Int. J. Appl. Math. Comput. Sci., 2011, Vol. 21, No. 1, 57–68
DOI: 10.2478/v10006-011-0004-3

amcs

APPLICATION OF AGENT–BASED SIMULATED ANNEALING AND TABU SEARCH PROCEDURES TO SOLVING THE DATA REDUCTION PROBLEM

IRENEUSZ CZARNOWSKI, PIOTR JĘDRZEJOWICZ

Department of Information Systems
Gdynia Maritime University, Morska 83, 81–225 Gdynia, Poland
e-mail: {irek,pj}@am.gdynia.pl

The problem considered concerns data reduction for machine learning. Data reduction aims at deciding which features and instances from the training set should be retained for further use during the learning process. Data reduction results in increased capabilities and generalization properties of the learning model and a shorter time of the learning process. It can also help in scaling up to large data sources. The paper proposes an agent-based data reduction approach with the learning process executed by a team of agents (A-Team). Several A-Team architectures with agents executing the simulated annealing and tabu search procedures are proposed and investigated. The paper includes a detailed description of the proposed approach and discusses the results of a validating experiment.

Keywords: data reduction, machine learning, A-Team, optimization, multi-agent system.

1. Introduction

Learning from examples remains the most important paradigm of machine learning. It is understood as the process of finding a model (or a function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class labels are unknown (Talukdar *et al*., 1996). The process of finding a classification model is also called the learning process or the learning classifier and the final model is, in short, called the classifier. Training classifiers is also considered a basic task in many application areas, including, for example, data mining, pattern recognition and computer vision.

Research in the field of machine learning resulted in the development of numerous approaches and algorithms for identifying classes (clustering), discovering associations and predicting new, unknown objects (classification). The last one, i.e., the classification problem, is considered here. One of the recent focuses of such research includes methods of selecting relevant information. Data reduction techniques are approaches to lower the quantity of information. In practice, this means that these approaches are concerned with selecting informative instances and, finally, with producing a minimal set of instances or prototypes to represent a training set and presenting the reduced dataset to a machine learning algorithm (Wilson and Martinez, 2000a). It is obvious that removing some instances from the training set reduces the time and memory complexity of the learning process (Hart, 1968). Data reduction performed without losing extractable information is also considered an approach to increase the effectiveness of the learning process when the available datasets are large, such as those encountered in data mining, text categorization, financial forecasting, the mining of multimedia databases and meteorological, financial, industrial or science repositories (Kim and Oommen, 2003).

Selecting relevant data is also one of approaches to data mining, when data are stored in separated and physically distributed repositories. Moving all of the data to a central site and building a single global learning model may not be feasible due to restricted communication bandwidth among sites or high expenses involved. The selection of relevant data in distributed locations and then moving only the local patterns can eliminate or reduce the above restrictions and speed up the distributed learning process (Czarnowski, 2010).

Data reduction is often called editing, condensing, filtering, etc, and that depends on the object of reduction. Data reduction can be achieved by selection of instances, or attributes/features, or by simultaneous reduction in both

dimensions (Bhanu and Peng, 2000). Since instance selection and feature selection are known to belong to the NP-hard problem class (Hamo and Markovitch, 2005; Kohavi and John, 1997), none of the approaches proposed so far can be considered superior nor guaranteeing satisfactory results in terms of learning error reduction or increased efficiency of the learning process. Hence, searching for robust and efficient approaches to data reduction is still a lively field of research. This paper focuses on data reduction through simultaneous instance and feature selection.

The main contribution of the paper is proposing and evaluating through a computational experiment an approach to data reduction with the learning process executed by a team of agents, and discussing its alternative architectures. The investigated architectures are based on various combinations of agents working within an A-Team and executing the simulated annealing and tabu search procedures.

The goal of the paper is to establish experimentally how the cooperation of agents within different team architectures can influence the learning performance of the classifier developed using the obtained reduced dataset. The proposed agent-based approach is compared with that where the data reduction problem is solved using standard simulated annealing and tabu search algorithms. The performance of the proposed algorithms is evaluated experimentally using several UCI datasets (Asuncion and Newman, 2007). The obtained results are also compared with some state-of-the-art approaches to data reduction.

The paper is organized as follows. Section 2 contains the formulation of the data reduction problem and briefly overviews data reduction algorithms. Section 3 gives some details on simulated annealing, tabu search, and agent-based population learning algorithms. Section 4 explains main features of the proposed implementation of the agent-based data reduction algorithm. Section 5 provides details on the computational experiment setup and discusses its results. Finally, the last section contains conclusions and suggestions for future research.

2. Data reduction

2.1. Problem formulation. The problem of learning from data can be formulated as follows. Given a dataset D, a set of hypothesis H, a performance criterion P, the learning algorithm L outputs a hypothesis $h \in H$. The data D consist of N training examples, also called instances. Each example is described by a set A of n features.

The goal of learning is to find L producing $h \in H$ to optimize the performance criterion P. In the pattern classification application, h is a classifier that has been induced based on the set D.

The learning process can be reinforced by data preprocessing of which data reduction is an important part. The data reduction process aims at finding patterns or regularities within certain features, allowing inducing the so-called prototypes or reference vectors. The ultimate goal of data reduction, also called prototype reduction (Nanni and Lumini, 2009), is to reduce the size of the training dataset without losing extractable information. A lot of research work confirms that data reduction through instance or feature or instance and feature selection can play a pivotal role in building robust learning models (Bhanu and Peng, 2000; Dash and Liu, 1997; Hart, 1968; Kirkpatrick *et al.*, 1983; Kuncheva and Bezdek, 1998; Wilson and Martinez, 2000a).

It can be stipulated that data reduction should be considered the problem of finding an optimal subset of prototypes. In the formal manner, the problem of data reduction can be formulated as follows. Given a learning algorithm L, and a dataset D with features described by a feature set A, the optimal prototype dataset, S_{opt}, is a subset of the dataset D, where each example is described by a set of $A' \subset A$, such that the performance criterion of the learning algorithm L is maximized. Commonly used performance criteria include the accuracy of classification, the complexity of the hypothesis, classification cost or classification error.

With respect to learning from data, when a data reduction process is carried out, the task of the learner L is to output the hypothesis $h \in H$ that optimizes the performance criterion P using the dataset S which is a subset of the set D, such that $|S| < |D|$ (ideally $S = S_{opt}$), where each example $x \in S$ is described by a set A' of m features, where $m < n$.

2.2. Data reduction algorithms. Data reduction algorithms can be divided into two categories: prototype selection and prototype extraction. Prototype selection is a technique of choosing an optimal set of reference vectors from the original set, whereas prototype extraction means constructing an entirely new set of instances smaller, with respect to its dimensionality, than the original dataset (Bezdek and Kuncheva, 2001). This paper deals with prototype selection.

Data reduction can be carried out in the feature or instance space or in both of these spaces simultaneously. Instance selection methods can be classified on the basis of several different criteria. Raman (2003) points out that instance selection methods can be grouped into three classes: filter methods, wrapper methods, and embedded methods. Filtering is based on random search, random sampling or genetic algorithms. In this case, selected instances are tested to find out whether they come from the same distribution as the original entry dataset and whether the current reduced dataset is sufficient to produce the desired classifier. Wrapper methods include boosting, where the distribution of data is modified (Raman, 2003). The windowing technique (Quinlan, 1993) belongs also to wrapper methods. Lazy learners, like the k nearest neighbors

method, belong to embedded methods. In this class of methods the instance selection strategy is integrated with their learning scheme (Raman, 2003).

Wilson and Martinez (2000a) suggested that instance selection methods can be categorized into incremental search, decremental search and batch search. Incremental search begins with an empty set of prototypes and adds each instance to the reduced set if it fulfills some criterion (example approaches include CNN (Hart, 1968), the IB family (Aha *et al*., 1999) or SNN (Ritter *et al*., 1975)). Decremental search begins with $S = D$ and successfully removes instances from S (example approaches include ENN (Wilson and Martinez, 2000b) or the DROP family (Wilson and Martinez, 2000b)). Finally, in the batch search mode, instances fulfilling the removal criteria are removed at one go.

Bezdek and Kuncheva (2001) group instance selection into three categories: condensation, error-edition, and search methods. The object of condensation methods is to find a consistent reference set. The original condensation method is CNN. Error-edition methods rely on the assumption that points from different classes that are close to apparent boundaries between them contain "noise" and should be removed from the original dataset. This category includes, for example, the DROP algorithm family. The last category includes search methods that can achieve the same goal by criterion-driven combinational optimization. This group includes local search heuristics, tabu search, and metaheuristics, like for example, particle swarm optimization, simulated annealing, genetic algorithms or evolutionary algorithms (e.g., Kuncheva and Bezdek, 1998; Nanni and Lumini, 2009; Skalak, 1994; Song and Lee, 1996).

The feature selection approaches can be grouped into two main classes: filter and wrapper methods. The former choose the best attribute set optimizing some assumed evaluation function, like, for example, distance, information value, dependency or consistency (Dash and Liu, 1997). A distinctive feature of filter methods is that they remove irrelevant features before the data are presented to the learning algorithm (Raman, 2003). Wrapper methods use the performance of the learning algorithm as the evaluation function and the classifier error rate is considered as the feature selection criterion. Other approaches attempt to evaluate, rank and select some subsets of features generated by heuristic, random or genetic search. A detailed discussion of feature selection approaches can be found in the works of Dash and Liu (1997), Kohavi and John (1997), Raman (2003) or Zongker and Jain (1996).

In the case of simultaneous reduction in both discussed dimensions some sophisticated heuristics or random search techniques are used. An example of the heuristic approach to data reduction can be found in the research by Raman (2003), who combines the SCRAP feature selection algorithm and LASER instance selection. The former is a sequential search filter and the latter an embedded instance method. It is also shown that such combination can yield maximum improvement to prediction accuracy of the learner compared with the case when only one dimension is reduced. Skalak (1994) proposes a random mutation hill climbing algorithm for selecting the most accurate of a prototype set. The algorithm uses a representation that records both the instances and features in a single binary string. The length of the string is adequate to the number of bits required to represent the selected prototypes and to the number of features. Within such a representation the i-th bit records whether to use the corresponding instance or feature. The search mechanism is based on the fact that the mutation of the bit vector changes the selection of the instances or features. At each iteration of the algorithm only one bit is muted. The idea of data reduction by the genetic algorithm is discussed by Rozsypal and Kubat (2003). In this approach each solution is described by two chromosomes, one representing the selected instances and the other representing the selected features. Each chromosome consists of integers which point to selected instances or features. Such encoding is more flexible in a domain with many instances as opposed to a binary chromosome representation proposed by Kuncheva and Jain (1999).

Another approach to data reduction is proposed in the work of Czarnowski and Jędrzejowicz (2010), where data reduction in both of the discussed dimensions is integrated with the learning process. Similar ideas are investigated by Aksela (2007), Bhanu and Peng (2000), Bull (2004) or Sahel *et al*. (2007). Integration of data reduction with the learning process may require introduction of some adaptation mechanisms as exemplified by the idea of learning classifier systems (Bull, 2004). Unfortunately, integrating the data reduction and classifier construction stages leads to a considerable enlargement of the decision space and hence the complexity of the problem. One possible way of dealing with such complexity was proposed by Czarnowski and Jędrzejowicz (2010). Considering that feature and instance selection are themselves computationally difficult, obtaining effective solutions in the case of the discussed integrative approaches requires application of some sophisticated metaheuristics and local search random techniques executed in parallel computational environments.

3. Simulated annealing, tabu search and an agent-based population learning algorithm for data reduction

3.1. Simulated annealing. The idea of using simulated annealing (SA) was proposed by Kirkpatrick *et al.* (1983). SA is a local search technique simulating the physical process of "annealing". In contrast to other metaheuristics as, for example, population based techniques, in the SA a sin-

gle solution is maintained. SA starts with a random solution and next, at each step during neighbourhood search, a move is made. Moves are controlled by some probability function. The acceptance of a downhill move depends on the magnitude of the decrease in the objective function value and the search time. The pseudocode of the basic SA algorithm is shown as Algorithm 1. More details of this algorithm and its applications can be found in the work of Aarts and van Laarhoven (1987).

Algorithm 1
Simulated annealing algorithm

1. Set an initial temperature T_0 and a low temperature T_L;
2. Set the number of iterations I;
3. Set the control temperature $T_i := T_0$;
4. Set $i := 0$;
5. Form an initial solution s;
6. Calculate the fitness of $s : f(s)$;
7. While $(i < I)$
8. Execute a move producing a new solution s';
9. Calculate the fitness of s': $f(s')$;
10. If (s' is better than s) then $s := s'$;
11. Else
12. Generate a random number d in the interval $(0, 1)$;
13. If $(d < \exp(\mod (f(s) - f(s'))/T_i))$ then $s := s'$;
14. End if
15. Update the control temperature $T_i := f(T_0, T_L, i)$;
16. $i := i + 1$;
17. End while.

3.2. Tabu search. Tabu search (TS) is a heuristic local search introduced by Glover (1989; 1990). It is an interesting alternative to randomized methods and enhances the performance of the local search method. TS operates on one solution called the current one. It starts with a random solution or a solution generated by some other technique implemented with a view of producing the initial solution. Next, at each step, the TS algorithm performs a move which replaces the current solution with its neighbor from a neighborhood. A move is defined by a neighborhood function that with each solution associates a set of neighbor solutions. A move can take place only if it is not on the tabu list. The tabu list is a short memory structure used to prevent cycling and to escape from local optima. Each executed move is placed on the tabu list for a number of iterations called the tabu tenure. Finally, a new solution is reached from the current solution. This is repeated over and over until some stopping criterion is satisfied. TS has been successfully applied to a variety of difficult combinatorial optimization problems (e.g., Glover and Laguna, 1993). The pseudocode of the basic tabu search algorithm is shown as Algorithm 2. More details of this algorithm and its applications can be found in the work of Glover and Laguna (1993).

Algorithm 2
Tabu search algorithm

1. Set $i := 0$;
2. Set the number of iterations I;
3. Initiate the list of tabu active moves consisting of moves which are not on tabu list $T_M := \emptyset$;
4. Set the tabu tenure t;
5. Form an initial solution s;
6. While $(i < I)$
7. Execute a move producing a new solution s';
8. If (s' is better than s) then $s := s'$;
9. End if
10. Update T_M and remove the moves older than t from the tabu list;
11. $i := i + 1$;
12. End while.

3.3. Agent-based population learning algorithm. The A-Team architecture has served as a problem-solving paradigm used to design the proposed population learning algorithm. The A-Team concept was originally introduced by Talukdar *et al*. (1996). The concept of the A-Team was motivated by several approaches like blackboard systems and evolutionary algorithms, which proved to be able to successfully solve some difficult combinatorial optimization problems. Within an A-Team, agents achieve an implicit cooperation by sharing a population of solutions to the problem to be solved. Following evolutionary algorithms, such solutions form a population consisting of individuals.

An A-Team can also be defined as a set of agents and a set of memories, forming a network in which every agent remains in a closed loop. Each agent possesses some problem-solving skills and each memory contains a population of temporary solutions to the problem at hand. This also means that such an architecture can deal with several searches conducted in parallel. In each iteration of the process of searching for the best solution, agents cooperate to construct, find and improve solutions which are read from the shared, common memory. All agents can work asynchronously and in parallel.

The main functionality of the agent-based population learning approach includes organizing and conducting the process of searching for the best solution. It involves a sequence of the following steps:

- Generation of the initial population of solutions to be stored in the common memory.
- Activation of optimizing agents which apply solution improvement algorithms to solutions drawn from the

common memory and store them back after the attempted improvement applying some user defined replacement strategy.

- Continuation of the reading–improving–replacing cycle until a stopping criterion is met. Such a criterion can be defined either or both as a predefined number of iterations or a limiting time period during which optimizing agents do not manage to improve the current best solution. After computation has been stopped, the best solution achieved so far is accepted as the final one.

More information on the population learning algorithm with optimization procedures implemented as agents within an asynchronous team of agents (A-Team) can be found in the work of Barbucha *et al*. (2009), where also several A-Team implementations are described.

4. A-Team with simulated annealing and tabu-search optimizing agents

Data reduction and learning from examples are computationally difficult combinatorial optimization problems (see, e.g., Hamo and Markovitch, 2005; Kohavi and John, 1997). The proposed approach, integrating features such as heuristics like evolutionary computation (Michalewicz, 1996), local search algorithms (Glover and Laguna, 1993), or the population learning algorithm (Jędrzejowicz, 1999), has the ability to solve such a combinatorial optimization problem. The paper proposes several A-Team architectures in which optimizing agents execute the simulated annealing and tabu search procedures with a view to solve instances of the data reduction problem. The approach is named $ABRA$ (**A**gent-**B**ased Data **R**eduction **A**lgorithm). To implement it, one has to set and define the following:

- solution representation format,
- initial population of individuals,
- fitness function,
- neighborhood search procedure,
- replacement strategy implemented for managing the population of individuals.

4.1. Solution representation format. In the discussed case the shared common memory is used to store a population of solutions to the data reduction problem. A potential solution is represented by a string consisting of two parts. Its first part represents numbers of the selected reference instances. The second part includes numbers of the selected features.

4.2. Initial population. The initial population of solutions is generated randomly. When it is generated, the instances are grouped into clusters using an instance grouping procedure. The instance grouping procedure uses the values of the similarity coefficient computed as in the work of Czarnowski and Jędrzejowicz (2004) to identify clusters of instances.

To show the instance grouping procedure in a formal manner, the following notation has to be introduced. Let x denote a training example, N the number of instances in the set D and n the number of features. The total length of each instance (i.e., the training example) is equal to $n+1$, where the element numbered $n+1$ contains the class label. The class label of each example can take any value from a finite set of decision classes $C = \{c_l : l = 1, \ldots, k\}$, which has cardinality k. Also, let $X = \{x_{ij} : i = 1, \ldots, N; j = 1, \ldots, n+1\}$ denote the matrix of $n+1$ columns and N rows containing values of all instances from D. The detailed pseudo-code of the procedure producing clusters of instances is shown as Algorithm 3.

Algorithm 3
Instance grouping procedure

Input: X, the matrix containing the values of all instances from D.
Output: clusters from which prototypes can be selected.
1. Transform data instances: each $\{x_{ij}\}$ for $i = 1, \ldots, N$ and $j = 1, \ldots, n$ is normalized into interval $[0, 1]$ and then rounded to the nearest integer, that is, 0 or 1;
2. Determine

$$s_j = \sum_{i=1}^{N} x_{ij}, \tag{1}$$

where $j = 1, \ldots, n$.
3. For instances from X, belonging to the class c_l (where $l = 1, \ldots, k$), compute the value of its similarity coefficient I_i:

$$\forall_{x:x_{i,n+1}=c_l}\ I_i = \sum_{j=1}^{n} x_{ij} s_j, \tag{2}$$

where $i = 1, \ldots, N$.
4. Map input vectors from X with the same value of the similarity coefficient I_i into clusters.
5. Let $Y_1, \ldots, Y_t$ denote the obtained clusters such that $D = \bigcup_{i=1}^{t} Y_i$ and $\forall_{i \neq j : i,j=1,\ldots,t} \quad Y_i \cap Y_j = \emptyset$.

Random selections of instances from such clusters produced by Algorithm 3 constitute the main step of the first stage of generating the initial population of individuals. The number of selected instances is equal to the number of initialized clusters, which means that in a feasible solution each cluster is represented by a single instance from the training set.

The second part of a string representing the solution to the data reduction problem and containing numbers of the selected features is, at this stage, generated randomly. To assure the required diversity, the initial solutions are drawn in such a way that random individuals represent different numbers of features. The discussed part of the string has a length of minimum 1 and maximum n.

4.3. Fitness function. Each solution from the population is evaluated and the value of its fitness is computed. The fitness function is the predictive accuracy on the dataset, which is constructed taking into account the instances and features as indicated by the solution, using the learning algorithm.

4.4. Optimizing agents. In the proposed approach there are two kinds of optimizing agents operating on individual solutions. These are SA and TS agents. Each optimizing agent tries to improve the quality of the obtained solutions by applying the respective SA or TS algorithm, aiming at finding a better solution.

SA optimizing agents use the following three moves to explore the neighborhood of the solution:

- Instance replacement procedure (IRP): the move is defined as replacing a randomly selected instance from a randomly chosen cluster with some other randomly chosen instance thus far not included within the solution. The pseudocode of the IRP is shown as Algorithm 4.

Algorithm 4
Instance replacement procedure

Input: s, individual representing a solution; L, list of numbers of instances not in s; t, number of clusters in s.
Output: s', modified solution.
1. Set k by drawing it at random from $\{1, 2, \ldots, t\}$;
2. Identify r which is an instance number representing the k-th cluster of s;
3. Set r' by drawing it at random from L;
4. Replace the instance numbered r by the instance numbered r' within the k-th cluster of s thus producing individual s'.

- Feature replacement procedure (FRP): the move is defined as replacing a randomly selected feature with some other random feature thus far not included within the solution. The pseudocode of the FRP is shown as Algorithm 5.

Algorithm 5
Feature replacement procedure

Input: s, individual representing a solution; M, list of numbers of features in s; M', list of numbers of features not in s.
Output: s', modified solution.
1. Set f by drawing it at random from M;
2. Set f' by drawing it at random from M';
3. Replace f in s by f' thus producing s'.

- Feature adding/removing procedure (FARP): the move is defined as adding a new feature so far not included within the solution or removing a randomly selected feature from the solution. Operations of adding or removing a feature are equally likely. The pseudocode of the FARP is shown as Algorithm 6.

Algorithm 6
Feature adding/removing procedure

Input: s, individual representing a solution; M, list of the numbers of features in s; M', list of the numbers of features not in s.
Output: s', modified solution.
1. Set h by drawing it at random from the set $\{0, 1\}$;
2. If $(h == 0)$ then
3. Set f' by drawing it at random from M';
4. Add f' to s producing s';
5. Else
6. Set f by drawing it at random from M;
7. Remove f from s producing s';
8. End if.

The TS optimizing agents use the following two moves to explore the neighborhood of the solution:

- Instance exchange with the tabu list (IETL): the move is defined as replacing a randomly selected instance from a randomly chosen cluster with some other randomly chosen instance providing the exchange is tabu active, and not on the tabu list. The pseudocode of the IETL is shown as Algorithm 7.

Algorithm 7
Instance exchange with tabu list

Input: s, individual representing the current solution; L, list of numbers of instances not in s; t, number of clusters in s; T_M, list of tabu active moves.
Output: s', modified solution.
1. Set k by drawing it at random from $\{1, 2, \ldots, t\}$;
2. Identify r which is an instance number representing the k-th cluster of s;
3. Set r' by drawing it at random from L;
4. If $(r' \notin T_M)$ then
5. Replace the instance numbered r by the instance numbered r' within the k-th cluster of s thus producing individual s';
6. Add r to T_M;
7. End if.

- Feature selection with the tabu list (FSTL): the move is defined as replacing, adding or removing a randomly selected feature with some other random feature thus far not included within the current solution providing the operation is tabu active and not on the tabu list. The pseudocode of the FSTL is shown as Algorithm 8.

Algorithm 8
Feature selection with tabu list

Input: s, individual representing the current solution; M, list of numbers of features in s; M', list of numbers of features not in s; T_M, list of tabu active moves.
Output: s', modified solution.
1. Set h by drawing it at random from
the set $\{0, 1, 2\}$;
2. If $(h == 0)$ then
3. Set f' by drawing it at random from M';
4. If $(f' \notin T_M)$ then
5. Add f' to s producing individual s';
6. End if;
7. End if;
8. If $(h == 1)$ then
9. Set f by drawing it at random from M;
10. Remove f from s producing individual s';
11. Add f to T_M;
12. End if;
13. If $(h == 2)$ then
14. Set f' by drawing it at random from M';
15. If $(f' \notin T_M)$ then
16. Set f by drawing it at random from M;
17. Replace f in s by f' thus producing
individual s';
18. Add f to T_M;
19. End if;
20. End if.

4.5. Replacement strategy. The proposed A-Team uses a simple replacement strategy. Each optimizing agent obtains a solution drawn at random from the population of solutions (individuals). The solution returned by the optimizing agent is merged with the current population replacing the current worst solution.

5. Computational experiment

5.1. Computational experiment settings. To validate the proposed approach, we decided to carry out a computational experiment. It aimed at answering the following questions:

- Does *ABRA* perform better than the standard simulated annealing approach?
- Does *ABRA* perform better than the standard tabu search approach?
- Does the choice of the A-Team architecture influence the performance of the classification model?

In the reported research, the following A-Team configurations have been considered:

- *ABRA #1*: one optimizing agent running the simulated annealing algorithm using one of the neighborhood search moves randomly drawn from the three implemented procedures, that is the IRP, the FRP or the FARP.
- *ABRA #2*: two optimizing agents—one running the simulated annealing algorithm using the IRP neighborhood search procedure and the other running the simulated annealing algorithm using one of the neighborhood search moves randomly drawn from the two implemented procedures, that is the FRP or the FARP.
- *ABRA #3*: three optimizing agents, each running the simulated annealing algorithm using the implementation of the IRP, the FRP and the FARP, respectively.
- *ABRA #4*: one optimizing agent running the tabu search algorithm using one of the neighborhood functions randomly drawn from the two implemented procedures, that is, IETL or FSTL.
- *ABRA #5*: two optimizing agents—one running the tabu search algorithm with the IETL move and the second running the tabu search algorithm with the FSTL move.
- *ABRA #6*: two optimizing agents—one running the simulated annealing algorithm using one of the neighborhood search moves randomly drawn from the three implemented procedures, that is, the IRP, the FRP or the FARP and the other running the tabu search algorithm using one of the neighborhood functions randomly drawn from the two implemented procedures, that is, IETL or FSTL.
- *ABRA #7*: two optimizing agents—one running the simulated annealing algorithm with the IRP move and the other running the tabu search algorithm with the FSTL move.
- *ABRA #8*: two optimizing agents—one running the simulated annealing algorithm with the neighborhood search moves, that is, the FRP or the FARP, the other running the tabu search algorithm with the IETL move.

In each of the above cases, random drawing (if applicable) takes place at each iteration, that is, each time a solution to be improved has been forwarded to an optimizing agent.

5.2. Dataset choice and the experiment plan. Datasets used in the experiment were obtained from the UCI Machine Learning Repository (Asuncion and Newman, 2007). They include Cleveland heart disease (303 instances, 13 attributes, 2 classes, the best reported classification accuracy 90% (DataSet, 2009)), credit approval (690, 15, 2, 86.9% (DataSet, 2009)), Wisconsin breast cancer (699, 15, 2, 97.5% (Asuncion and Newman, 2007)), and the sonar problem (208, 60, 2, 87.1% (DataSet, 2009)).

Each benchmarking problem was solved 30 times, and the experiment plan involved three repetitions of the 10-cross-validation scheme. The reported values of the quality measure were averaged over all runs. The quality measure in all cases was the correct classification ratio. In the 10-cross-validation scheme, for each fold, the training dataset was reduced using the proposed approach.

During the experiment the population size for each investigated A-Team architecture was set to 20. The process of searching for the best solution was stopped either after 100 iterations or earlier in case there was no improvement of the current best solution for one minute of computation. The values of these parameters were set arbitrarily.

In the case of the simulated annealing algorithm, the initial temperature T_0 and the low temperature T_L were set following the suggestions of Aarts and van Laarhoven (1987). The control temperature was calculated as $T_{i+1} = T_i/(1 + \alpha T_i)$, where i is the iteration number, $\alpha = (T_0 - T_L)/(IT_0T_L)$, and I is the number of SA iterations.

The tabu tenure for instance selection procedures was set to 15, 30, 35, 10 iterations, respectively, for heart, credit, cancer and sonar problems. These values were calculated as about 5% of the number of instances in the original, non-reduced dataset. In the case of the feature selection, the tabu tenure was set arbitrarily to 3 for heart, credit and cancer, respectively, and to 10 for the sonar problem.

The proposed A-Teams were implemented using the middleware environment called JABAT (Barbucha *et al.*, 2009), based on the JAVA code and built using JADE (Java agent development framework) (Bellifemine *et al.*, 2003).

5.3. Experiment results. The classification accuracy of the classifier obtained using the proposed approach (i.e., using the set of prototypes, found by selecting instances and removing irrelevant attributes) was compared with

- results obtained by classifier learning without data reduction, i.e., using the full, non-reduced dataset;
- results obtained using the reduced dataset produced by the standard simulated annealing algorithm (SA);
- results obtained using the reduced dataset produced by the standard tabu search algorithm (TS).

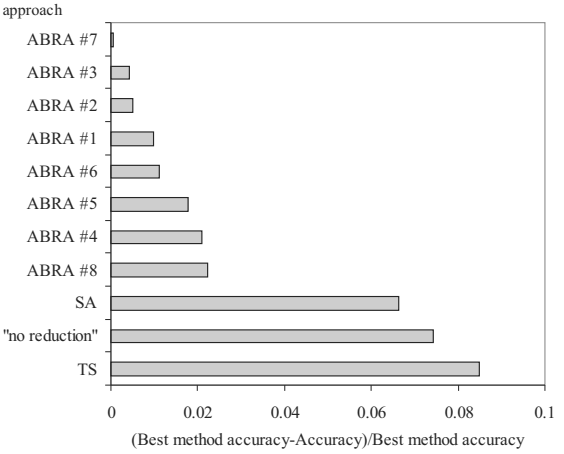

Fig. 1. Ranking of data reduction methods.

The generalization accuracy was used as the performance criterion. The classification tool applied was the C 4.5 algorithm (Quinlan, 1993). The experiment results are shown in Table 1. The ranking of the compared approaches is shown in Fig. 1, where the horizontal axis represents the mean relative difference between the mean accuracies of the best method and the given method.

It should be noted that the proposed algorithm produces very good results as compared with the case when data reduction is carried out by the standard SA algorithm or by the TS algorithm. A similar observation holds true when the classifier is induced using the original, non-reduced dataset.

From Table 1 it is also clear that the choice of the A-Team architecture understood as a particular combination of agents in *ABRA* can have an impact on the classification accuracy. Results of the experiment indicate that the *ABRA #7* configuration, where the instance and feature selection is carried out in parallel and where the instance selection is carried out by the simulated annealing algorithm and feature selection by the tabu search algorithm, assures significantly better results than other architectures. Moreover, it can be observed that *ABRA* architectures based on the simulated annealing technique (*ABRA #1*, *ABRA #2* and *ABRA #3*) outperform these where optimizing agents run the tabu search algorithm (*ABRA #4* and *ABRA #5*). Comparing *ABRA #7* and *ABRA #8*, it can also be noted, that better results can be obtained using the simulated annealing algorithm for instance selection and the tabu search algorithm for feature selection than applying the architecture with the interchanged algorithm assignment.

To confirm the above findings, the non-parametric Friedman test (Friedman, 1937) to check whether particular data reduction algorithms are equally effective independently of the kind of problem being solved, was applied. The test is based on weights (points) assigned to the data reduction algorithms used in the experiment. To

Table 1. Accuracy of the classification results (%).

Problem	heart	cancer	credit	sonar
C 4.5: non-reduced dataset	77.89	94.57	84.93	74.04
ABRA #1	86.9	96.41	88.14	82.44
ABRA #2	87.49	96.53	88.38	83.06
ABRA #3	87.69	96.47	88.22	**83.35**
ABRA #4	85.56	96.43	84.72	83.12
ABRA #5	87.43	95.21	85.94	82.32
ABRA #6	86.54	**96.61**	87.31	82.87
ABRA #7	**88.31**	96.42	**89.01**	**83.35**
ABRA #8	85.32	94.50	86.88	82.52
SA	82.00	93.05	83.22	75.68
TS	78.17	93.13	82.60	73.64

assign weights, a 11-point scale was used with 11 points for the best and 1 point for the worst algorithm. The test aimed at deciding among the following hypotheses:

- H_0: zero hypothesis—data reduction approaches are statistically equally effective regardless of the kind of the problem being solved,
- H_1: alternative hypothesis—not all data reduction approaches are equally effective.

The analysis was carried out at the significance level of 0.5. The respective value of the χ^2 statistics with 11 architectures and 4 instances of the considered problems is 138 and the value of the respective χ^2 quantile is equal to 18.31 for 10 degrees of freedom. Thus, it can be observed that not all approaches are equally effective regardless of the kind of the problems being solved. In Fig. 2 average weights for each data reduction approach are shown.

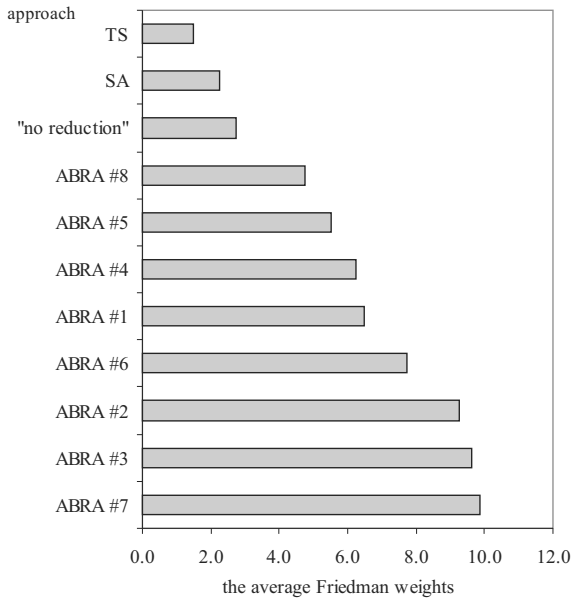

Fig. 2. Friedman test weights for the compared approaches.

The *ABSA #7* algorithm is also competitive in comparison with other classifiers and other approaches to data reduction, which can be concluded from data shown in Table 2.

Furthermore, the experiment results showed that the proposed approaches are also quite effective when a compression of the data is considered. While the average number of retained instances is similar in the case of *ABRA*, standard SA and standard TS, the number of retained features is, on average, smaller when using the agent-based simulated annealing approach. This can be concluded from the data shown in Table 3. The best performer from the point of view of the data compression rate is clearly *ABRA #7*, achieving the best result in terms of decreasing the complexity of knowledge representation as well as reducing the required computation time.

6. Conclusions

The main contribution of the paper was proposing and validating a novel approach to solving a data reduction problem. The proposed approach uses the multi-agent paradigm and enables solving a difficult data reduction optimization problem using a set of optimizing agents executing the simulated annealing and tabu search procedures.

The proposed approach extends the available range of algorithms for solving the data reduction problem. Computational experiment results confirmed that the agent-based data reduction algorithm using the simulated annealing and tabu search procedures is an effective data reduction tool contributing to achieving higher quality of machine learning classification in comparison with standard simulated annealing and tabu search implementations. It was also shown that parallel reduction of both instances and features assures better overall results as compared with other schemes considered. Additionally, the computational experiment results confirmed that simulated annealing procedures outperform tabu search ones when the instance selection is carried out.

Properties of the proposed approach in terms of its

Table 2. Comparison of different classifiers and data reduction approaches (I: instance selection only, F: feature selection only, IF: instance and feature selection, N: no data reduction).

Classifier	Reduction type	Accuracy (%)			
		heart	cancer	credit	sonar
ABRA #6	IF	86.54	**96.61**	87.31	82.87
ABRA #7	IF	**88.31**	96.42	**89.01**	**83.35**
k-NN (Wilson and Martinez, 2000b)	N	81.19	96.28	84.78	58.8
k-NN + RELIEF (Raman, 2003)	F	77.85	72.12	79.57	-
CNN (Wilson and Martinez, 2000b)	I	73.95	95.71	77.68	74.12
SNN (Wilson and Martinez, 2000b)	I	76.25	93.85	81.31	79.81
IB3 + RELIEF (Raman, 2003)	IF	79.94	73.25	71.75	-
RMHC (Sahel *et al*., 2007)	IF	82.3	70.9	-	-
GA-KJ (Rozsypal and Kubat, 2003)	IF	74.7	95.5	-	55.3

Table 3. Average number of retained instances and features.

Approach	heart	cancer	credit	sonar
	average number of retained features			
ABRA #1	10.1	8.1	10.4	39
ABRA #2	9.8	7.1	9.8	38.7
ABRA #3	9.2	7.6	9.6	32.2
ABRA #4	10.4	8.1	10.2	37.3
ABRA #5	9.9	7.4	9.8	37.0
ABRA #6	9.5	8.0	9.5	36.1
ABRA #7	9.1	7.8	9.2	34.1
ABRA #8	10.2	9.1	9.4	36.5
SA	9.86	9.4	10.6	42.2
TS	9.2	8.6	10.1	36.7
	average number of the retained instances			
all algorithms	162	133	184	94

efficiency and computation time will be further studied. Future research will also focus on examining the influence of the number of the copies of optimizing agents on the quality of the selected prototypes and the computation speed-up factor.

Acknowledgment

This research has been supported by the Polish Ministry of Science and Higher Education within a grant for the years 2008–2010 and a grant for the years 2010–2013.

References

Aarts, E. and van Laarhoven, P. (1987). Simulated anealing: A pedestrian review of the theory and some applications, *in* J. Kittler and P.A. Devijver (Eds.), *Pattern Recognition and Applications*, Springer-Verlag, Berlin.

Aha, D., Kibler, D. and Albert, M. (1999). Instance-based learning algorithms, *Machine Learning* **6**(1): 37–66.

Aksela, M. (2007). *Adaptive Combinations of Classifiers with Application to On-line Handwritten Character Recognition*, Ph.D. thesis, Helsinki University of Technology, Helsinki.

Asuncion, A. and Newman, D. (2007). Website of the UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml/`.

Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak-Ropel, E. and Wierzbowska, I. (2009). e-JABAT—An implementation of the Web-based A-Team, *in* N. Nguyen and L. Jain (Eds.), *Intelligent Agents in the Evolution of Web and Applications*, Studies in Computational Intelligence, Vol. 167, Springer-Verlag, Berlin/Heidelberg, pp. 57–86.

Bellifemine, F., Caire, G., Poggi A. and Rimassa, G. (2003). Jade. A white paper, *Technical Report Exp, 3(3)*, September 2003, pp. 6–19, `http://exp.telecomitalialab.com`.

Bezdek, J. and Kuncheva, L. (2001). Nearest prototype classifier designs: An experimental study, *International Journal of Intelligent Systems* **16**(2): 1445–1473.

Bhanu, B. and Peng, J. (2000). Adaptive integration image segmentation and object recognition, *IEEE Transactions on Systems, Man and Cybernetics* **30**(4): 427–44.

Bull, L. (2004). Learning classifier systems: A brief introduction, *in* L. Bull (Ed.), *Applications of Learning Classifier Systems*, Studies in Fuzziness and Soft Computing, Springer, Berlin, pp. 3–14.

Czarnowski, I. (2010). Prototype selection algorithms for distributed learning, *Pattern Recognition* **43**(6): 2292–2300.

Czarnowski, I. and Jędrzejowicz, P. (2004). An approach to instance reduction in supervised learning, *in* F. Coenen, A. Preece and A. Macintosh (Eds.), *Research and Development in Intelligent Systems XX*, Springer, London, pp. 267–282.

Czarnowski, I. and Jędrzejowicz, P. (2010). An approach to data reduction and integrated machine classification, *New Generation Computing* **28**(1): 21–40.

Dash, M. and Liu, H. (1997). Feature selection for classification, *Intelligence Data Analysis* **1**(3): 131–156.

DataSet, C. (2009). Website of the Datasets Used for Classification: Comparison of Results, `http://www.is.umk.pl/projects/datasets.html`.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *Journal of the American Statistical Association* **32**(200): 675–701.

Glover, F. (1989). Tabu search, Part I, *ORSA Journal on Computing* **1**(3): 533–549.

Glover, F. (1990). Tabu search, Part II, *ORSA Journal on Computing* **2**(1): 4–32.

Glover, F. and Laguna, M. (1993). Tabu search, *in* C. Reeves (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley and Sons, New Jork, NY, pp. 70–150 .

Hamo, Y. and Markovitch, S. (2005). The COMPSET algorithm for subset selection, *Proceedings of the 19th International Joint Conference for Artificial Intelligence, Edinburgh, UK*, pp. 728–733.

Hart, P. (1968). The condensed nearest neighbour rule, *IEEE Transactions on Information Theory* **14**: 515–516.

Jędrzejowicz, P. (1999). Social learning algorithm as a tool for solving some difficult scheduling problems, *Foundation of Computing and Decision Sciences* **24**(2): 51–66.

Kim, S.-W. and Oommen, B. (2003). A brief taxonomy and ranking of creative prototype reduction schemes, *Pattern Analysis Application* **6**(3): 232–244.

Kirkpatrick, S., Gelatt, C. and Vecci, M. (1983). Optimisation by simulated annealing, *Science* **220**(4598): 671–680.

Kohavi, R. and John, G. (1997). Wrappers for feature subset selection, *Artificial Intelligence* **97**(1–2): 273–324.

Kuncheva, L. and Bezdek, J. (1998). Nearest prototype classification: Clustering, genetic algorithm or random search?, *IEEE Transactions on Systems, Man and Cybernetics* **28**(1): 160–164.

Kuncheva, L. and Jain, L. (1999). Nearest-neighbor classifier: Simultaneous editing and feature selection, *Pattern Recognition Letters* **20**(11–13): 1149–1156.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin.

Nanni, L. and Lumini, A. (2009). Particle swarm optimization for prototype reduction, *Neurocomputing* **72**(4–6): 1092–1097.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA.

Raman, B. (2003). Enhancing learning using feature and example selection, *Technical report*, Texas AM University, College Station, TX.

Ritter, G., Woodruff, H., Lowry, S. and Isenhour, T. (1975). An algorithm for a selective nearest decision rule, *IEEE Transactions on Information Theory* **21**(6): 665–669.

Rozsypal, A. and Kubat, M. (2003). Selecting representative examples and attributes by a genetic algorithm, *Intelligent Data Analysis* **7**(4): 291–304.

Sahel, Z., Bouchachia, A., Gabrys, B. and Rogers, P. (2007). Adaptive mechanisms for classification problems with drifting data, *in* B. Apolloni, R. Howlett and L. Jain (Eds.), *KES 2007*, Lecture Notes in Artificial Intelligence, Vol. 4693, Springer-Verlag, Berlin/Heidelberg, pp. 419–426.

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithm, *Proceedings of the International Conference on Machine Learning, New Brunswick, NJ, USA*, pp. 293–301.

Song, H. and Lee, S. (1996). LVQ combined with simulated annealing for optimal design of large set reference models, *Neural Networks* **9**(2): 329–336.

Talukdar, S., Baerentzen, L., Gove, A. and de Souza, P. (1996). Asynchronous teams: Co-operation schemes for autonomous, computer-based agents, *Technical Report TR EDRC 18-59-96*, Carnegie Mellon University, Pittsburgh, PA.

Wilson, D. and Martinez, T. (2000a). An integrated instance-based learning algorithm, *Computational Intelligence* **16**(1): 1–28.

Wilson, D. and Martinez, T. (2000b). Reduction techniques for instance-based learning algorithm, *Machine Learning* **33**(3): 257–286.

Zongker, D. and Jain, A. (1996). Algorithm for feature selection: An evaluation, *Proceedings of the International Conference on Pattern Recognition, ICPR96, Vienna, Austria*, pp. 18–22.

Ireneusz Czarnowski holds a B.Sc. and an M.Sc. in electronics and communication systems from Gdynia Maritime University. In 2004 he obtained a Ph.D. in computer science from Poznań Technical University. Presently he is an assistant professor at the Department of Information Systems, Faculty of Business Administration, Gdynia Maritime University. His research interests cover combinatorial optimization, artificial intelligence, data mining and Internet technologies.

Piotr Jędrzejowicz received his M.A., Ph.D. and D.Sc. degrees in operations research from Gdańsk University. Currently he is a professor of information systems and the head of the Information Systems Department, Faculty of Business Administration, Gdynia Maritime University. He is also the vice-rector for research of the same university. His research interests include decision support systems, computational intelligence, data mining and multi-agent systems.

Received: 13 April 2010
Revised: 15 November 2010