An Agent-Based Approach to the Multiple-Objective Selection of Reference Vectors

Ireneusz Czarnowski and Piotr Jędrzejowicz

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

Abstract. The paper proposes an agent-based approach to the multiple-objective selection of reference vectors from original datasets. Effective and dependable selection procedures are of vital importance to machine learning and data mining. The suggested approach is based on the multiple agent paradigm. The authors propose using JABAT middleware as a tool and the original instance reduction procedure as a method for selecting reference vectors under multiple objectives. The paper contains a brief introduction to the multiple objective optimization, followed by the formulation of the multiple-objective, agent-based, reference vectors selection optimization problem. Further sections of the paper provide details on the proposed algorithm generating a non-dominated (or Pareto-optimal) set of reference vector sets. To validate the approach the computational experiment has been planned and carried out. Presentation and discussion of experiment results conclude the paper.

1 Introduction

As it has been observed in [9], in supervised learning, a machine-learning algorithm is shown a training set, which is a collection of training examples called instances. After learning from the training set, the learning algorithm is presented with additional input vectors, and the algorithm must generalize, that is to decide what the output value should be.

It is well known that in order to avoid excessive storage and time complexity and to improve generalization accuracy by avoiding noise and overfitting, it is often advisable to reduce original training set by removing some instances before learning phase or to modify the instances using a new representation.

Instances reduction, often referred to as a selection of reference vectors, becomes especially important in case of large data sets, since overcoming storage and complexity constraints might become computationally very expensive. Although a variety of instance reduction methods has been so far proposed in the literature (see, for example the review [9]), no single approach can be considered as superior nor guaranteeing satisfactory results and a reduction of the learning error or increased efficiency of the supervised learning. Therefore, the problem of selecting the reference instances remains an interesting field of research.

One of the most important application areas of the machine learning methods and tools is data mining understood as the extraction of implicit, previously unknown, and potentially useful information from data. Unfortunately, several useful machine learning tools and techniques as for example neural networks, support vector machines or statistical methods do not provide explanations on how they solve problems. In some application areas like medicine or safety assurance this may cause some doubts or even lower the trust of the users. In such cases users may prefer approaches where the process of knowledge extraction from data is easier comprehensible by human beings. An obvious approach would be using methods leading to the extraction of some logical rules representing the knowledge about phenomenon at hand. Extracting precise, reliable, useful and easy to comprehend rules from datasets is not a trivial task [10][13].

Most widely used techniques for the rules generation, such as, for example, algorithms C4.5 and CART [14][15], are based on decision trees. However in case of the large datasets the resulting decision tree might become very complex making it difficult to understand and evaluate by the human being. Possible way to overcome the problem is to select a set of reference vectors as an input to the decision tree generating algorithm producing than, so called, prototype-based model [10]. It is expected that instance reduction through selection of reference vectors may bring about several benefits including increased quality of generalization, easier to comprehend set of rules, decreased requirements for storage and computational resources and increased simplicity of the extracted knowledge.

Selecting reference vectors is inherently a multiple-objective problem. The resulting set should be evaluated not only in terms of generalization (classification) quality of the prototype model, but also in terms of the resulting number of rules, their complexity, data compression level, computational time required etc. Considering the above, in this paper the selection of reference vectors is seen as a multi-objective optimization problem which solution is a non-dominated (or Pareto-optimal) set of reference vector sets. To obtain solutions to such problems an agent-based approach is suggested.

The paper proposes the multiple-objective agent-based optimization of reference vectors selection algorithm, implemented using the JABAT environment. JABAT is a middleware supporting the construction of the dedicated A-Team architectures that can be used for solving a variety of computationally hard optimization problems [3].

The paper is organized as follows. Section 2 reviews briefly a general multiple-objective, optimization problem. Section 3 of the paper contains formulation of the multiple-objective, agent-based, reference vectors selection optimization problem. Section 4 provides details on the proposed algorithm generating a non-dominated (or Pareto-optimal) set of reference vector sets. To validate the approach the computational experiment has been planned and carried out. Its results are presented and discussed in Section 5. Finally, in the last section some conclusions are drawn and directions for future research are suggested.

2 Multiple-Objective Optimization

The general multiple-objective optimization problem is formulated following [11] as:

$$max\{z_1, \ldots, z_J\} = max\{f_1(x), \ldots, f_J(x)\} \tag{1}$$

or

$$min\{z_1, \ldots, z_J\} = min\{f_1(x), \ldots, f_J(x)\} \tag{2}$$

where $x \in D$ and solution $x = [x_1, \ldots, x_l]$ is a vector of decision variables, D is the set of feasible solutions and z is a vector of objective functions z_j, $j = 1, \ldots, J$. The type of the variables may describe different classes of problems. When the variables are discrete the multiple-objective optimization problem is called as multiple-objective combinational optimization problem.

The image of a solution x in the objective space is a point $z^* = [z_1^*, \ldots, z_J^*]$, where $z_j^* = f(x_j)$, $j = 1, \ldots, J$.

Point z dominates z', if, for the maximization case, $z_j \geq z'_j$ (for each j) and $z_j > z'_j$ for at least one j, and vice versa for the minimization problem.

A solution $x \in D$ is Pareto-optimal, if there is no $x' \in D$ that dominates x. A point being an image of Pareto-optimal solution is called non-dominated. The set of all Pareto-optimal solutions is called the Pareto-optimal set. The image of the Pareto-optimal set in objective space is called the non-dominated set.

An approximation of the non-dominated set is a set A of feasible points such that $\neg\exists z_1, z_2 \in A$ such that z_1 dominated z_2.

Weighted linear scalarizing functions are defined as:

$$s_l(z, \Lambda) = \sum_{j=1}^{J} \lambda_j z_j, \tag{3}$$

where $\Lambda = [l_1, \ldots, l_J]$ is a weight vector such that $\lambda_j \geq 0$ and $\sum_{j=1}^{J} \lambda_j = 1$.

Others scalarizing functions are based on calculation of distances between z_j and z_j^0, where z^0 is a references point. The weighted Tchebycheff scalarizing function may serve as an example of such a function. It is defined as follows:

$$s_\infty(z, z^0, \Lambda) = \max_j\{\lambda_j(z_j^0 - z_j)\}. \tag{4}$$

Further details in respect to the multiple-objective optimization can be found, for example, in [11].

3 Multiple-Objective Selection of Reference Instances

Instance reduction problem concerns removing a number of instances from the original training set T and thus producing the reduced training set S. Let N denote the number of instances in T and n-the number of attributes. Total length of each instance (i.e. training example) is equal to $n + 1$, where element numbered $n+1$ contains the output value. Let also $X = \{x_{ij}\}$ ($i = 1,\ldots,N$, $j = 1,\ldots,n+1$) denote a matrix of $n+1$ columns and N rows containing values of all instances from T.

Usually, instance reduction algorithms are based on distance calculation between instances in the training set. In such a case selected instances, which are situated close to the center of clusters of similar instances, serve as the reference instances. The approach requires using some clustering algorithms. Other methods, known as similarity-based methods, remove k nearest neighbors from a given category based on an assumption that all instances from the neighbor will be, after all, correctly classified. The third group of methods eliminates training examples based on an evaluation using some removal criteria [9] [12].

In this paper instance reduction (or reference vector selection) is seen as a multiple-objective optimization problem. It can be solved by producing a set of Pareto-optimal solution instances each being a non-dominated set of reference vectors. The following criteria are used to evaluate reference vectors:

- Classification quality - f_1
- Data compression level - f_2
- Number of rules - f_3
- Length of rules -f_4

It is clear that the above set of criteria represents a situation with several conflicting goals. Selection of the preferred reference vector from the set of Pareto-optimal ones is left to the user. Hence, solving an instance reduction problem is seen as generating a set of non-dominated solutions each, in turn, representing a set of the selected reference vectors.

4 Agent-Based Algorithm for Generating Pareto-Optimal Sets of Reference Vectors

4.1 Instance Reduction Algorithm

It is proposed to base instance reduction on the idea of Instance Reduction Algorithm (IRA) proposed in the earlier paper of the authors [12]. The IRA was originally proposed as a tool for solving a single objective version of instance reduction problem. It was shown in [12] that the approach can result in reducing the number of instances and still preserving a quality of the data mining results. It has been also demonstrated that in some cases reducing the training set size can increase efficiency of the supervised learning. The proposed algorithm is based on calculating, for each instance from the original set, the value of its similarity coefficient, and then grouping instances into clusters consisting of instances with identical values of this coefficient, selecting the representation of instances for each cluster and removing the remaining instances, thus producing the reduced training set. The algorithm involves the following steps:

Stage 1. Transform X normalizing value of each x_{ij} into interval $[0,1]$ and then rounding it to the nearest integer, that is 0 or 1.

Stage 2. Calculate for each instance from the original training set the value of its similarity coefficient I_i:

$$I_i = \sum_{j=1}^{n+1} x_{ij} s_j, i = 1, \ldots, N, \tag{5}$$

where:

$$s_j = \sum_{i=1}^{N} x_{ij}, j = 1, \ldots, n+1. \tag{6}$$

Stage 3. Map input vectors (i.e. rows from X) into t clusters denoted as Y_v, $v = 1, \ldots, t$. Each cluster contains input vectors with identical value of the similarity coefficient I and t is a number of different values of I.

Stage 4. Select input vectors to be retained in each cluster. Let $|Y_v|$ denote a number of input vectors in cluster v. Then the following rules for selecting input vectors are applied:

- If $|Y_v| = 1$ then $S = S \cup Y_v$.
- If $|Y_v| > 1$ then $S = S \cup \left\{x_j^v\right\}$, where x_j^v are reference instances from the cluster Y_v selected by applying the JABAT and where the number of selected instances corresponds to multi objective optimization problem.

4.2 Overview of the JABAT

The single objective instance reduction is a combinatorial and computationally difficult problem [12]. Its multiple-objective version can not be computationally easier. To deal with the multiple-objective instance reduction it is proposed to use the population-based approach with optimization procedures implemented as an asynchronous team of agents (A-Team), originally introduced by Talukdar [2]. An A-Team is a collection of software agents that cooperate to solve a problem by dynamically evolving a population of solutions. An A-Team usually uses combination of approaches inspired by natural phenomena including, for example, insect societies [4], evolutionary processes [5] or particle swarm optimization [7], as well as local search techniques like, for example, tabu search [6].

An A-Tam is a cyclic network of autonomous agents and shared, common memories. Each agent contains some problems solving skills and each memory contains a population of temporary solutions to the problem to be solved. All the agents can work asynchronously and parallel. During their works agents cooperate by selecting and modifying these solutions. In the reported approach the A-Team was designed and implemented using JADE-based A-Team (JABAT) environment.

JABAT is a middleware allowing to design and implement an A-Team architecture for solving combinatorial optimization problems. The main features of JABAT include:

- The system can in parallel solve instances of several different problems.

- A user, having a list of all algorithms implemented for given problem may choose how many and which of them should be used.
- The optimization process can be performed on many computers. The user can easily adjoin or delete a computer from the system. In both cases JABAT will adapt to the changes, commanding the agents working within the system to migrate.

The JABAT produces solutions to combinatorial optimization problems using a set of optimising agents, each representing an improvement algorithm. To escape getting trapped into a local optimum an initial population of solutions called individuals is generated or constructed. Individuals forming an initial population are, at the following computation stages, improved by independently acting agents, thus increasing chances for reaching a global optimum.

Main functionality of the proposed environment is searching for the optimum solution of a given problem instance through employing a variety of the solution improvement algorithms. The search involves a sequence of the following steps:

- Generation of an initial population of solutions.
- Application of solution improvement algorithms which draw individuals from the common memory and store them back after attempted improvement, using some user defined replacement strategy.
- Continuation of the reading-improving-replacing cycle until a stopping criterion is met.

The above functionality is realized by the two main types of classes. The first one includes *OptiAgents*, which are implementations of the improvement algorithms. The second are *SolutionManagers*, which are agents responsible for maintenance and updating of individuals in the common memory. All agents act in parallel. Each *OptiAgent* is representing a single improvement algorithm (for example simulated annealing, tabu search, genetic algorithm, local search heuristics etc.). An *OptiAgent* has two basic behaviors defined. The first is sending around messages on readiness for action including the required number of individuals (solutions). The second is activated upon receiving a message from some *SolutionManager* containing the problem instance description and the required number of individuals. This behaviour involves improving fitness of individuals and resending the improved ones to a sender. A *SolutionManager* is brought to life for each problem instance. Its behaviour involves sending individuals to *OptiAgents* and updating the common memory.

Main assumption behind the proposed approach is its independence from a problem definition and solution algorithms. Hence, main classes *Task* and *Solution* upon which agents act, have been defined at a rather general level. Interfaces of both classes include function *ontology()*, which returns JADE's ontology designed for classes *Task* and *Solution*, respectively. Ontology in JADE is a class enabling definition of the vocabulary and semantics for the content of message exchange between agents. More precisely, an ontology defines how the class is transformed into the text message exchanged between agents and how the text message is used to construct the class (here either *Task* or *Solution*).

4.3 Implementation of the Multiple-Objective Instance Reduction Algorithm

The JABAT environment has served as the tool for solving instances of the multiple-objective instance reduction problem. All the required classes have been defined in the package called MORIS (*multiple-objective reference instances selection*). The MORIS includes the following classes: *MORIS_Task* inheriting form the *Task* class, *MORIS_Solution* inheriting from the *Solution* class. The *MORIS_Task* identifies data set and creates the clusters of potential reference instances. *MORIS_Solution* contains representation of the solution. It consists of the list of the selected references instances from original data set and the values of the cost factors corresponding respectively to the classification accuracy, the percentage of compression of the training set and the number of rules. To obtain values of these factors the C 4.5 classification tool is used. For each decision tree produced by the C 4.5 the size of rules is additionally calculated and recorded.

To communication between optimization agents and the solution manager the *MORIS_TaskOntology* and *MORIS_SolutionOntology* classes have been also defined through over-ridding the *TaskOntology* and *SolutionOntology*, respectively. The *TaskOntology* is needed to enable sending between agents and the common memory task parameters and instance numbers belonging to respective clusters and representing potential reference instances. The *SolutionOntology* is needed to enable sending around potential solutions.

Each optimization agent operates on one individual (solution) provided and randomly selected form the population by the *SolutionManager*. Its role is to improve quality of the solution. After the stopping criterion has been met, each agent resends individuals to the *SolutionManager*, which, in turn, updates common memory by replacing randomly selected individual with the improved ones. Generally, the *SolutionManager* manages the population of solutions, which on initial phase is generated randomly. The generation of an initial population of solutions is designed to obtain a population consisting of solutions with different number of reference instances in each clusters. The *SolutionManager*, after adding to the population a solution received from the *OptiAgent*, overwrides and updates the set of potentially Pareto-optimal solutions.

To solve the discussed multiple objective problem two types of agents representing different improvement procedures have been implemented. In each case the agent's classes are inherited from the *OptiAgent* class. Both procedures aim at improving current solution through modification and exchange of the reference vectors in different clusters. After having received a solution to be improved an optimization agent generates random vector of weights Λ. It is used to obtain the normalized function $s(z, \Lambda)$, which, in turn, is used to evaluate potential solutions.

The first optimization agent - local search with tabu list (in short: RLS), modifies the current solution by removing the randomly selected reference vector from the randomly chosen cluster and replacing it with some other randomly chosen reference vector thus far not included within the improved solution. The modification takes place providing the vector to be replaced is not on the tabu list. After the modification the newly added reference vector is placed on the tabu list and remains there for a given number of iterations. This number depends on the cluster size and decreases for smaller clusters. The modified solution replaces the current one if it is evaluated as a better one using the current normalized function $s(z, \Lambda)$.

The second optimization agent - incremental/decremental local serach (in short: IDLS), modifies the current solution either by removing the randomly selected reference vector from the randomly chosen cluster or by adding some other randomly chosen reference vector thus far not included within the improved solution. Increasing or decreasing a number of reference vectors within clusters is a random move executed with equal probabilities. Pseudo-codes showing both types of the discussed optimization agents are shown in Example 1 and 2.

Example 1: Pseudo code of the RLS type optimization agent

```
public class RandomLocalSearch extends OptiAgent {
 public void improveSolution() {
  Initiate the list of tabu moves;
  Draw at random a weight vector L;
  MORIS_Solution x = (MORIS_Solution)solution.clone();
  /*where x is the solution that has been sent to optimize*/
  do{
     Select randomly cluster from x;
     Select randomly n, where n corresponds to instance number
     from selected cluster;
     If (n is not on the list of tabu active moves){
       Select randomly n', where n' corresponds to instance
       number which is not represented within x;
       Remove n from x and add n' to x producing x';
       Calculate fitness of the x' on s(z,L);
       if(x' is better on s(z,L) then x) x=x';
       Add n to the list of tabu moves and during next s
       iterations do not change this instance number;
     }
     Update the list of tabu moves;
   }while (!terminatingCondition);
   /*solution is ready to be sent back*/
   solution = x;}
}
```

Example 2: Pseudo code of the IDLS type optimization agent

```
public class IncDecLocalSearch extends OptiAgent {
 public void improveSolution() {
  Draw at random a weight vector L;
  Set s as a parameter determining decremental/incremental phase;
  MORIS_Solution x = (MORIS_Solution)solution.clone();
  /*where x is the solution that has been sent to optimize*/
  do{
    counter=0;
    Select randomly cluster from x;
    if(( counter % s ) == 0)
    {
      Generate a random binary digit;
      if(a random digit is 0)
      {
       Select randomly n, where n corresponds to instance number
       which is not represented within x;
       Add n to x;
      }
      else
      {
      Select randomly n, where n corresponds to instance
      number from selected cluster;
      Remove n from x;
      }
    }
    Select randomly n, where n corresponds to instance number
    from selected cluster;
    Select randomly n', where n' corresponds to instance
    number which is not represented within x;
    Remove n from x and add n' to x producing x';
    Calculate fitness of the x' on s(z,L);
    if (x' is better on s(z,L) then x) x=x';
    counter++;
   }while (!terminatingCondition);
   /*solution is ready to be sent back*/
   solution = x;}
}
```

5 Computational Experiment Results

To validate the proposed approach several benchmark instances have been solved. The main aim of the experiment has been to evaluate usefulness and effectiveness of the agent-based approach to solving the problem of multiple-objective selection of reference vectors. This has been achieved through establishing experimentally how different strategies of selecting and using optimization agents affect the computation results.

The proposed approach has been used to solve four well known classification problems - Cleveland heart disease (303 instances, 13 attributes, 2 classes), credit approval (690, 15, 2), Wisconsin breast cancer (699, 9, 2) and sonar problem (208, 60, 2). The respective datasets have been obtained from [8].

Experiment plan has been based on the 10-cross-validation approach. Each thus obtained training set T has been then reduced to a subset S containing reference vectors. Each reference vectors set has been, in turn, used to produce a decision tree. This has been evaluated from the point of view of the four criteria discussed in Section 3. Each decision tree was created using only the instances in S and each C 4.5 classifier was trained without pruned leaves.

For each benchmarking problem the experiment has been repeated 50 times and the reported values of the quality measures have been averaged over all runs. All optimization agents have been allowed to continue iterating until 100 iterations have been performed. The common memory size in JABAT was set to 100 individuals. The number of iterations, the size of common memory and selection criteria have been set out experimentally at the fine-tuning phase. The search for solutions was satisfactory performed at reasonable computation time.

In order to evaluate the resulting Pareto-optimal sets approximations two quality measures have been used [11]. The first measure is the average of the best values of weighted Tchebycheff scalarizing function over a set of systematically generated normalized weight vectors. The set of such weight vectors is denoted and defined as $\Psi_s = \{\Lambda = [\lambda_1, \ldots, \lambda_J] \in \Psi | \lambda_j \in \{0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}, 1\}\}$, where Ψ is the set of all normalized weight vectors and k is a sampling parameter.

Finally, the measure is calculated in the following way:

$$R(A) = 1 - \frac{\sum_{\Lambda \in \Psi_s} s^*_\infty(z^0, A, \Lambda)}{|\Psi_s|}, \tag{7}$$

where $s^*_\infty(z^0, A, \Lambda) = min_{z \in A}\{s_\infty(z, z^0, \Lambda)\}$ and is the best value achieving by function $s_\infty(z, z^0, \Lambda)$ on set A. Before calculating the value of this measure the reference point z^0 was set as an ideal point.

Table 1. Performance of different agent combinations measured using average values of C and R

Optimizing	C measure and standard deviations						R measure and standard deviations		
	C(RLS, IDLS)	C(IDLS, RLS)	C(RLS +IDLS, RLS)	C(RLS, RLS +IDLS)	C(RLS +IDLS, IDLS)	C(IDLS, RLS +IDLS)	RLS	IDLS	RLS+IDLS
f_1, f_2, f_3, f_4	0,464 $\pm 0,12$	0,618 $\pm 0,073$	0,862 $\pm 0,149$	0,208 $\pm 0,084$	0,760 $\pm 0,254$	0,328 $\pm 0,11$	0,858 $\pm 0,003$	0,857 $\pm 0,003$	0,859 $\pm 0,002$
f_1, f_2	0,361 $\pm 0,172$	0,798 $\pm 0,155$	0,735 $\pm 0,087$	0,430 $\pm 0,096$	0,867 $\pm 0,153$	0,422 $\pm 0,183$	0,732 $\pm 0,004$	0,732 $\pm 0,004$	0,734 $\pm 0,003$
f_1, f_3	0,523 $\pm 0,103$	0,728 $\pm 0,052$	0,827 $\pm 0,158$	0,390 $\pm 0,139$	0,824 $\pm 0,084$	0,435 $\pm 0,194$	0,959 $\pm 0,003$	0,961 $\pm 0,003$	0,959 $\pm 0,007$

The second measure is the coverage of the two approximations of the non-dominated set and is defined as:

$$C(A,B) = \frac{|\{z'' \in B\}|\exists z' \in A : z' \succ z''|}{|B|}, \tag{8}$$

where the value $C(A,B) = 1$ means that all points in B are dominated by or are equal to some points in A. The value $C(A,B) = 0$ means that no point in B is covered by any point in A.

Experiment results for different combinations of optimization agents averaged over all benchmark datasets and instances are shown in Table 1. The cost factors (optimization criteria) include classification accuracy, percentage of compression of the training set, number of rules and size of the decision tree. Values of the R measure have been calculated with the sampling parameter k set to 100 and 5 for the bi-objective and four-objective cases, respectively.

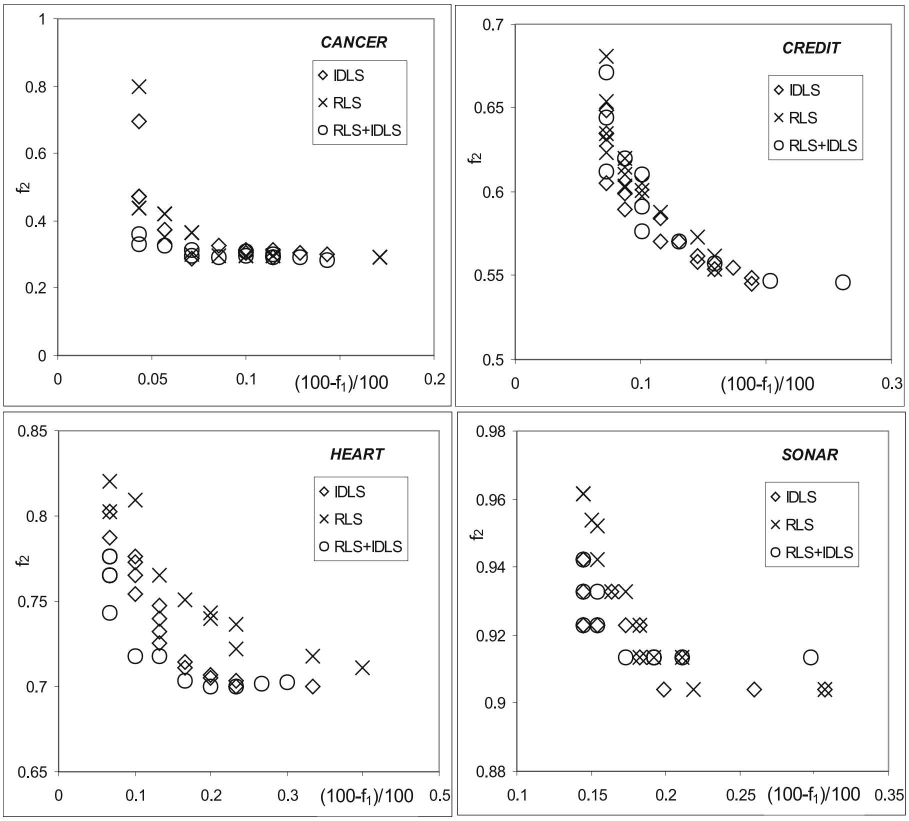

Fig. 1. Example Pareto fronts - instances of the bi-objective optimization (f_1 and f_2)

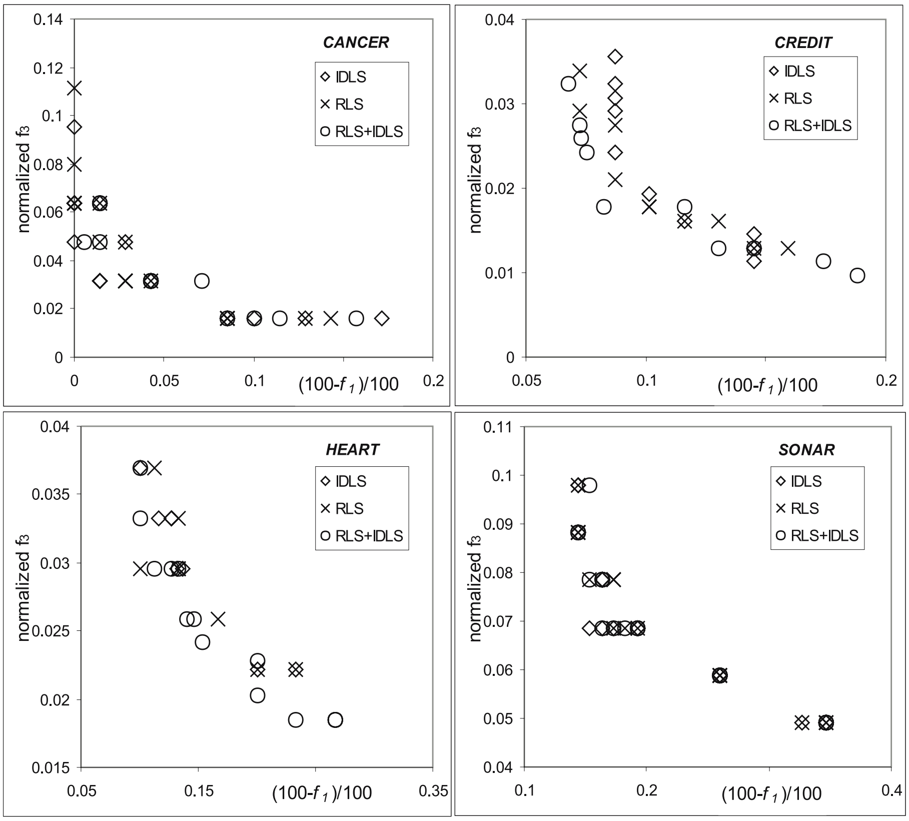

Fig. 2. Example Pareto fronts - instances of the bi-objective optimization (f_1 and f_3)

The results of Pareto-optimal set approximations using the R measure indicate that each combination of agents produces similar results. There are no statistically significant differences between average values of the R measure for all investigated combination of agents.

The results of Pareto-optimal set approximations using the C measure indicate that IDLS produces a better coverage then RLS and RLS+IDLS better coverage then either RLS or IDLS. This observation holds for all investigated cases i.e. multi-objective optimization with two and four objectives and is independent on dimensionality of problems. Thus RLS+IDLS generates best approximation of the Pareto-optimal (non-dominated set).

The values of C measure have been also used to carry a pair-wise comparison of average performance of different combinations of optimization agents. It has been observed that the following inequalities are statistically significant:

- $C(IDLS, RLS) > C(RLS, IDLS)$,
- $C(RLS + IDLS, RLS) > C(RLS, RLS + IDLS)$,
- $C(RLS + IDLS, IDLS) > C(IDLS, RLS + IDLS)$.

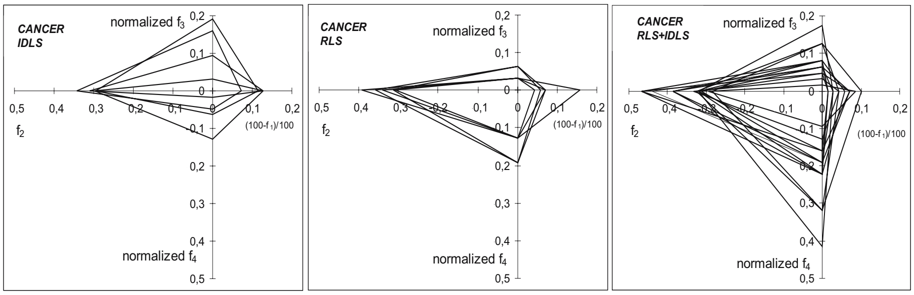

Fig. 3. Example approximations of Pareto-optimal sets - an instance of the four-objective optimization problem

In Fig. 1 and 2 example Pareto fronts obtained by solving a single instance of each of the considered problem types are shown. Each set of points has been obtained in a single run for the respective bi-objective optimization problem.

In Fig. 3 example approximations of Pareto-optimal sets produced by different combination of agents for an instance of the four-objective selection of reference vector problem are presented.

6 Conclusion

The paper proposes an agent-based multiple-objective approach to the selection of reference vectors from original datasets. Effective and dependable selection procedures are of vital importance to machine learning and data mining. The suggested approach is based on the multiple agent paradigm. Using a team of agents brings about several advantages including better use of computational resources, flexibility and ability to carry computations in the distributed environment. The focus of the paper is however not on efficiency of the agent based approach but rather on the methodology of dealing with the multiple-objective selection of reference vectors through employing a team of agents. It has been shown that there exist adequate methodology and suitable tools allowing to obtain good approximations of the Pareto-optimal solutions to problems of the discussed type. The proposed method and tools can be used to design customized machine learning and data mining systems corresponding better to the user requirements and needs. The approach allows also for discovery of interactions between composition of various vector selection optimization procedures and a quality of generalization measured using multiple criteria. Such knowledge can be used for evaluation and selection of optimization agents and procedures.

Future research should focus on refining the theoretical framework for agent-based, multiple-objective optimization of reference vector selection as well as on designing more user friendly tools for solving practical multiple objective reference vectors selection problems.

References

1. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE. A White Paper. Exp 3(3), 6–20 (2003)
2. Talukdar, S., Baerentzen, L., Gove, L., de Souza, P.: Asynchronous Teams: Cooperation Schemes for Autonomous, Computer-Based Agents, Technical Report EDRC 18-59-96, Carnegie Mellon University, Pittsburgh (1996)
3. Jędrzejowicz, P., Wierzbowska, I.: JADE-Based A-Team Environment, pp. 719–726. Springer, Berlin, Heidelberg (2006)
4. Oster, G.F., Wilson, E.O.: Caste and Ecology in the Social Insect, vol. 8. Princeton University Press, Princeton, NJ (1978)
5. Davis, L. (ed.): Handbook of Genetic Algorithms, Van Nostrand Reinhold (1991)
6. Glover, F.: Tabu Search. Part I and II, ORSA Journal of Computing. 1(3), Summer (1990) and 2(1) Winter (1990)
7. Kennedy, J., Eberhart, R.C.: Particle swarm optimisation. In: Proc. of IEEE International Conference on Neural Networks, Piscataway, N.J. pp. 1942-1948 (1995)
8. Merz, C.J., Murphy, P.M.: UCI Repository of Machine Learning Databases Irvine, CA: University of California, Department of Information and Computer Science (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
9. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithm. In: Machine Learning, vol. 33(3), pp. 33–33. Kluwer Academic Publishers, Boston (2000)
10. Duch, W., Blachnik, M., Wieczorek, T.: Probabilistic distance measure for prototype-based rules. In: Proc. of the 12 International Conference on Neural Information Processing, ICONIP, pp. 445–450 (2005)
11. Jaszkiewicz, A.: Multiple objective metaheuristic algorithms for combinational optimization. Habilitation thesis, 360, Pozna University of Technology, Poznań (2001)
12. Czarnowski, I., Jędrzejowicz, P.: An Approach to instance reduction in supervised learning. In: Coenen, F., Preece, A., Macintosh, A. (eds.) Research and Development in Intelligent Systems, vol. XX, pp. 267–282. Springer, London (2004)
13. Stefanowski, J.: Algorytmy indukcji regu decyzyjnych w odkrywaniu wiedzy, Habilitation thesis, 361, Pozna University of Technology, Poznań (in Polish) (2001)
14. Quilan, J.R.: C 4.5: programs for machine learning, San Matoe. Morgan Kaufman, Seattle (1993)
15. Breiman, L., Friedman, J.H., Oslhen, R.A., Stone, C.J.: Classification and Regression Trees. Belmont, CA: Wadsworth International Group (1984)