first feature, each feature must be studied, which has a time complexity of $O(\sum_{a \in C} |V_a||U|)$. Other operations take much less time. Consequently, the overall time complexity of Algorithm 2 is $O(|U||C| \sum_{a \in C} |V_a|)$. However, the average time complexity is much lower, because (1) relatively few features will be selected, and (2) many objects can be removed in the process of reduction (Qian et al. 2010).

3.2 Competition

The use of a competition strategy to obtain better results has been discussed in Min et al. (2011). In this paper, p is a user-specified population variable that controls the number of candidates. In each loop of Algorithm 1, a new candidate is compared with the current best sub-reduct. If the candidate is better, it becomes the current best sub-reduct. Once all p competitions have finished, the winner is kept as the final result. The aim of this competition strategy is to balance the trade-off between the quality of the results with the runtime.

Let R_1 and R_2 be two sub-reducts obtained by Algorithm 1. Naturally, they all meet the test cost constraint. According to Problem 1 and Eq. (5), R_1 is better than R_2 in two cases: (1) $cs(R_1) > cs(R_2)$; or (2) $cs(R_1) > cs(R_2)$ and $c(R_1) < c(R_2)$.

3.3 A running example

The semi-greedy heuristics and competition strategy are interdependent—they are both important to SFTC obtaining good results. We now present a full example for the algorithm.

Example 2 Consider the DS listed in Table 1 and the cost vector listed in Table 2. For brevity, {Sneeze, Cough, Fever, Dizziness} and Influenza are represented by $\{a_1, a_2, a_3, a_4\}$ and d, respectively. One can find that the optimal result is $\{a_2, a_3\}$ with $POS_{\{a_2,a_3\}}(\{d\}) = \{x_1, x_2, x_3, x_4, x_5\}$. We let $k = 2$, $m = 35$, $p = 2$, and compute $H(\{d\}|\{a_4\}) = 0$, $H(\{d\}|\{a_1\}) = 0.95098$, $H(\{d\}|\{a_2\}) = H(\{d\}|\{a_3\}) = 0.55098$, $H(\{d\}) = 0.9709$. Figure 1 illustrates a possible execution of the SFTC algorithm.

The candidate construction and competition processes of SFTC are presented in different boxes. The top two features are highlighted. In each iteration, one of these features is selected at random. After running Algorithm 2 twice, SFTC obtains two candidate sub-reducts $R_1 = \{a_3, a_1\}$ and $R_2 = \{a_3, a_2\}$. The final result is R_2, because $cs(R_2) > cs(R_1)$.

The notation employed throughout the paper is listed in Table 3

4 Experiments

In this section, we perform extensive computational tests to address the following questions.

1. Does SFTC outperform existing greedy heuristics in terms of the quality of results?
2. How does the efficiency of SFTC compare with that of existing algorithms?
3. Does the test cost distribution influence the quality of results?
4. Is there a rational setting of k that is valid for any dataset?
5. For fixed k, how does the performance of SFTC change with an increase in p?
6. How does the performance of SFTC change for different samplings of the Mushroom dataset?

The first three questions concern the overall performance of SFTC, whereas the final three are related to the parameter settings.

4.1 Datasets

Our experiments were performed using four real-world datasets from the UCI repository (Blake and Merz 1998), namely Zoo, Tic-tac-toe, Voting, and Mushroom. Some basic information about these datasets is listed in Table 4, where $|C|$ is the number of features, $|U|$ is the number of instances, and d is the name of the decision.

Different real-world applications have different test cost distributions. However, these four datasets do not include any test cost information. Thus, for statistical purposes, three different distributions (Uniform, Normal, and Pareto)

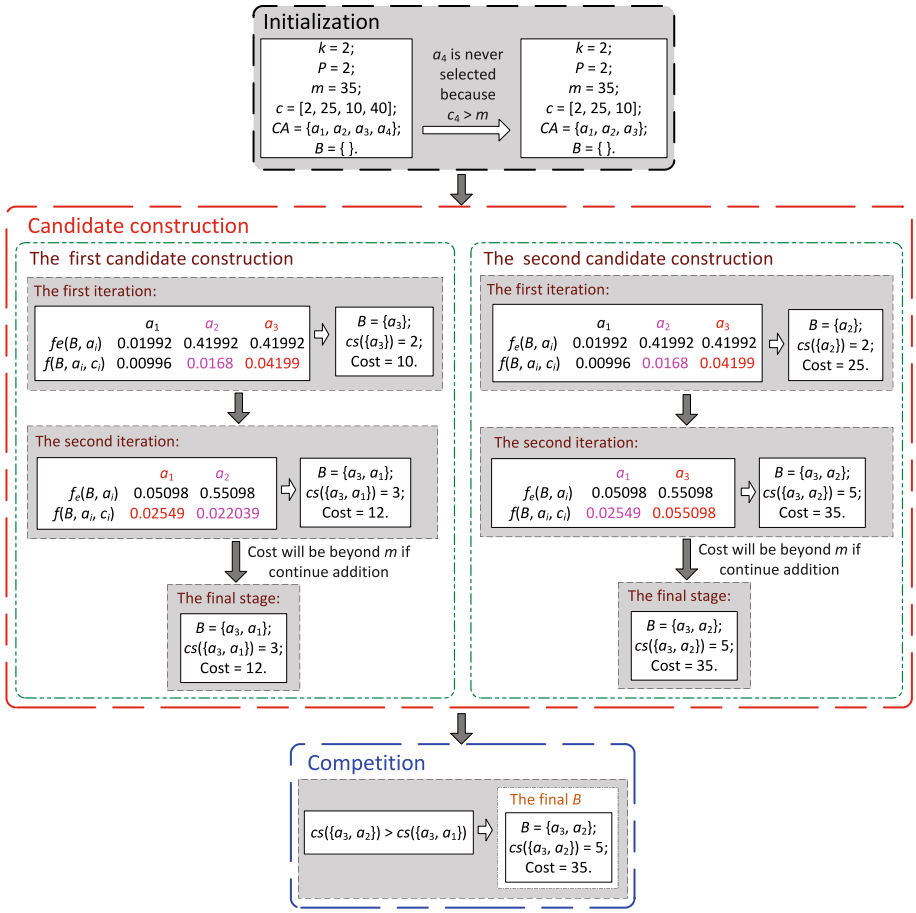

Fig. 1 A running example for SFTC

Table 3 Notation list

Notation	Meaning
S	Test-cost-independent decision system
U	The universe
C	The set of features
d	The decision class
V_a	The set of values for feature a
I_a	The information function for a
c	The test cost function
B	A subset of C
$POS_B(d)$	The positive region of B wrt. d
R	A sub-reduct
$cs(R)$	The consistency of $R \subseteq C$, defined as $POS_R(d)$
$H(P)$	The information entropy of P
$H(Q\|P)$	The conditional information entropy of Q wrt. P
$f_e(B, a)$	The information gain of a respect to B
$Sred(S)$	The set of all sub-reducts
k	The number of candidates for semi-greedy selection
p	The population of the feature subsets for competition
K	The number of experiments for different test cost settings

Table 4 Dataset information

Name	Domain	$\|C\|$	$\|U\|$	d
Zoo	Zoology	16	101	Type
Voting	Society	16	435	Vote
Tic-tac-toe	Game	9	958	Class
Mushroom	Botany	22	8124	Class

were generated for each dataset. In the experiments, the test costs were chosen to be integers in the range [1100].

4.2 Experimental design

We designed four groups of experiments to answer the questions asked at the beginning of this section.

Experiment 1 Test the influence of k. If k is small, it is more likely that locally optimal choices will be made. In the extreme case, i.e., $k = 1$, SFTC coincides with GHA. With large k, SFTC can obtained more diverse candidate sub-reducts.

However, many of these will be of poor quality. Therefore, we wish to determine whether there is a setting of k that is valid for any dataset with any test cost distribution. We set $p = 9$, and test SFTC with different settings of k on the four datasets. In this group of experiments, k was set to $1, 2, \ldots, 8$. The budget of the cost constraint was restricted to $0.8c'$, where c' is the cost of the optimal reduct in the MTR problem (Min et al. 2011). For example, if the minimal test cost of making an accurate classification were \$100, then m would be \$80. The test cost constraint was kept fixed throughout the following experiments.

Experiment 2 Test the influence of p. We set $k = 2$ and varied p on the four datasets with three distributions. Empirically, an increase in p caused the effectiveness to improve significantly and the runtime to increase linearly. However, there must be a limit to p, as k remains fixed. This conjecture will be proved in the following. We deliberately let p vary from 2 to 56 in steps of 2, i.e., $p = 2, 4, 6, \ldots, 56$. We conducted similar experiments for λ-GHA with different values of λ. In this way, we obtained comparable results.

Experiment 3 This experiment was designed to answer question 6. First, we sampled the Mushroom dataset to produce sub-datasets with different numbers of objects. The sampling was based on the uniform distribution. That is, all objects had an equal probability of being chosen. Second, the SFTC algorithm was applied to each sub-dataset. In this experiment, the Mushroom dataset was divided into eight sub-datasets of different scale. The number of instances varied from $0.125\,|U|$ to $|U|$, and the step length was set to $0.125|U|$.

Experiment 4 This experiment was designed to answer questions 1 and 2. We compared the efficiency and effectiveness of SFTC with that of two greedy heuristic algorithms, namely GHA and λ-GHA. For the algorithms that adopted a competition strategy, we set the number of candidate sub-reducts to 9.

All groups of experiments were run on the four datasets with three test cost distributions. Each experiment was performed with 1000 different test cost settings on each dataset.

4.3 The influence of parameter settings

We now examine the influence of different k settings on SFTC. In Fig. 2, the FOF results exhibit a similar trend with respect to k on all four datasets and for all three test cost distributions. We can observe the following.

1. The optimal setting for k varies among datasets and distributions. With the Normal and Pareto distributions, the optimal setting on all datasets was $k = 2$; for the Uniform distribution, the optimal setting was $k =$

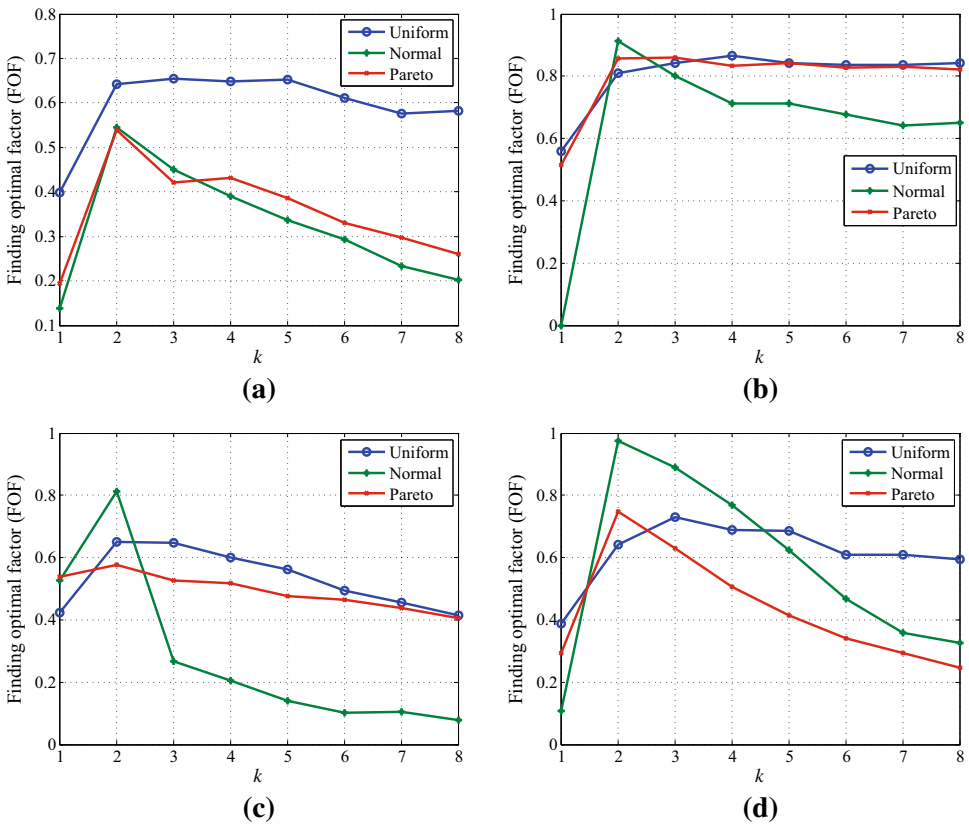

Fig. 2 Trend of FOF with respect to k on four datasets: **a** Zoo, **b** Tic-tac-toe, **c** Voting, **d** Mushroom

3, 4, 2, 3 for the Zoo, Tic-tac-toe, Voting, and Mushroom datasets, respectively. With an increase in k, there was a tendency for the performance of SFTC to increase until some maximum, and then decrease. This trend was particularly strong for the Normal and Pareto distributions.

2. The performance of SFTC tends to be worst when $k = 1$. In this case, SFTC coincides with GHA. The competition strategy is of no use, because SFTC produces only one sub-reduct, which is formed by the best feature in each iteration. In particular, FOF was 0 on the Tic-tac-toe dataset with the Normal distribution. That is, SFTC never generated an optimal sub-reduct. When $k > 1$, the algorithm performed much better in most cases.
3. There is a big gap in terms of the FOF between $k = 2$ and $k \neq 2$; this is more obvious for the Normal and Pareto distributions than for the Uniform distribution. In particular, FOF increased significantly when k changed from 1 to 2. For example, for the Mushroom dataset with the Normal and Pareto distributions, the FOFs were 0.107 and 0.293, respectively, when $k = 1$, but increased to 0.974 and 0.749 when we set $k = 2$. Therefore, $k = 2$ seems to be a rational setting for all of the datasets we tested.

For $k = 2$, the trend in FOF with respect to p for each dataset with the three distributions is depicted in Fig. 3. It is obvious that the competition strategy significantly improved the quality of results, especially when the Normal distribution was used. There is a tendency for FOF to change rapidly when $p \leq 8$, after which the change becomes less observable. In addition, FOF reached an extreme when $p = 50$, after which it stabilized. That is, bigger values of p do not give any further improvement in the quality of results. Of the different test cost distributions, SFTC performed best with the Normal distribution.

In contrast, λ-GHA uses different heuristic functions with different λ values to produce diverse candidate sub-reducts. Figure 4 shows the trend of FOF with respect to λ. The trend is similar to that of SFTC, but the overall performance is different. First, λ-GHA became stable earlier than SFTC. There was typically no improvement once λ exceeded 20, especially for the Uniform and Pareto distributions. Second, FOF generally remained within the range [0.74, 0.85]. The worst FOF was about 0.23 (Zoo dataset with Pareto distribution), and the best was 1 (Voting with Normal distribution). Third, except for the Voting dataset, the results given by SFTC are more acceptable than those of λ-GHA, because the performance is more stable. Fourth, unlike for SFTC, λ-GHA does not have a best test cost distribution.

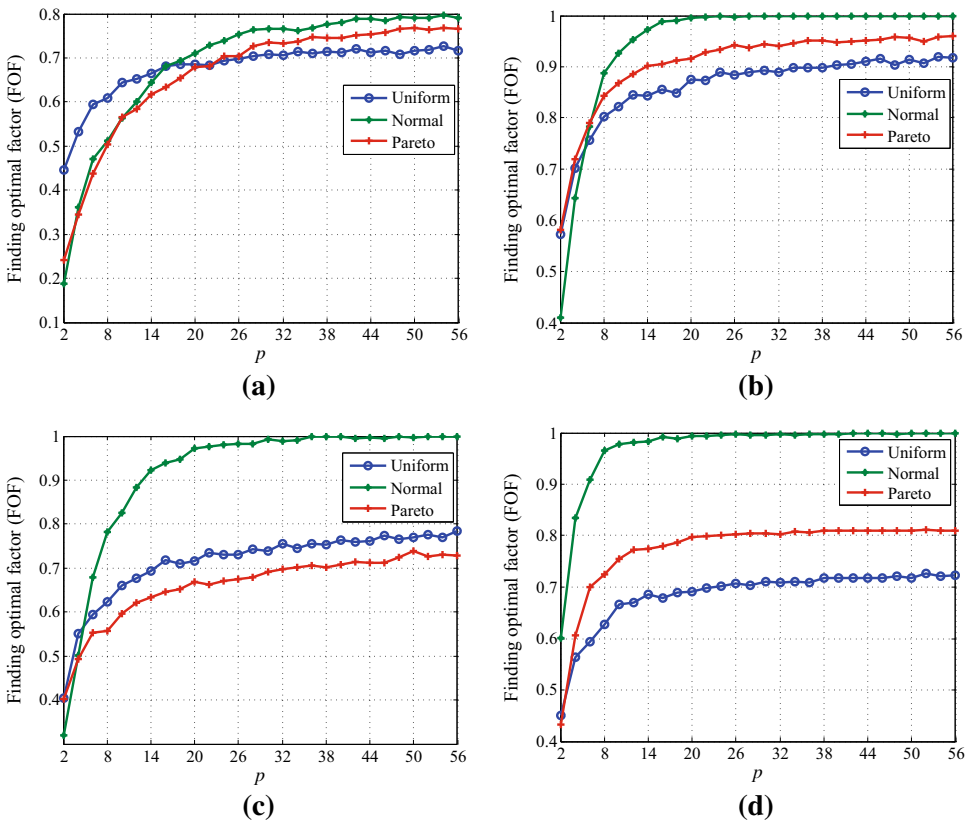

Fig. 3 Trend of FOF with respect to p on four datasets: **a** Zoo, **b** Tic-tac-toe, **c** Voting, **d** Mushroom

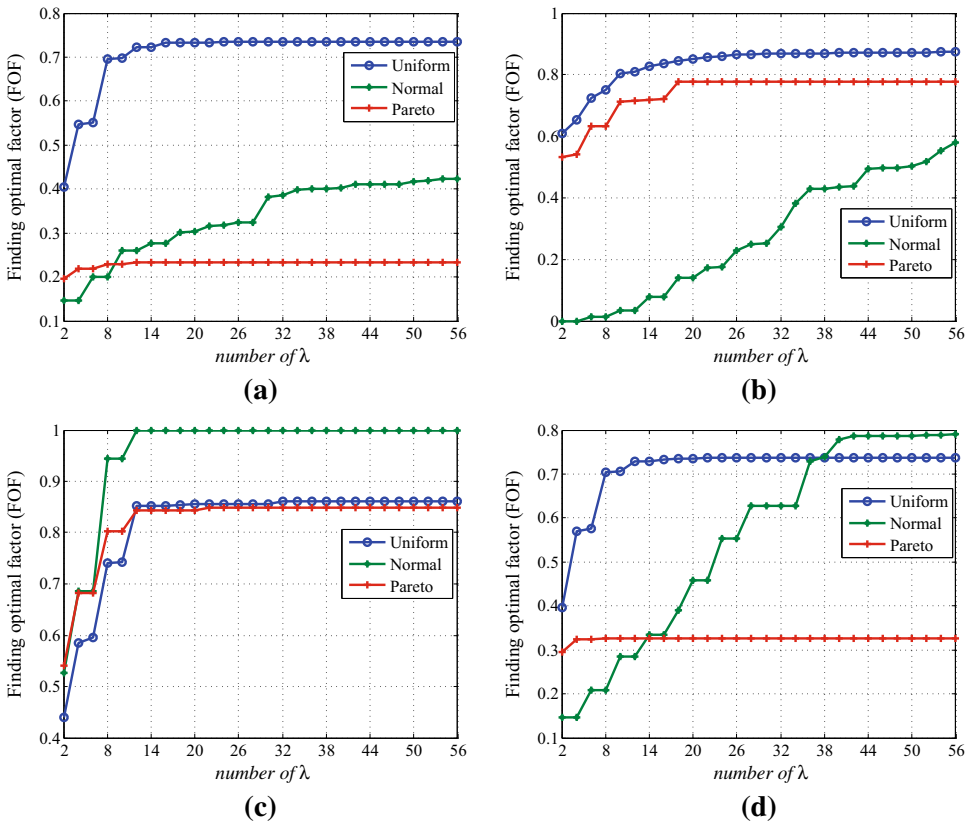

Fig. 4 Trend of FOF with respect to λ on four datasets: **a** Zoo, **b** Tic-tac-toe, **c** Voting, **d** Mushroom

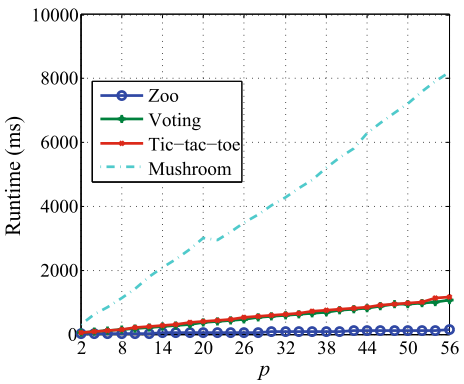

Fig. 5 Trend of runtime with respect to p for the Uniform distribution on the Connect-4 datasets

Figure 5 shows the runtime with respect to p for the four datasets with the Uniform test cost distribution. We compared the runtime of a single experiment with fixed variables, where each value is the mean of 1000 test cost settings. We can observe that the runtime is proportional to p. Thus, we can control the runtime by adjusting p.

Figure 6 shows the change in FOF and MCF with different instance sizes of the Mushroom dataset. We divided the dataset into eight sub-datasets of different sizes. As we can see from Fig. 6a, better values of FOF were obtained for bigger instance sizes under the Normal and Pareto test cost distributions. For instance, FOF was just 0.677 with 1015 instances, but this increased to 0.974 with 8124 instances. However, with the Uniform distribution, FOF was stable for different dataset sizes. From Fig. 6b, we can observe that MCF decreased as the dataset size increased with the Uniform and Pareto test cost distributions.

However, MCF remained relatively stable under the Normal distribution.

4.4 Effectiveness comparison

The experimental results comparing the effectiveness of GHA, λ-GHA, and SFTC are presented in Table 5. The optimal results for all datasets with the different test cost distributions are noted in italic type. The main observations that can be made across the different datasets and test cost distributions are as follows.

1. GHA is generally the worst of the three algorithms. It is not suitable for the FSTC problem, because the results for all three metrics are unacceptable on all of the datasets tested. For example, the FOF score of GHA with the Zoo dataset and Pareto distribution was just 0.194, whereas λ-GHA and SFTC scored 0.23 and 0.539, respectively.
2. λ-GHA is much better than GHA in all cases. In some conditions, it also outperformed SFTC. With the

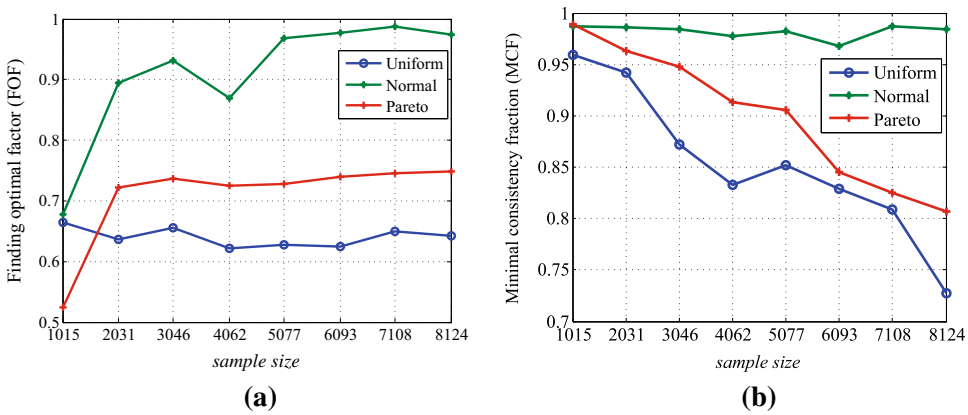

Fig. 6 Performance of SFTC in terms of: **a** FOF, **b** MCF

Table 5 Comparison of the three algorithms in terms of effectiveness metrics (FOF, MCF, ACF) for 1000 test cost settings

Dataset	Test cost distribution	GHA			λ-GHA			SFTC		
		FOF	MCF	ACF	FOF	MCF	ACF	FOF	MCF	ACF
Zoo	Uniform	0.398	0.4556	0.9493	*0.699*	*0.7888*	*0.9867*	0.642	0.72222	0.98134
	Normal	0.139	0.7659	0.9063	0.255	0.7835	0.9361	*0.544*	*0.7935*	*0.9730*
	Pareto	0.194	0.5714	0.9273	0.23	0.8351	0.9457	*0.539*	*0.8428*	*0.9792*
Voting	Uniform	0.424	0.9044	0.9945	*0.840*	*0.9834*	*0.9994*	0.655	0.9626	0.9979
	Normal	0.527	0.9929	0.9969	*1*	*1*	*1*	0.813	0.9906	0.9994
	Pareto	0.537	0.8936	0.9937	*0.843*	*0.9738*	*0.9992*	0.592	0.9509	0.9969
Tic-tac-toe	Uniform	0.559	0.6090	0.9689	0.806	0.8262	0.9939	*0.811*	*0.8349*	*0.9955*
	Normal	0	0.9787	0.9787	0.033	0.9387	0.9795	*0.912*	*0.9887*	*0.9994*
	Pareto	0.515	0	0.9362	0.714	*0.8052*	0.9892	*0.856*	0.4055	*0.9894*
Mushroom	Uniform	0.388	0.2330	0.9724	*0.719*	0.4314	0.9937	0.642	*0.7269*	*0.9943*
	Normal	0.107	0.9551	0.9989	0.288	0.9239	0.9992	*0.974*	*0.9839*	*0.9999*
	Pareto	0.293	0.8064	0.9890	0.327	0.8064	0.9916	*0.749*	*0.8220*	*0.9937*

Voting dataset, in particular, λ-GHA gave the best results in terms of all three metrics. The FOF scores with the Uniform and Pareto distributions are above 0.8, and that with the Normal distribution is 1.

3. In most cases, SFTC performs significantly better than the other two algorithms. In particular, for the Tic-tac-toe and Mushroom datasets with the Normal distribution, SFTC achieved FOF scores of close to 1. That is, more than 900 optimal sub-reducts were obtained by SFTC over the 1000 experiments. However, on the Tic-tac-toe and Mushroom datasets, GHA and λ-GHA scored just 0 and 0.107, 0.033 and 0.288, respectively. Although the FOF given by SFTC is not always good enough, SFTC is a good algorithm from the viewpoint of MCF.

4.5 Efficiency comparison

A comparison of the runtimes of SFTC, GHA, and λ-GHA is depicted in Table 6. We can observe that GHA is the most efficient algorithm. This is because GHA only needs to construct one reduct. In contrast, SFTC must construct p reducts. Hence, the runtime of SFTC is about p times that

Table 6 Runtime (ms) on four datasets (mean values for 1000 test cost settings)

Dataset	Distribution	GHA	λ-GHA	SFTC
Zoo	Uniform	2.47	20.197	21.497
	Normal	1.21	10.762	11.052
	Pareto	1.67	13.996	15.354
Voting	Uniform	17.67	160.69	166.86
	Normal	12.60	113.12	114.09
	Pareto	15.72	163.69	175.39
Tic-tac-toe	Uniform	21.35	184.64	189.94
	Normal	18.35	160.79	161.92
	Pareto	19.47	164.84	168.55
Mushroom	Uniform	161.7	1219.7	1325.4
	Normal	87.68	843.12	844.16
	Pareto	99.35	855.13	887.57

of GHA. The runtime of λ-GHA is determined by the number of λ values we set. In our experiments, λ-GHA had a similar runtime to that of SFTC, because we set the number of λ values to p.

4.6 Discussion

In general, SFTC is more appropriate than GHA and λ-GHA for FSTC problems. There are two main causes: (1) the greedy schema; and (2) the competition strategy. GHA always chooses the best feature in each iteration, and thus only produces one reduct. Once GHA has made a wrong choice, the optimal reduct cannot be constructed. λ-GHA follows the same greedy schema. However, the heuristic function also takes into consideration the user-specified parameter λ. In contrast, SFTC does not always make the locally optimal choice, and is therefore able to find different reducts.

The competition strategy is not applicable to GHA, which only constructs one reduct. λ-GHA constructs a number of reducts according to the setting of λ. The best reduct is then selected using the competition strategy. Therefore, λ-GHA is better than GHA in terms of the quality of results. With the same population, and therefore the same runtime, SFTC outperforms λ-GHA in terms of FOF, MCF, and ACF.

The semi-greedy selection stage of SFTC can be parallelized to run on p processors. λ-GHA can also be run on a parallel system, with different processors taking different λ values. In this way, the efficiency of these algorithms can be made comparable to that of GHA. In future work, we will implement a parallel version of these algorithms using Hadoop/MapReduce. This task is less challenging than the work of Qian et al. (2014), who employed a somewhat complex mechanism.

We can also observe that different algorithms are appropriate for different datasets. For example, λ-GHA outperformed SFTC on the Voting dataset. The test cost distribution also influenced the quality of the results. SFTC generally performed better with the Normal test cost distribution. One possible reason is that the Normal distribution satisfies the central limit theorem, which reduces the significance of the test cost.

5 Conclusions

In this paper, we proposed a semi-greedy heuristic method of feature selection under test cost constraint. The results of experiments on four datasets with three test cost distributions indicated that $k = 2$ is a rational setting. Based on this setting, the value of p can be adjusted to further improve performance. The value of p directly determines the number of candidate sub-reducts involved in the competition phase. Given the runtime constraints, we can set an appropriate value of p. Compared with some existing algorithms, the proposed algorithm is simple yet effective and efficient.

There are a number of areas for further study. The first is to apply the semi-greedy mechanism to other feature selection problems. The second is to design a heuristic function using measures such as the positive region, discernibility measure, and Gini index. The third is to design a parallel version of the algorithm and run it on Hadoop/MapReduce.

Acknowledgments This work is in part supported by the National Natural Science Foundation of China under Grant No. 61379089.

References

Aarts E, Korst J (1988) Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing. Wiley, New York

Al-Khatib W, Day YF, Ghafoor A, Berra PB (1999) Semantic modeling and knowledge representation in multimedia databases. IEEE Trans Knowl Data Eng 11(1):64–80

Bargiela A, Pedrycz W (2012) Granular computing: an introduction. Springer Science and Business Media, Berlin

Blake C, Merz CJ (1998) UCI Repository of machine learning databases

Boehm O, Hardoon DR, Manevitz LM (2011) Classifying cognitive states of brain activity via one-class neural networks with feature selection by genetic algorithms. Int J Mach Learn Cybern 2(3):125–134

Burke EK, Newall JP, Weare RF (1995) A memetic algorithm for university exam timetabling. In: Practice and theory of automated timetabling, pp 241–250. Springer, Berlin

Cai JL, Zhu W, Ding HJ, Min F (2014) An improved artificial bee colony algorithm for minimal time cost reduction. Int J Mach Learn Cybern 5(5):743–752

Chen DG, Wang CZ, Hu QH (2007) A new approach to attribute reduction of consistent and inconsistent covering decision systems with covering rough sets. Inform Sci 177(17):3500–3518

Fan AJ, Zhao H, Zhu W (2015) Test-cost-sensitive attribute reduction on heterogeneous data for adaptive neighborhood model. Soft Comput 1–12

Feo TA, Resende MG (1995) Greedy randomized adaptive search procedures. J Glob Optim 6(2):109–133

Gu SM, Wu WZ (2013) On knowledge acquisition in multi-scale decision systems. Int J Mach Learn Cybern 4(5):477–486

Hart JP, Shogan AW (1987) Semi-greedy heuristics: an empirical study. Op Res Lett 6(3):107–114

He X, Min F, Zhu W (2013) Parametric rough sets with application to granular association rule mining. Math Probl Eng 2013:1–13

Hu QH, Xie ZX, Yu DR (2007) Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation. Pattern Recognit 40(12):3509–3521

Hu QH, Yu DR, Liu JF, Wu CX (2008) Neighborhood rough set based heterogeneous feature subset selection. Inform Sci 178(18):3577–3594

Janusz A, Ślęzak D (2014) Rough set methods for attribute clustering and selection. Appl Artif Intell 28(3):220–242

Jensen R, Shen Q (2003) Finding rough set reducts with ant colony optimization. In: Proceedings of the 2003 UK workshop on computational intelligence, vol 1, issue 2

Jia XY, Liao WH, Tang ZM, Shang L (2013) Minimum cost attribute reduction in decision-theoretic rough set model. Inform Sci 219:151–167

Kirkpatrick S (1984) Optimization by simulated annealing: quantitative studies. J Statis Phys 34(5–6):975–986

Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1):273–324

Li XJ, Zhao H, Zhu W (2014a) An exponent weighted algorithm for minimal cost feature selection. Int J Mach Learn Cybern 1–10

Li J, Zhao H, Zhu W (2014b) Fast randomized algorithm with restart strategy for minimal test cost feature selection. Int J Mach Learn Cybern 6(3):435–442

Lin TY (1998) Granular computing on binary relations I: data mining and neighborhood systems. Rough Sets Knowl Discov 1:107–121

Lin TY, Cercone N (1996) Rough sets and data mining: analysis of imprecise data. Kluwer Academic Publishers, Dordrecht

Liu JB, Min F, Zhao H, Zhu W (2014) Feature selection with positive region constraint for test-cost-sensitive data. Trans Rough Sets XVIII:23–33

Min F, He HP, Qian Y, Zhu W (2011) Test-cost-sensitive attribute reduction. Inform Sci 181(22):4928–4942

Min F, Hu QH, Zhu W (2014) Feature selection with test cost constraint. Int J Approx Reason 55(1):167–179

Min F, Liu QH (2009) A hierarchical model for test-cost-sensitive decision systems. Inform Sci 179(14):2442–2452

Min F, Zhu W (2011) Optimal sub-reducts with test cost constraint. Rough Sets Knowl Technol 57–62

Modrzejewski M (1993) Feature selection using rough sets theory. In: Machine learning: ECML-93, pp 213–226. Springer, Berlin

Papadimitriou CH, Steiglitz K (1998) Combinatorial optimization: algorithms and complexity. Courier Dover Publications, USA

Pawlak Z (1982a) Rough sets. Int J Comput Inform Sci 11:341–356

Pawlak Z (1982b) Rough sets. Int J Comput Inform Sci 11(5):341–356

Pedrycz W (2001) Granular computing: an emerging paradigm, vol 70. Springer Science and Business Media, Berlin

Pedrycz W (2013) Granular computing: analysis and design of intelligent systems. CRC press, Boca Raton

Qian YH, Liang JY, Pedrycz W, Dang CY (2010) Positive approximation: an accelerator for attribute reduction in rough set theory. Artif Intell 174(9–10):597–618

Qian J, Miao DQ, Zhang ZH, Yue XD (2014) Parallel attribute reduction algorithms using mapreduce. Inform Sci 279:671–690

Qin YX, Zheng DQ, Zhao TJ (2012) Research on search results optimization technology with category features integration. Int J Mach Learn Cybern 3(1):71–76

Skowron A, Rauszer C (1992) The discernibility matrices and functions in information systems. Intell Decis Support 331–362

Ślęzak D (2002) Approximate entropy reducts. Fundamenta informaticae 53(3–4):365–390

Ślęzak D, Wróblewski J (2003) Order based genetic algorithms for the search of approximate entropy reducts. In: Rough sets, fuzzy sets, data mining, and granular computing, pp 308–311. Springer, Berlin

Słowiński R (1992) Intelligent decision support: handbook of applications and advances of the rough sets theory, vol 11. Springer Science and Business Media, Berlin

Turney PD (1995) Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. J Artif Intell Res 2:369–409

Turney PD (2000) Types of cost in inductive concept learning. Proceedings of the workshop on cost-sensitive learning at the 17th ICML, pp 1–7

Wang GY, Zhang QH (2009) Granular computing based cognitive computing. In: 8th IEEE international conference on cognitive informatics, IEEE, pp 155–161

Wang GY, Yu H, Yang DC (2002a) Decision table reduction based on conditional information entropy. Chin J Comput 2(7):759–766

Wang GY, Yu H, Yang DC (2002b) Decision table reduction based on conditional information entropy. Chin J Comput 25(7):759–766

Wang XZ, Zhai JH, Lu SX (2008) Induction of multiple fuzzy decision trees based on rough set technique. Inform Sci 178(16):3188–3202

Wei P, Ma PJ, Hu QH, Su XH, Ma CQ (2014) Comparative analysis on margin based feature selection algorithms. Int J Mach Learn Cybern 5(3):339–367

Wu WZ, Leung Y (2011) Theory and applications of granular labelled partitions in multi-scale decision tables. Inform Sci 181(18):3878–3897

Yang XB, Qi YS, Song XN, Yang JY (2013) Test cost sensitive multigranulation rough set: model and minimal cost selection. Inform Sci 250:184–199

Yao YY (2000) Granular computing: basic issues and possible solutions. In: Proceedings of the 5th joint conference on information sciences, vol 1, pp 186–189

Yao YY (2004) A partition model of granular computing. In: Transactions on rough sets I, pp 232–253. Springer, Berlin

Yao YY, Zhao Y, Wang J (2006) On reduct construction algorithms. In: Rough sets and knowledge technology, pp 297–304. Springer, Berlin

Yao JT, Vasilakos AV, Pedrycz W (2013) Granular computing: perspectives and challenges. IEEE Trans Syst Man Cybern Part C Appl Rev 43(6):1977–1989

Yao YY, Zhao Y (2008) Attribute reduction in decision-theoretic rough set models. Inform Sci 178(17):3356–3373

Zhai JH, Zhai MY, Bai CY (2013) An improved algorithm for calculating fuzzy attribute reducts. J Intell Fuzzy Syst Appl Eng Technol 25(2):303–313

Zhang WX, Wei L, Qi JJ (2005) Attribute reduction in concept lattice based on discernibility matrix. In: Rough sets, fuzzy sets, data mining, and granular computing, pp 157–165. Springer, Berlin

Zhang X, Zhou B, Li P (2012) A general frame for intuitionistic fuzzy rough sets. Inform Sci 216:34–49

Zhao H, Min F, Zhu W (2013) Test-cost-sensitive attribute reduction of data with normal distribution measurement errors. Math Probl Eng 2013:1–12

Zhao H, Zhu W (2014) Optimal cost-sensitive granularization based on rough sets for variable costs. Knowl Based Syst 65:72–82

Zhu W, Wang F (2003) Reduction and axiomization of covering generalized rough sets. Inform Sci 152(1):217–230