Dynamic Tabu Search for Dimensionality Reduction in Rough Set

ZALINDA OTHMAN, AZURALIZA ABU BAKAR, SALWANI ABDULLAH, MOHD ZAKREE
AHMAD NAZRI AND NELLY ANAK SENGALANG
Faculty Of Information Sciences and Technology
Universiti Kebangsaan Malaysia
43600 Bangi, Selangor
MALAYSIA
zalinda@ftsm.ukm.my http://dmo.ukm.my

Abstract:- This paper proposed a dynamic tabu search (DTSAR) that incorporated a dynamic tabu list to solve an attribute reduction problem in rough set theory. The dynamic tabu list is use to skip the aspiration criteria and to promote faster running times. A number of experiments have been conducted to evalute the performance of the proposed technique with other published metaheuristic techniques, rough sets and decision tree. DTSAR shown promising results on reduct generation time. It ranges between 0.20 minutes to 22.18 minutes. For comparison on the performance on number of reduct produced, DTSAR is on par with other metaheuristic techniques. DTSAR outperforms some techniques on certain dataset. Quality of classification rules generated by adopting DTSAR was comparable with two other methods i.e. Rough Set and Decision Trees.

Keywords: Tabu search, attribute reduction, rough set, computational intelligence, dynamic Tabu list

1. Introduction

For the last decade, dealing with data is not a problem for an organisation as the amount of data that they are dealing with is just a small. Today, there are too much of data that we can handle. Data can be from business transaction, educational reports, health reports, satellite images, scientific data and many more. With these huge collections of data, a better solution for information retrieval is needed to improve the managerial decision making process. Data mining plays an important part to deliver a higher level of information, called knowledge to the users. It is a task of discovering interesting patterns from large amount of data from databases (Han et al. 2006). According to Han and Kamber (2001), data mining consists of few stages such as data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation and knowledge presentation. The initial stage in data mining are known as the data pre-processing, such as data cleaning, data integration, data selection / data reduction and data transformation. Data reduction in data pre-processing stage is one of the crucial steps in data mining. The purpose of data reduction is to reduce the data dimension into a smaller dimension but produces almost the same analytical result as the original data.

Dimensionality reduction is the study of a method to reduce the number of data dimensions that used to describe the object. It is aim to remove irrelevant and redundant data thus to reduce the computational cost and to improve the data quality (Dash and Liu, 2008). In dimensionality reduction, the method of attribute subset selection is applied. According to Han & Kamber (2001), the aim of attribute subset selection is to find a minimum set of attributes such that the result given by the probability distribution of the data classes is almost the same to the original distribution obtained using all attributes. There are approaches used in dimension reduction; i.e the heuristic methods and rough set theory based reducts computation.

The main focus in this research is to investigate Tabu search algorithm for dimensionality reduction problem in rough set theory and use the reduce dataset to generate good classification rules. This paper proposes a rough set classifier model based using dynamic Tabu list.

This paper is organized as follows. In the next section, we briefly given the principles of rough set, attribute reduction, and tabu search. In section 3, the main components of Dynamic Tabu Search Attribute Reduction (DTSAR) are formally presented. In section 4, we report numerical results with DTSAR using some well-known datasets. Finally, the conclusion makes up Section 5.

2. Preliminaries

2.1 *Dimensionality reduction*

One of the important stages in data mining is the attribute reduction stage. There are several techniques to do attribute reduction such as data aggregation, data compression, numerosity reduction and dimensionality reduction. Dimensions refer to the measurement of object perspective. Dimensionality reduction is the methods for decreasing the object dimensions. Reducing the dimension of object also can be referring as reducing the number of attributes. The general objective is to remove irrelevant and redundant data to reduce the computational cost and to avoid data over-fitting (Ng, 1997). Dimensionality reduction will improve the quality of data for efficient data processing task such as pattern recognition and data mining (Dash & Liu, 2008). It is often classified into feature selection or feature extraction. Dash and Liu (2008) stated that in feature selection, a subset of original features are selected in the end while in feature extraction, features are extracted using some mapping from the original set of features.

Dash (2008) defined feature extraction as given a set of feature $S = \{v_1, v_2, \dots, v_D\}$, find a new set of features S' derived from a linear or non-linear mapping of S. The cardinality of $|S'| = d$ and $J(S') \geq J(T)$ for all derived set of features T with $|T| = d$, where J is the evaluation function. 'd' is the parameter set by the user. Feature extraction is when all existing features are recombined to generate new features. Mapping in feature extraction defined as transforming any original 'D' dimensional feature vector to a new 'd' dimensional feature vector.

Feature selection is defines as given a set of feature $S = \{v_1, v_2, \dots, v_D\}$, find a subset S' of S with $|S'| = d$ such that $J(S') \geq J(T)$ for all $T \subset S$, $|T| = d$ where J is the evaluation function, the value 'd' specified by the user. There are four components required by feature selection such as a generation or search strategy, evaluation method, a stopping criterion and/or validation method. From the 'D' original features, the generation of strategy is the process where a selected set of features combination is decided.

2.2 *Rough set theory*

Rough set theory has been proposed by Pawlak in 1982 and is one of the powerful mathematical tool to deal with uncertainty and vagueness of data (Thangavel & Pethalakshmi, 2009). Rough set act as a rough set classifier in data mining to compute a set of reducts which consist of indispensable attribute required for the decision. Main issue in rough set is to find a set of interesting attribute called reduct. It is known that calculation of reducts of an information system is a key problem in Rough Set Theory (Pawlak, 1991; Swiniarski and Skowron, 2003; Jensen and Shen, 2004). Detail of rough set theory can be found in Pawlak (1991).

Finding minimal reduct is NP-hard (Pawlak, 1991). In order to find minimal reduct, it is needed to locate and generate all possible solution reduct and choose the reduct with minimal cardinality. This procedure is only applicable for small datasets as bigger datasets will cause complex calculation of dependency measures and indiscernibility relation. There are many others approach using rough set for attribute reduction which is using the metaheuristic approach such as genetic algorithm (GA), ant colony optimization (ACO), particle swarm optimization (PSO), scatter search and Tabu search.

2.3 *Tabu search*

Fred Glover introduced tabu search in 1986. It is one of the search space technique used to solve the non-linear problem. The word Tabu means something or action that is prohibited or banned due to religions and believes.

Tabu search is a memory-based approach and the most important component in Tabu search is its adaptive memory that allows more searching behaviour. This unavoidable element of Tabu search allows it to increase the memory so that it allows the status of Tabu changes from time to time. Tabu

search is a metaheuristic that guides a local heuristic search procedure to explore the solution space beyond the local optima (Glover and Laguna, 1997).

The basic approach for Tabu search is based on the hill climbing algorithms. Tabu search has a concept of short term memory called Tabu list. Tabu list is used to avoid the searching process trap in local optima. Tabu list is a list that contains the previous solutions located in the search. When a local minimum is found, Tabu search allows non improving moves to be made and disallows any solutions that is already appear in the Tabu list to be sampled again. This mean that the search is not repeated and save a lot of times. Any improvements

to the candidate solution are sure to be new un-sampled solutions.

2.4. *Metaheuristic techniques in attribute reduction*

There are a number of researches done in rough set attribute reduction using metaheuristic techniques. Wang et al. (2009) proposed scatter search for rough set attribute reduction (SSAR). Their SSAR method is according to the structure of the scatter search was discussed earlier which consists of diversification generation procedure, improvement procedure, reference set update, subset generation procedure, solution combination procedure and intensification procedure. The experimental result for SSAR shows that SSAR performed better than other Metaheuristic technique in term of numbers of minimal reducts obtained by each of them. SSAR have also proven that it is much cheaper in computing the dependency degree function γ than other metaheuristic techniques.

According to Jensen and Shen (2003) in their research on finding rough set reduct with ant colony optimization technique (ACO), the authors found out that the proposed AntRSAR performed better in term of time spent to discover reducts than to GenRSAR.

Wang et al., (2005) used particle swam optimization (PSO) to find minimal reducts. To apply PSO to rough set attribute reduction, the authors represent the particle's position as binary bit strings of length N, where N is the total attribute number. Every bit represents an attribute where 1 represent that the attribute is selected and vice versa.

Shi and Fu, (2006) introduced an attribute reduction method based on GA with heuristic information. According to the authors, there are two elements required to generate attribute reducts by GA. The first element is the definition and implementation of the GA. For example, the solution of reduction problem must be represented as a genome/chromosome. Second element is the definition of the objective function or fitness function.

Hedar et al (2006) proposed Tabu search technique to solve the problem for attribute reduction in rough set theory. According to the authors, the proposed Tabu search is a high level Tabu search with long term memory where it will invokes diversification and intensification search scheme besides the Tabu search neighbourhood search methodology. Even though the three mechanism of diversification and intensification scheme do help to achieve a better performance, the drawback of this mechanism would be the challenge to apply these three mechanism in an appropriate time to avoid more unneeded complexity or premature convergence. The result obtained when applying Tabu search to attribute reduction is promising and showed low computation lost in the dependency degree function γ.

3. Tabu Search for Attribute Reduction

In this section, a Tabu search based method called Dynamic Tabu Search Attribute Reduction (DTSAR) is proposed to deal with the attribute reduction problem in Rough Set theory. First, the component of Tabu search for attribute reduction will be described and then the DTSAR algorithm is stated formally.

3.1. *Solution representation*

Tabu search attribute reduction uses binary value to represent the solution that represents the attribute subsets. The trial solution x is represented as a 0-1 vector and number of attribute of the solution x is equal to the number of condition attribute from the original dataset, denoted as $|C|$. If the solution subset have the value "0", it means that the attribute is not contained in the attribute subset, while if the trial solution subset value is "1", means that the attribute is chosen to be inside the attribute subset.

3.2 Solution quality measurement

To measure the solution quality, the dependency degree Υ measurement is used. Assuming that the current dependency degree for current solution attribute D is $\Upsilon x(D)$, and the dependency degree for trial solution for attribute D is $\Upsilon x'(D)$.

Current solution x is better than the trial solution x' if

- $\Upsilon x\ (D) > \Upsilon x'\ (D)$. , where the dependency degree for current solution x is more than the dependency degree for trial solution x'.
  or
- $\sum_i X_i < \sum_i X'_i$ *if* $\Upsilon x\ (D) = \Upsilon x'\ (D)$, where the lesser number of attribute is accepted if both of the dependency degree for both solutions are the same.

The parameter setting for the DTSAR algorithm are as follows:

- |TL|, size of Tabu List, min =1, max = 5.
- Length of neighbourhood solution = $round\,\frac{|C|}{2}$

3.3. *Generating initial solutions*

In Tabu search, initial solution is created before we could apply to the data reduction. The initial solution is constructed randomly, where each value in the attribute subset is assigned a value "1" or "0" randomly. For example, a dataset in Figure 1 have five attributes, denoted as dataset A.
In order to get the initial solution, there are two concept need to follows:

i. First, the attribute of the initial solution must be less than the attribute of the original dataset A, which is attribute for the initial solution < 5.
ii. The dependency degree of original dataset A will calculated and the dependency degree have to be equal with the dependency degree of the initial solution where dependency degree(initial solution) = Dependency Degree (Original dataset).

To generate initial solutions, first, it is needed to calculate the dependency degree of each of the attributes. Pawlak (1991) introduced the formula to calculate the dependency degree measure as follow:

$$Yp(Q) = \frac{|POSp(Q)|}{U} \quad (1)$$

If $Yp(Q) = 1$, we say that Q depend totally on P
If $Yp(Q) < 1$, we say that Q depend partially on P

The attribute for the highest dependency degree will be chosen. The attribute with highest dependency degree will be assigned '1' and included in all related attribute subsets and can't be removed at this stage. Other attributes will be assigned '1' or '0' and changing randomly until the highest dependency value is achieved. The attribute subsets with highest dependency degree value are chosen as initial solutions and denoted as "*x*". The solutions will be used in Tabu search attribute reduction.

3.4. *Neigbourhood Generation*

The generation of neighborhood solution is by randomly selecting one attribute from subset and switches the cell value of this attribute. If the selected attribute cell value is 1, it will automatically change to 0. This means, the selected of attribute will not be in the attribute subset (e.g. remove one attribute). If the selected attribute cell value is 0, it will automatically change to 1. This means, the selected attribute is selected to be in the attribute subset (e.g. add one attribute). Figure 2 depicts the Tabu Search algorithm for generating the neighborhood solution. The number of possible neighborhood solution is dependent on the condition attribute, |*C*| which equivalent to $round\,\frac{|C|}{2}$ (adopt from Hedar et al, 2006). This is to avoid worsen the quality of reduction method when the problem size increases (Hedar et al, 2006). The dependency degree for every generated neighborhood solution *x(i)* is calculated γ *(x(i))* based on rough set dependency degree measurement and the number of attribute for that particular solution *x(i)* also calculated #(*x(i)*). If the dependency of trial solution γ *(x(i))* is more (better) than the dependency degree of the current solution γ *(x)*, then the trial solution is accepted. If the dependency degree for both solutions is the same, then the solution with lesser number of attribute will be accepted.

Patient	Headache	Muscle pain	Temperature	Flu
A1	1	1	0	0
A2	1	1	1	1
A3	1	1	1	1
A4	0	1	0	0
A5	0	0	1	0
A6	0	1	0	1

} attribute

Objects

Fig. 1. Dataset A

Initialization Step:
Generate initial solution randomly, *x*;
Calculate dependency degree for the initial solution *x, γ (x);*
Calculate the number of attribute for the initial solution *x, #(x);*
Set best solution, *xbest* ← *x;*
Set the number of attribute for *xbest, #(xbest)= #(x);*
Set the dependency degree of *xbest, γ (xbest)* ← *γ (x);*
Set *C*= the number of condition attributes in the current datasets
Set maximum number of neighborhood, *Max_n=|C|/2;*
Set maximum Tabu duration, *Max_TT=5;* Tabu list length *TList, =5;*
Set Attribute index, *att_indx=0,* attribute index Tabu tuner, *att_indx_tt=0*
Do while (*dependency degree not equal to one*)
Neighborhood Step
for i=1 to Max_n do
Generate a neighborhood solution which is not in *TList,* by randomly select one attribute *att_indx(i)* from *x* and flip-flop it's cell, *x(i)*
Calculate dependency degree for the neighborhood solution *x(i)*, *γ* (*x(i)*);
Calculate the number of attribute for neighborhood solution *x(i)*, #(*x(i)*);
If (*i* > 1)
if (*γ* (*x*(*i*)) > *γ* (*x(i-1)*))
x← *x*(*i*);
γ (*x*) ← *γ* (*x*(*i*));
#(*x*) ← #(*x*(*i*));
(Update Tabu list)
else if ((*γ* (*x*(*i*)) == *γ* (*x(i-1)*)) and (#(*x*(*i*)) < #(*x(i-1)*)))
x ← *x*(*i*);
γ (*x*) ← *γ* (*x*(*i*));
#(*x*) ← #(*x*(*i*));
(Update Tabu list)
endif
end-for
Update the best solution
if (*γ* (*x*) >= *γ* (*xbest*))
xbest ← *x*;
γ (*xbest*)← *γ* (*x*);
#(*xbest*) ← #(*x*);
end while;
Return the best solution, *xbest* , *γ* (*xbest*) and #(*xbest*);

Fig. 2. Neighborhood Generation Algorithm

3.5. *Tabu list*

The purpose of Tabu list is to avoid generating improper solutions such as all attributes are considered (all zero) and all attributes are not considered (all one) and to avoid revisiting the same solutions (Glover et al., 1997, Hedar et al., 2006). Tabu list plays an important role in search of high quality solutions. The size of Tabu list in this research is according to the research by Hedar et al. (2006) where the min Tabu list is 1 and the maximum Tabu list is 5. Hedar et al. (2006) have run the preliminary experiments before setting the Tabu list size and his experiments showed that the current setting of Tabu list is good enough to escape from local optima. This research will use the same Tabu list size as proposed by Hedar et al. (2006) because the same 13 datasets is used in our Tabu search reduction experiment. The main difference between our Tabu search and Hedar et al. (2006) is that in this research we used short term memory dynamic Tabu list that keep the recency only (only the recent changed attributes). Whilst, Hedar et al. (2006) used static Tabu list to keep track of the recent visited solution. The main point of using dynamic Tabu list is to reduced the computational time and to increase the efficiency by releasing some attribute from Tabu based on the attached Tabu duration which is different from one attribute to another.

Figure 3 show the part of dynamic Tabu list in our proposed algorithm. The size of dynamic Tabu list will keep changing from time to time. Unlike static Tabu list, dynamic Tabu list does not need to wait until all 5 spaces of Tabu list to be fully filled to remove a list. Dynamic Tabu list are more flexible because for each of the Tabu list, an attribute index tabu tuner, *att_indx_tt(i)* of 1 – 5 will be randomly assigned to it. After every iteration, the attribute index tabu tuner, *att_indx_tt(i)* will decrease by – 1. If the attribute index tabu tuner, *att_indx_tt(i)* is 0, the value for the Tabu list is discarded. The benefit of dynamic Tabu list is to skip the aspiration criteria because, if there is an improving solution is found and that solutions is already in the dynamic Tabu list but it is discarded based on the attribute index tabu tuner = 0, it is still possible to accept that improving solution if the same improving solution is re-generated in the next iterations. Furthermore, the usage of dynamic Tabu list can help to shorten the running time of Tabu search.

```
Update Tabu list
   For j=1 to Tlist,
      att_indx_tt(j)= att_indx_tt(j)-1;
     if(att_indx_tt(j))==0     then remove att_indx_tt(j) form Tlist,
  endfor
  Generate random number r between [1, 5];
 Attribute index tabu tuner,  att_indx_tt(i)=r
 Add att_indx(i) and att_indx_tt(i) to Tlist,
Tlist,= Tlist,+1;
```

Fig.3. Dynamic Tabu list algorithm

3.6. *Termination criterion*

Tabu search will terminate the neighbourhood generation when the dependency degree of current solution $\gamma\,(x(i)) = 1$. The generated solution with dependency degree $\gamma\,(x(i)) = 1$ is the reduct/ reduce dataset of the current dataset. The data reduction obtained by using Tabu search attribute reduction is called reduct.

Reduct obtain is then used to obtain classification rules and classification accuracy. The reduct is imported into the rough set analysis tool to produce the classification rules.

3.7. *Development of rough classifier*

The classification rules then are used to generate the accuracy of the classification rules. The accuracy of the classification rules is based on the confusion matrix. Confusion matrix is a useful tool for analyzing how well the classifier can recognize tuples of different classes (Han and Kamber, 2007). In our example for dataset A, we have two classes, where we can say in term of positive tuples (*Flu=*

Yes), and negative tuples (*Flu=No*). True positives refer to the positive tuples that were correctly labeled by the classifier. False positives are the negative tuples that were labeled incorrectly. False negatives are the positive tuples that were incorrectly labeled. Positive is the number of positive (*"Yes"*) tuples, and negative is the number of negative (*"No"*) tuples. The accuracy of the classification rules will be calculate as following,

$$Accuracy = sensitivity \frac{positive}{positive+negative} + specificity \frac{negative}{positive+negative}$$

where, $sensitivity = \frac{true\ positive}{positive}$

and, $specificity = \frac{true\ negative}{negative}$

4. Numerical Experiments

This section explains the analysis of experimental results of the research. This experiment is developed using JAVA programming language. 13 well-known datasets from UCI machine learning, namely m-of-n, exactly, exactly2, heart, vote, credit, mushroom, led, letters, derm, derm2, wq and lung datasets are used in the experiment as shown in Table 1. Two experiments are conducted to compare the performance of the proposed DTSAR. Firstly, DTSAR is compared to other well-know attribute reduction techniques. Then DTSAR is being compared with rough set technique and decision tree. The results of the experiments are explained in the following paragaraph. The Tabu search attribute reduction code was run 20 times for each datasets with different initial solutions. The results of reduct are reported in Table 3. In Table 2, the number of running times for each datasets varies from 0.20 minutes to 22.18 minutes. There are significant different of running time taken to produce reduct for dataset with attributes 20-25. The datasets mushrooms have 22 attributes while LED have 24 attributes. Even thought there is only 2 attributes different between the two datasets, the run time are different by approximately 12.38 minutes. This is due to the DTSAR algorithm trying to filter the best reduct for each dataset until the dependency equal to 1, then only the process terminate. As long as the dependency degree does not equal to 1, the searches continues. These does explain the complexity of data do affect the run times as well. Besides that, looking at dataset letters, the number of attributes is 25 but the run times is 0.07 minutes, this is because in letters dataset, there is only 26 number of objects found in the contents. Same goes to the dataset Lung, even though the number of attributes is the highest, 56, but the run time is only 0.20 because the number of object for the dataset is 32 only.

4.1. *Comparison of reduct using DTSAR with other metaheuristic methods*

Table 5 shows the comparison of results for this current research by using Tabu search which applying dynamic Tabu list with other results that applied metaheuristic technique for attribute reduction. The number of runs to achieve the number is represented as superscripts in parentheses. The number without superscripts means that for all the 20 run times, only that particular number appear. DTSAR outperforms GenRSAR for all the tested data except for heart dataset. DTSAR overall outperformed GenRSAR, SimRSAR, TSAR in the derm2 dataset based on the number of reduct appear more in the run times. DTSAR is better than AntRSAR in small different of 3 datasets, namely LED, Derm2 and WQ dataset, while AntRSAR outperformed DTSAR for Lung dataset. The result for other dataset between AntRSAR and DTSAR is comparable because not much different in terms of result. DTSAR outperforms SimRSAR for Vote, Derm2 and WQ datasets. As for SSAR, DTSAR outperformed SSAR for mushroom, letters, derm2 and wq dataset. As for TSAR, DTSAR outperformed TSAR for letters, Derm, Derm2 and WQ dataset. DTSAR outperformed SSAR for mushroom, letters, derm2 and wq datasets.

As conclusion, DTSAR did outperform other metaheuristic techniques in term of reduct. This proved that applying dynamic Tabu list as part of the Tabu search approach does generate a better yet comparable result with TSAR that using static Tabu list and other metaheuristic approaches.

Table 1. Datasets used in the experiment

Dataset	No. of Attributes	No. Of Objects
M-of-N	13	1000
Exactly	13	1000
Exactly2	13	1000
Heart	13	294
Vote	16	300
Credit	20	1000
Mushroom	22	8124
LED	24	2000
Letters	25	26
Derm	34	366
Derm2	34	358
WQ	38	521
Lung	56	32

Table 2. Running Time obtained from DTSAR

Dataset	No. of attributes	DTSAR	Running time (min)
M-of-N	13	6	0.56
Exactly	13	6	0.54
Exactly2	13	10	1.42
Heart	13	$6^{(15)}7^{(5)}$	0.82
Vote	16	$8^{(19)}9^{(1)}$	0.28
Credit	20	$8^{(10)}9^{(6)}10^{(4)}$	6.65
Mushroom	22	$4^{(16)}5^{(4)}$	9.80
LED	24	$5^{(17)}6^{(3)}$	22.18
Letters	25	$8^{(19)}9^{(1)}$	0.07
Derm	34	$6^{(16)}7^{(4)}$	4.12
Derm2	34	$8^{(4)}9^{(15)}10$	4.45
WQ	38	$12^{(4)}13^{(15)}14$	9.67
Lung	56	$5^{(15)}6^{(5)}$	0.20

4.2. *Comparison of reduct with rough set and decision tree*

With referring to the Table 3, it can be seen that the reduct from three datasets outperformed the rough set technique and decision tree technique, based on credit, mushroom and led dataset. As for rough set, it outperformed DTSAR in five datasets, which is heart, letters, lung, vote and wq datasets. The same number of reduct obtained from DTSAR and rough set technique is exactly, exactly2 and m-of-n dataset. Overall for reduct obtained, DTSAR and decision tree outperformed decision tree technique for all datasets except for letters dataset which

decision tree outperformed DTSAR for that particular dataset. Figure 4 depicted the comparison of reduct obtain from DTSAR and other techniques in graph.

4.3. *Comparison of reduct accuracy with rough set and decision tree*

As we look into classification accuracy value between the three techniques in Table 5, decision tree outperformed the other two techniques by five datasets namely Credit, Derm, Derm2, Letters and Vote. The classification accuracy is very useful because from the accuracy obtain, it is possible to know if certain reduct is reliable or not reliable. For example, the derm dataset have the best reduct in rough set theory, but the accuracy for derm dataset is very low if compared to DTSAR and Decision Tree which is 53.49%. This means that the reduct on by Rough set is not reliable. This condition is the same for Letters, Heart, Derm2, and exactly2 datasets. These four dataset produce a better reduct in rough set but the classification accuracy obtained is lower than other methods. As for letters dataset, the accuracy of reduct from rough set is 0%. This means that using rough set to measured the reduct for dataset is not reliable. Referring to Table 3, the number of reducts obtain for letters is 2 while reduct for WQ is 1, but both produced different value in accuracy. The reduct for WQ is 1 but the accuracy of reduct is 99.05%. The accuracy may differ in term of how well the rough set classifier can recognize tuples for the datasets (Han & Kamber 2007). Tuples for the datasets can recognize as true positive, false positive, true negative and false negative. All the tuples are defines based on the four classes mentioned, are dependent on how well the rough set classifier can recognize whether the tuples are correctly or incorrectly labelled in tested datasets. In the case of letters, with accuracy equal to 0%, which means that the reduct produced are mostly wrong labelled by the classifier hence giving a smaller yet 0% to the accuracy. Figure 4 depicted the comparison on the three techniques in accuracy

Table 3. Result of DTSAR with other methods

Dataset	DTSAR No. Of Reduct	Rough Set	Decision Tree
Credit	$8^{(10)}9^{(6)}10^{(4)}$	15	18
Derm	$6^{(16)}7^{(4)}$	4	19
Derm2	$8^{(4)}9^{(15)}10$	5	18
Exactly	6	6	13
Exactly2	10	10	11
Heart	$6^{(15)}7^{(5)}$	3	10
Letters	$8^{(19)}9^{(1)}$	2	3
Lung	$5^{(15)}6^{(5)}$	4	14
m-of-n	6	6	13
Mushroom	$4^{(16)}5^{(4)}$	6	20
Led	$5^{(17)}6^{(3)}$	6	21
Vote	$8^{(19)}9^{(1)}$	4	12
WQ	$12^{(4)}13^{(15)}14$	1	21

Table 4. Comparison of techniques based on accuracy

Dataset	DTSAR Accuracy (%)	Rough Set	Decision Tree
Credit	81.80	77	89.7
Derm	92.07	53.49	100
Derm2	88	67	100
Exactly	100	100	94.2
Exactly2	100	63.50	91.83
Heart	96.25	65.28	93.22
Letters	88.46	0	100
Lung	96.87	100	100
m-of-n	100	100	100
Mushroom	94.87	100	98.80
Led	100	100	100
Vote	88.79	78.51	89.80
WQ	97	99.05	92.45

Table 5. Result of DTSAR with other methods

Dataset	No. of attributes	AntRSAR	SimRSAR	GenRSAR	TSAR	SSAR	DTSAR
M-of-N	13	6	6	$6^{(6)}7^{(12)}$	6	6	6
Exactly	13	6	6	$6^{(10)}7^{(10)}$	6	6	6
Exactly2	13	10	10	$10^{(9)}11^{(11)}$	10	10	10
Heart	13	$6^{(18)}7^{(2)}$	$6^{(29)}7^{(1)}$	$6^{(18)}7^{(2)}$	6	6	$6^{(15)}7^{(5)}$
Vote	16	8	$8^{(15)}9^{(15)}$	$8^{(2)}9^{(18)}$	8	8	$8^{(19)}9^{(1)}$
Credit	20	$8^{(12)}9^{(4)}10^{(4)}$	$8^{(18)}9^{(1)}11^{(1)}$	$10^{(6)}11^{(14)}$	$8^{(13)}9^{(5)}\ 10^{(2)}$	$8^{(9)}9^{(8)}10^{(3)}$	$8^{(10)}9^{(6)}10^{(4)}$
Mushroom	22	4	4	$5^{(1)}6^{(5)}7^{(14)}$	$4^{(17)}5^{(3)}$	$4^{(12)}5^{(8)}$	$4^{(16)}5^{(4)}$
LED	24	$5^{(12)}6^{(4)}7^{(3)}$	5	$6^{(1)}7^{(3)}8^{(16)}$	5	5	$5^{(17)}6^{(3)}$
Letters	25	8	8	$8^{(8)}9^{(12)}$	$8^{(17)}9^{(3)}$	$8^{(5)}9^{(15)}$	$8^{(19)}9^{(1)}$
Derm	34	$6^{(17)}7^{(3)}$	$6^{(12)}7^{(8)}$	$10^{(6}11^{(14)}$	$6^{(14)}7^{(6)}$	6	$6^{(16)}7^{(4)}$
Derm2	34	$8^{(3)}9^{(17)}$	$8^{(3)}9^{(7)}$	$10^{(4)}11^{(16)}$	$8^{(2)}9^{(14)}10^{(4)}$	$8^{(2)}9^{(18)}$	$8^{(4)}9^{(15)}10$
WQ	38	$12^{(2)}13^{(7)}14^{(11)}$	$13^{(16)}14^{(4)}$	16	$12^{(1)}13^{(13)}14^{(6)}$	$13^{(4)}14^{(16)}$	$12^{(4)}13^{(15)}14$
Lung	56	4	$4^{(7)}5^{(12)}\ 6^{(1)}$	$6^{(8)}7^{(12)}$	$4^{(6)}5^{(13)}6^{(1)}$	4	$5^{(15)}6^{(5)}$

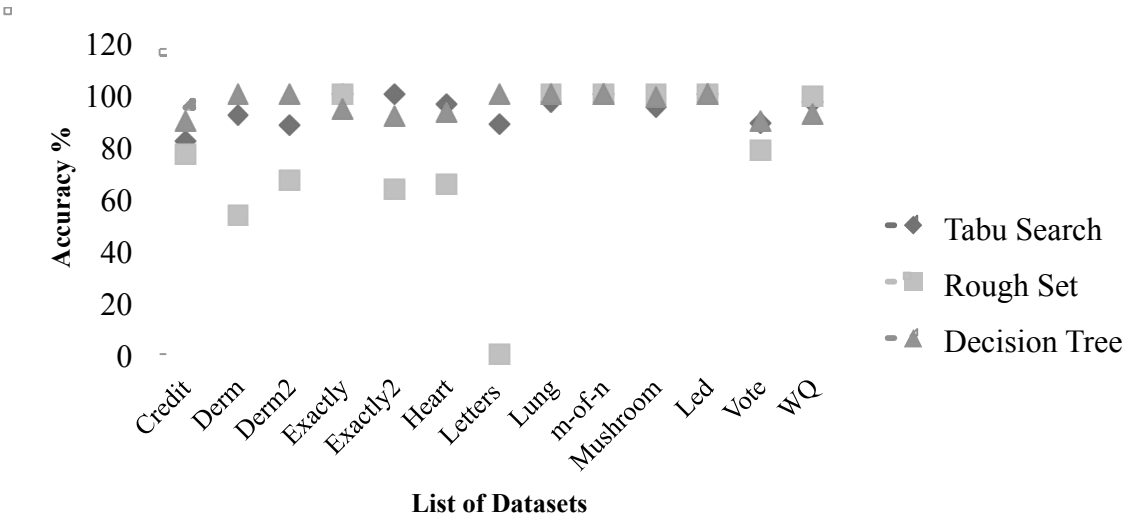

Fig. 4. Comparison of techniques based on accuracy

Table 6. Comparison of techniques based on number of rules

Dataset	DTSAR	Rough Set	Decision Tree
	No. Of Rules		
Credit	340	879	179
Derm	174	16	13
Derm2	186	18	18
Exactly	64	64	91
Exactly2	648	576	27
Heart	231	20	17
Letters	23	117	1
Lung	23	2417	7
m-of-n	64	64	69
Mushroom	163	2046	169
Led	11	6209	19
Vote	125	14	15
WQ	254	3	5

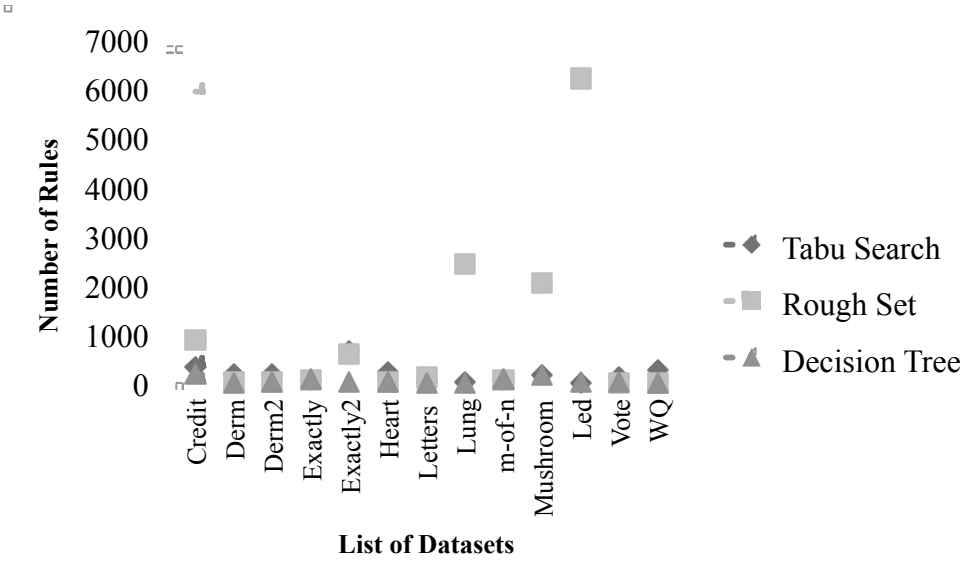

Fig. 5. Comparison of techniques based on number of rules

4.4. *Comparison of reduct classification rules with rough set and decision tree*

From the reduct obtained, rules are generated. From Table 6, it is clear that although the result for reduct is worst for decision tree technique but decision tree have the least rules generation for all datasets. There are some significant difference between DTSAR and rough set technique. Even though rough sets do outperformed DTSAR in generating reduct for the five datasets, the rule generated by one of the dataset is too huge until it reaches 2417 rules for lung dataset. DTSAR and rough set technique produce the same reduct for dataset exactly, exactly2 and m-of-n dataset as in Table 6, the rules generated by DTSAR are more than the rules generated by rough set. Overall, the rules generated by rough set are too many and is too complex to analyze if compared to DTSAR and decision tree. Figure 5 depicted the comparison of number of rules generated from reduct obtain from DTSAR and other techniques in graph.

Overall, among the three techniques, it can be summarized that DTSAR and rough set is comparable in term of the reduct obtained and the classification accuracy except for wq dataset, the rough set technique outperformed DTSAR in reduct very significantly.

5. Conclusion

This section summarized the works done in this research. The research is based on Tabu search approach towards attribute reduction using rough set theory. The approach introduced in this paper is the dynamic Tabu list and has been applied to the rough set attribute reduction. There are 13 well-known datasets being used in the research. In this research reducts obtained represent the data reduction for the dataset, the dimension of the dataset is reduced into smaller size. The results of the research show that dynamic Tabu list that is introduce to the Tabu search attribute reduction does give a promising result to the reduct.

The dynamic Tabu list is use to skip the aspiration criteria and to promote faster running times. However, the result obtained is comparable with previous research on Tabu search technique with static Tabu list. If the result is significant and better, it could be use to hybrid with other Metaheuristic for future attribute reduction purposes.

References:

[1] Dash, M. & Liu, H. 2008. Dimensionality Reduction. http://www.public.asu.edu/~huanliu/papers/dm07.pdf [12 April 2009].

[2] Glover, F. & Laguna, M. 1997. Tabu search. Boston: Kluwer Academic Publishers.

[3] Han, J., Kamber, M. & Chiang, J. 2006. Data Mining : Concepts and Techniques. San Francisco: Morgan Kauffman Publishers.

[4] Hedar, A. R., Wang, J. & Fukushima, M., 2006. Tabu search for attribute reduction in rough set theory. Soft Computing - A Fusion of Foundations, Methodologies and Applications 12: 909-918.

[5] Jensen, R., Shen, Q. 2004. Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches. IEEE Transactions on Knowledge and Data Engineering 16(12): 1457-1471.

[6] Jensen, R., Shen, Q. 2003. Finding rough set reducts with ant colony optimization. In: Proceeding of 2003 UK Workshop Computational Intelligence: 15-22.

[7] Pawlak, Z. 1991. Rough Sets: Theoretical Aspects of Reasoning About Data. Boston: Kluwer Academic Publishers.

[8] Pawlak, Z., Grzymala-Busse, J., Slowinski, R., & Ziarko, W. 1995. Rough sets. Communications of the ACM 38(11): 88-95.

[9] Swiniarski, R.W. and Skowron, A. 2003. Rough set methods in feature selection and recognition. Pattern Recognition Letters 24(6): 833-849.

[10] Thangavel, K. & Pethalakshmi, A. 2009. Dimensionality reduction based on rough set theory: A review. Applied Soft Computing 9(1): 1-12.

[11] Wang, J., Hedar, A.R., Zheng, G., & Wang, S. 2009. Scatter search for rough set attribute reduction. International Joint Conference on Computational Sciences and Optimization 1: 531-535.