Rough Sets and Evolutionary Computation to Solve the Feature Selection Problem

Rafael Bello[1], Yudel Gómez[1], Yailé Caballero[2], Ann Nowe[3] and Rafael Falcón[1]

[1] Computer Science Department, Central University of Las Villas (UCLV) Carretera Camajuaní km 5.5, Santa Clara, Cuba
{rbellop, ygomezd, rfalcon}@uclv.edu.cu
[2] Informatics Department, University of Camagüey, Cuba
yaile@inf.reduc.edu.cu
[3] CoMo Lab, Computer Science Department, Vrije Universiteit Brussel, Belgium
asnowe@info.vub.ac.be

Summary. The feature selection problem has been usually addressed through heuristic approaches given its significant computational complexity. In this context, evolutionary techniques have drawn the researchers' attention owing to their appealing optimization capabilities. In this chapter, promising results achieved by the authors in solving the feature selection problem through a joint effort between rough set theory and evolutionary computation techniques are reviewed. In particular, two new heuristic search algorithms are introduced, i.e. Dynamic Mesh Optimization and another approach which splits the search process carried out by swarm intelligence methods.

Key words: meta-heuristic, evolutionary computation, feature selection

1 Introduction

The solution of a great deal of problems can be formulated as an optimization problem. The quest for the problem's solution is often stated as finding the optimum of an objective function $f : D \rightarrow \Re$; i.e., finding a point $x_0 \in D$ such that $f(x_0) \leq f(x)\ \forall\, x \in D$, for the minimization case. The Feature Selection Problem (FSP) can well illustrate this point.

The relevance of the feature selection methods has been widely acknowledged [15, 28]. These methods search throughout the space of feature subsets aiming to find the best subset out of the $2^N - 1$ possible feature subsets (N stands for the number of attributes characterizing the problem). The search is guided by an evaluation measure. Every state denotes a subset of features in the search space.

All feature selection techniques share two crucial components: an evaluation function (used for numerically assessing the quality of a candidate feature subset) and a search engine (an algorithm responsible for the generation of the feature subsets).

The evaluation function attempts to estimate the capability of an attribute or a subset of attributes to discriminate between the collection of existing classes. A subset is said to be optimal with regards to a given evaluation function. Several categories of evaluation functions stand nowadays, like distance measures, information measures (e.g., entropy), dependency measures, consistency measures and classification error measures [8]. More recently, the "quality of the classification" measure borrowed from rough set theory (RST) has been employed as a numerical estimator of the quality of reducts [11, 1, 2, 24, 25]. A reduct is a minimal subset of attributes which preserves the partition over a universe, as stated in [14] wherein also the role played by reducts in feature selection and reduction is explained.

The second component of a feature selection algorithm is the search engine, which acts as a procedure for the generation of the feature subsets. The search strategies are important because this type of problem can be extremely time-consuming and an exhaustive search of a rather "optimal" subset can be proved infeasible, even for moderate values of N. Algorithms to feature selection are usually designed by using heuristics or random search strategies in order to reduce complexity. Heuristic search is very fast because it is not necessary to wait until the search ends but it doesn't guarantee to find the best solution although a better one is known when it is found in the process. An illustrative example of search strategies is given by the evolutionary methodologies.

Evolutionary algorithms perform on the basis of a subset of prospective solutions to the problem, called "population", and they locate the optimal solution through cooperative and competitive activities among the potential solutions. Genetic Algorithms (GA) [10], Ant Colony Optimization (ACO) [9] and Particle Swarm Optimization (PSO) [12] are genuine exemplars of this sort of powerful approaches. They have also been termed as "bioinspired computational models" owing to the natural processes and behaviors they have been built upon.

Diverse studies have been carried out concerning the performance of the above meta-heuristics in the feature selection problem. Some of them have exhibited good results, mainly attained by using ACO- or PSO-based approaches, such as [11, 1, 2, 24, 25, 26]. In [3] and [4], a new approach to feature selection based on the ACO and PSO methodologies is presented. The chief thought is the split of the search process accomplished by the agents (ants or particles) into two stages, such that an agent is commanded in the first stage to find a partial solution to the problem, which in turn is afterwards used as an initial state during the upcoming phase. The application of the two-step approach to the feature selection problem provokes that, after finishing the first stage, agents hold feature subsets which are prospective reducts of the system. They are taken as initial states for the agents during the remaining phase.

The new meta-heuristic named Dynamic Mesh Optimization (DMO) also falls under the umbrella of the evolutionary computation techniques. A set of nodes characterizing potential solutions of an optimization problem make up a mesh which dynamically expands itself and moves across the search space. To achieve this, intermediate nodes are generated at each cycle (iteration) between the mesh nodes and those nodes regarded as local optima, as well as between the mesh nodes and the global optimum. Moreover, new nodes are also generated out of the most external mesh nodes, thus allowing for a broader covering of the search space. The fittest nodes of the ensuing mesh are promoted to make up the mesh at the next cycle. The performance achieved by the DMO procedure in the context of feature selection is studied.

In this chapter, we study the integration between rough set theory and the aforementioned evolutionary algorithms for working out the feature selection problem. No attempt has been made to cover all approaches currently existing in literature but our intention has been to highlight the role played by rough set theory across several evolutionary algorithms in the quest for really meaningful attributes.

The study is structured as follows: after enunciating the fundamentals of rough set theory in section 2, the application of a greedy algorithm to feature selection is presented. Next we elaborate on the way this challenge is tackled by Genetic Algorithms and Swarm Intelligence meta-heuristic approaches, like ACO and PSO. Section 6 discusses the novel DMO evolutionary optimization method whereas section 7 is devoted to unfold a comparative study between the different algorithmic models under consideration. Finally, some conclusions are derived.

2 Rough Set Theory: Basic Concepts

Rough set theory (RST) was proposed by Z. Pawlak [19]. The rough set philosophy is anchored on the assumption that some information is associated with every object of the universe of discourse [21]. Rough set data analysis is one of the main techniques arising from RST; it provides a manner for gaining insight into the underlying data properties [29]. The rough set model has several appealing advantages for data analysis. It is based on the original data only and does not rely on any external information, i.e. no assumptions about data are made. It is suitable for analyzing both quantitative and qualitative features leading to highly interpretable results [23].

In RST a training set can be represented by a table where each row represents an object and each column represents an attribute. This table is called an “information system”; more formally, it is a pair $S = (U, A)$, where U is a non-empty finite set of objects called the universe and A is a non-empty finite set of attributes. A decision system is a pair $DS = (U, A \cup \{d\})$, where $d \in A$

is the decision feature or class attribute. The basic concepts of RST are the lower and upper approximations of a subset $X \subseteq U$. These were originally introduced with reference to an indiscernibility relation $IND(B)$, where objects x and y belong to IND(B) if and only if x and y are indiscernible from each other by features in B.

Let $B \subseteq A$ and $X \subseteq U$. It can be proved that B induces an equivalence relation. The set X can be approximated using only the information contained in B by constructing the B-lower and B-upper approximations of X, denoted by $\underline{B}X$ and $\overline{B}X$ respectively, where $\underline{B}X = \{x \in U : [x]_B \in X\}$ and $\overline{B}X = \{x \in U : [x]_B \cap X \neq \emptyset\}$ and $[x]_B$ denotes the equivalence class of x according to the B-indiscernible relation. The objects in $\underline{B}X$ are guaranteed to be members of X while those in $\overline{B}X$ are possible members of X. The boundary region $BNX = \overline{B}X - \underline{B}X$ determines the roughness of a concept X, namely X is said to be rough if its boundary region is not empty, otherwise it is said to be a crisp (precise) concept.

RST offers several measures for gauging the quality of a decision system. Among them one can find the "quality of classification", displayed in expression (1). It quantifies the percentage of objects which are correctly classified into the given decision classes $Y = \{Y_1, \ldots, Y_n\}$ employing only the knowledge induced by the set of features in B.

$$\gamma_B(Y) = \frac{\sum_{i=1}^{n} |\underline{B}Y_i|}{|U|} \tag{1}$$

An important issue concerning RST is attribute reduction based on the reduct concept. A reduct is a minimal set of features that preserves the partitioning of the universe and hence the ability to perform classifications. The subset B is a reduct if $IND(A) = IND(B)$; that is, $\gamma_A(Y) = \gamma_B(Y)$. The notion of reduct is one of the most important concepts within rough set theory.

However, their practical use is limited because of the heavy workload involved in computing the reducts. The problem of finding a globally minimal reduct for a given information system is NP-hard. For that reason, methods for calculating reducts have been developed on the basis of heuristic-driven approaches [22].

3 A Greedy Algorithm to Feature Selection

The incorporation of rough sets to a greedy approach for finding reducts has been studied in [7]. The method begins with an empty set of attributes and constructs good reducts in an acceptable time. The heuristic search performed by the algorithm adds the fittest attributes to the solution according to some predefined criterion.

The criterion for assessing the quality of an attribute is borrowed from the ID3 classifier with respect to the normalized entropy and the gain of the attributes as well as the degree of dependency between attributes, this latter indicator coming from RST. In this algorithm we use the terms $R(A)$ and $H(A)$ proposed in [20]. $R(A)$ lies within [0,1] and stands for the relative importance of attribute A while $H(A)$ represents heuristic information about a subset of candidate features.

$R(A)$ can be computed by the following expression:

$$R(A) = \sum_{i=1}^{k} \frac{|S_i|}{|S|} \cdot e^{1-C_i} \tag{2}$$

where k is the number of different values of attribute A whereas C_i represents the number of different classes present in the objects having the i-th value for the feature A. Moreover, $|S_i|$ indicates the amount of objects with the value i in the feature A and $|S|$ the total number of objects.

On the other hand, the term $H(A)$ is obtained by the procedure below:

1. Calculate $R(A)$ for each attribute in the problem and make up a vector with the best n attributes (n selected by the user) according to the $R(A)$ indicator. As a result of it, the vector $\mathbf{BR} = (R(A_i), R(A_j), \ldots)$ with $n = |\mathbf{BR}|$ is obtained.
2. Create another vector holding the combinations of n in p (this value also inputted by the user) of the attributes in $\mathbf{BR}$. The combination vector looks like $\mathbf{Comb} = (\{A_i, A_j, A_k\}, \ldots, \{A_i, A_t, A_p\})$
3. Compute the degree of dependency of the decision classes with respect to every combination lying in $\mathbf{Comb}$. Let us denote by $\mathbf{DEP}(d)$ the vector containing the degree of dependency of decision class d with respect to every subset of attributes in $\mathbf{Comb}$, that is $\mathbf{DEP}(d) = (k(Comb_1, d), \ldots, k(Comb_{|Comb|,d}))$
4. Compute $H(A) = \sum_{\forall i: A \in \mathbf{Comb_i}} k(Comb_i, d)$

where $k = \frac{|POS_B(d)|}{|U|}$ and $POS_B(d) = \bigcup_{X \in U/B} \underline{B}X$

If $k = 1$ then d totally depends on B else it partially depends on it.

Another alternative measure that has been used successfully is the gain ratio [17] which is defined in terms of the following measure:

$$\text{SplitInformation}(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \cdot log_2 \frac{|S_i|}{|S|} \tag{3}$$

where c is the cardinality of the domain of values of attribute A. This measure is the entropy of S with respect to A.

The gain ratio $G(A)$ quantifies how much information gain attribute A produces or how important it is to the data set. The formal expression is shown below:

$$G(A) = \frac{G(S,A)}{\text{SplitInformation}(S,A)} \tag{4}$$

$$G(S,A) = \text{Entropy}(S) - \sum_{v \in V_A} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v) \tag{5}$$

where V_A is the set of values of attribute A and S_v is the subset of S for which attribute A has the value v, namely $S_v = \{s \in S \mid A(s) = v\}$

$$\text{Entropy}(S) = \sum_{i=1}^{c} - P_i \cdot \log_2 P_i \tag{6}$$

where P_i is the ratio of objects in S belonging to the i-th decision class.

The cost of an attribute can be defined using expressions (7) or (8):

$$C(A) = \frac{G^2(S,A)}{Cost(A)} \tag{7}$$

where $Cost(A)$ is the cost of attribute A (say, for instance, the cost of running a medical exam). This value ranges between 0 and 1 and must be specified by the user.

$$C(A) = \frac{2^{G(S,A)} - 1}{(Cost(A) + 1)^w} \tag{8}$$

where $Cost(A)$ is just like in (7) and w is a constant value also in [0,1] that determines the relative importance of the cost versus information gain.

Bearing the measures $R(A)$, $H(A)$, $G(A)$ and $C(A)$ in mind, the RSReduct algorithm was devised and implemented as shown in Algorithm 1.

The RSReduct approach has been tested with several data sets from the UCI machine learning repository [6] that are available at the ftp site of the University of California. Some of the databases belong to real-world data such as Vote, Iris, Breast Cancer, Heart and Credit while the other ones represent results obtained in labs such as Balloons-a, Hayes-Roth, LED, M-of-N, Lung Cancer and Mushroom. The results portrayed in Table 1 were obtained after using RSReduct with the three heuristic functions defined in step 2 of Algorithm 1. Furthermore, the execution time of the algorithm has been recorded in each case.

In the experiments displayed in Table 1, $C(A)$ has been computed as in (8), $Cost(A)$ takes random values and $w = 0.1$.

Algorithm 1 RSReduct

```
procedure RSREDUCT( )
    STEP 1 Form the distinction table with a binary matrix $B$ $(m^2 - m)/2 \times (N + 1)$. Each row corresponds to a pair of different objects. Each column of this matrix corresponds to an attribute; the last column corresponds to the decision value (treated as an attribute).
    For each attribute, let $b((k, n), i) \in B$ corresponding to the pair $(O_k, O_n)$ and attribute $i$ be defined as
        $b((k,n),i) = \begin{cases} 1, \text{ if } a_i(O_k) \neg R\, a_i(O_n) \;\; \forall i = \{1, \ldots, N\} \\ 0, \text{ otherwise} \end{cases}$
        $b((k,n),N+1) = \begin{cases} 0, \text{ if } d_i(O_k) \neq d_i(O_n) \\ 1, \text{ otherwise} \end{cases}$
    where $R$ is a similarity relation depending on the type of attribute $a_i$
    STEP 2 For each attribute $A$ calculate the value of $RG(A)$ for any of the following three heuristics and then form an ordered list of attributes starting from the most relevant attribute (that which maximizes $RG(A)$):
    • Heuristic 1: $RG(A) = R(A) + H(A)$
    • Heuristic 2: $RG(A) = H(A) + G(A)$
    • Heuristic 3: $RG(A) = H(A) + C(A)$
    STEP 3 With $i = 1$, $R = \emptyset$ and $(A_1, A_2, \ldots, A_n)$ an ordered list of attributes according to step 2, consider if $i \leq n$ then $R = R \cup A_i$, $i = i + 1$.
    STEP 4 If $R$ satisfies condition I (see below) then Reduct=minimal subset $R' \subseteq R$ does meet condition I, so stop otherwise go to step 3.
    Condition I uses the following relation between objects $x$ and $q$ for attribute $a$: $q_a R x_a \Leftrightarrow sim(x_a, q_a) \geq \varepsilon$ where $0 \leq \epsilon \leq 1$
end procedure
```

4 Feature Selection by using a Genetic Approach

Arising as a true exemplar of evolutionary computation techniques, Genetic Algorithms (GAs) have been widely utilized for attribute reduction. GAs are stochastic search methods based on populations. First, a population of random individuals is generated and the best individuals (in accord with some predefined criterion) are selected. Then, the new individuals making up the population will be generated using the mutation, crossover and (possibly) inversion operators. In [27], three methods for finding short reducts are presented. They use genetic algorithms together with a greedy approach and have defined the adaptability functions f_1, f_2 and f_3.

An adaptation of the GA plan is the Estimation of Distribution Algorithms (EDA) [18] but most of them don't use crossover or mutation because the new population is generated from the distribution of the probability estimated from the selected set. The principal problem of the EDA is the estimation of $ps(x, t)$ and the generation of new points according to this distribution in a way that yields reasonable computational efforts. For this reason, different manners to determine $ps(x, t)$ have been crafted.

Table 1: Average length of reducts and computational time required by the RSReduct approach

Data set name	Heuristic 1		Heuristic 2		Heuristic 3	
(cases#, attr#)	Time(s)	Avg. len	Time(s)	Avg. len	Time(s)	Avg. len
Ballons-a (20,4)	5.31	2	3.12	2	16.34	2
Iris (150,4)	40.15	3	30.79	3	34.73	3
Hayes-Roth(133,4)	36.00	3	32.30	3	39.00	3
Bupa(345,6)	74.20	6	89.00	6	89.00	6
E-Coli(336,7)	57.00	5	41.15	5	46.60	5
Heart(270,13)	30.89	9	16.75	9	54.78	10
Pima(768,8)	110.00	8	110.00	8	110.00	8
Breast-Cancer(683,9)	39.62	4	31.15	4	32.56	5
Yeast(1484,8)	82.00	6	78.00	6	85.70	6
Dermatology(358,34)	148.70	8	125.9	8	190.00	9
Lung-Cancer(27,56)	25.46	7	18.59	7	31.5	8
LED(226,25)	78.10	9	185.00	8	185.00	9
M-of-N(1000,14)	230.26	6	162.50	6	79.4	6
Exactly(780,13)	230.00	11	215.00	11	230.00	11
Mushroom(3954,22)	86.20	8	64.10	8	67.2	8
Credit(876,20)	91.20	14	86.01	14	90.2	15
Vote(435,16)	37.93	12	21.25	11	26.9	12

One of the members of this family is the Univariate Marginal Distribution Algorithm (UMDA) for discrete domains [18], which takes into account univariate probabilities alone. This algorithm is capable of optimizing non-linear functions as long as the additive (linear) variance of the problem has an acceptable weight in the total variance. The UMDA version for continuous domains [16] was introduced in 2000. In every generation and for each variable, UMDA carries out statistic tests to find the density function that best fits to the variable. The continuous variant of UMDA is an algorithm of structure identification in the sense that the density components are identified through hypotheses tests.

We have defined a method [7] for calculating reducts starting from the integration of the adaptability functions (f_1, f_2, f_3) of the methods reported by Wróblewski in [27] and the UMDA approach, thus leading to encouraging results which are shown in Table 2. The values of the parameters used were: $N = 100$; $g = 3000$; $e = 50$; $T = 0.5$ where N is the number of individuals, g is the maximum number of evaluations that will be executed, e is the number of elite (best fitting) individuals which pass directly to the next generation and T is the percentage of the best individuals that were selected to do all the calculations.

In Table 2, AT means the average time required to calculate the reducts (measured in seconds), ARL stands for the average length of the reducts found and ANR their average number.

Table 2: Results obtained with the proposed Estimation Distribution Algorithms (EDA)

Data set name	Algorithms with Wróblewski's functions								
(cases#, attr#)	f_1			f_2			f_3		
	AT	ARL	ANR	AT	ARL	ANR	AT	ARL	ANR
Ballons-a(20,4)	0.167	2	1	1.860	2	1	0.260	2	1
Iris(150,4)	82.390	3	4	3.540	3	4	17.250	3	4
Hayes-Roth(133,4)	40.830	4	1	30.100	4	1	22.450	4	1
Bupa(345,6)	436	3	6.85	995.300	3	8	466	3	8
E-Coli(336,7)	64.150	3	6.85	1514	3	7	169.200	3	7
Heart(270,13)	337	3	8	2782	3	18	1109	3	17
Pima(768,8)	2686	3	17	6460	3	18.4	4387	3	18.6
Breast-Cancer(683,9)	1568	4	6.55	8250	4	7.83	2586	4	8
Yeast(1484,8)	1772	4	2	12964	4	2	2709	4	2
Dermatology(358,34)	1017	6.05	10.15	15553	6	14.90	30658	6	47
Lung-Cancer(27,56)	7.780	4.2	9.55	0.0956	4	15.95	264.200	4	38.6

The use of the three functions reported in [27] in the Estimation of Distribution Algorithms turned out successful. EDA performed the calculation of small reducts in little time when the set of examples was not very large (say, less than 600 cases) even though the number of attributes characterizing the data set was huge. The best combination was accomplished with f_1 when it comes to the execution time; however f_3 found a larger number of reducts in a reasonable time frame.

5 Swarm Intelligence in Feature Selection

This section will unfold the potential of major swarm intelligence approaches to be applied in the feature selection (attribute reduction) problem.

5.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a heuristic method which uses a population of particles and is strongly inspired by the natural behavior of bird flocks and fish schools. Each particle symbolizes a potential solution to the optimization problem. The system begins with an initial population (most of the times, random individuals) and searches for optima according to some fitness function by updating particles over generations; that is, particles "fly" through the N-dimensional problem search space by following the current best-performing particles.

Each particle records its own best position X_{pbest} (that is, its fittest function value ever achieved) as well as the global best position X_{gbest} ever reached by the swarm. As shown in expression (9), the particles are drawn to some degree by X_{pbest} and X_{gbest}. At each iteration the velocity vector $\mathbf{V}$ associated with every particle is updated according to (9). Acceleration constants c_1 and c_2 are empirically determined and used to set up a tradeoff between the exploration and convergence capabilities of the algorithm. The particle's new position is calculated by means of (10).

$$\mathbf{V_i'} = w \cdot \mathbf{V_i} + c_1 \cdot r_1 \cdot (\mathbf{X_{pbest}} - \mathbf{X_i}) + c_2 \cdot r_2 \cdot (\mathbf{X_{gbest}} - \mathbf{X_i}) \tag{9}$$

$$\mathbf{X_i'} = \mathbf{X_i} + \mathbf{V_i} \tag{10}$$

where $\mathbf{V_i}$, $\mathbf{X_i}$, $\mathbf{X_{pbest}}$ and $\mathbf{X_{gbest}}$ are N-dimensional vectors and w is the inertia weight. A suitable selection of w provides a balance between global and local exploration. Random numbers r_1 and r_2 usually follow a normal distribution within [0,1] and outfit the algorithm with the stochastic component.

In feature selection we have a N-dimensional search space, where N is the number of features characterizing the problem. The optimal position along the search space is the shortest subset of features with the highest quality of classification. Being this so, the configuration of the PSO meta-heuristic is as follows: each particle encodes a N-dimensional binary vector with the i-th bit set to one if the corresponding feature is part of the subset and zero otherwise.

The algorithm seeks for minimal reducts R, that is, minimal subsets of features whose quality of the classification $\gamma_R(Y)$ is equal to that yielded by the whole set of features $\gamma_A(Y)$ (Y being the set of decision classes). In this case, the fitness function is the same used in [26], see expression (11), which takes into account the quality of classification and length of the reducts for deeming the worth of a prospective solution. The ensuing optimization activities attempt to maximize the fitness function value.

$$\text{fitness} = \alpha \cdot \gamma_R(Y) + \beta \cdot \frac{N - |R|}{N} \tag{11}$$

In light of the particle encoding scheme used in this proposal, it is necessary to redefine expression (10). The movement of the particle is realized by the flip of the bit value and the velocity is no longer a change ratio of its position but a change probability of it. We propose expression (12) in [4] to calculate the j-dimension of the i-th particle. This is based on the position and velocity update equations of the particle as shown in [13] and [30].

$$X'_{ij} = \begin{cases} 1, \text{ if rand}() \leq \dfrac{1}{1 + e^{1.5 \cdot N \cdot V_{ij}}} \\ 0, \text{ otherwise} \end{cases} \tag{12}$$

The value of the inertia weight w is defined by a positive linear function changing according to the current iteration, as shown below:

$$w = w_{max} - \frac{w_{max} - w_{min}}{NC} \times k \tag{13}$$

where w_{max} is the initial value of the inertia weight, w_{min} its final value, NC the maximal number of cycles (iterations) allowed and k denotes the current iteration number.

The PSO-driven approach is outlined in Algorithm 2.

Algorithm 2 PSO-RST-FS

```
 1: procedure PSO-RST-FS( )
 2:     Generate initial population by setting the $\mathbf{X_i}$ and $\mathbf{V_i}$ vectors
 3:     repeat
 4:         Compute $\mathbf{X_{pbest}}$ for each particle
 5:         Compute $\mathbf{X_{gbest}}$ for the swarm
 6:         $Reducts = Reducts \cup \{R\}$ such that $\gamma_R(Y) = \gamma_A(Y)$
 7:         Update velocity and position of every particle by (9) and (12)
 8:     until $k = NC$
 9:     Output the set $Reducts$
10: end procedure
```

A new approach concerning PSO is introduced in [4]. The Two-Step Particle Swarm Optimization (TS-PSO) algorithm is rooted on the idea of splitting the search process carried out by the particles into two stages so that, in the first stage, preliminary results are reached which could be subsequently used to make up the initial swarm for the second stage. In the case of FSP, this means that subsets of features which are potential reducts of the information system are generated along the first stage. These subsets are used to modify the swarm resulting from the last cycle in the first stage; the modified swarm is used as the initial population of the second stage.

Determining the state by which the search process should commence has long been an interesting problem in heuristic search. It is well known that setting up the initial state has an important bearing over the global search process. The purpose is to be able to approach the initial state to the goal state. Of course, it is necessary to consider an adequate balance between the computational cost of obtaining that initial state and the total cost; in other words, the sum of the cost of approaching the initial state towards the goal state plus the cost of finding the solution from that "improved" location should not be greater than the cost of looking for the solution from a random initial position.

More formally, the aim is the following. Let E_i be the initial state which has been either randomly generated or produced after the execution of any other method without a significant computational cost, E_i^* the initial state generated by some method M that approaches it to the goal state. By $CM(E_i, E_i^*)$ we denote the cost of getting E_i^* from state E_i by means of M and $CCHSA(x)$ is the computational cost involved in finding a solution from state x using a Heuristic Search Algorithm (HSA). Then, the goal is that $CM(E_i, E_i^*) + CCHSA(E_i^*) < CCHSA(E_i)$.

In the two-step approach proposed here, the procedures to generate E_i^* and the HSA are both the PSO algorithm, so the objective is $CPSO(E_i, E_i^*) + CCPSO(E_i^*) < CCPSO(E_i)$. Since PSO is used in both phases, some parameters of the model are meant to distinguish between them. A ratio r is introduced in order to establish the relative setting of the values of the algorithm parameters in both stages; the ratio indicates the proportion of the overall search that will be carried out during the first stage. For example, if $r = 0.3$ for the NC parameter, it means that the first part of the search process will involve 30% of the total number of iterations whereas the subsequent step will be responsible for realizing the remaining 70%.

The parameters that establish the differences between stages are the following: $ratio_Q$ and $ratio_C$. The first one is related to the definition of a quality threshold according to expression (14) and is involved in the selection of the candidate feature subsets. In the first stage, each candidate reduct R whose quality of classification exceeds the quality threshold is selected as a potential reduct. The $ratio_C$ parameter is used to compute the number of cycles in each stage according to (15) and (16).

$$\phi = ratio_Q \cdot \gamma_A(Y) \tag{14}$$

$$nc_1 = \text{round}(ratio_C \cdot NC) \tag{15}$$

$$nc_2 = NC - nc_1 \tag{16}$$

where round(x) denotes the closest integer to x.

The TS-PSO-RST-FS algorithm introduces a step between the first and second phases (called “postprocessing step”) in which the set of potential reducts PR is used to build the **UsedFeatures** and **NotUsedFeatures** N-dimensional binary vectors; the features highlighted in these vectors have value 1 in their corresponding vector component. The **UsedFeatures** vector sets to one its i-th component provided that the i-th feature in the data set belongs to a number of candidate reducts greater than a given percentage threshold, called PerUsed, of the total number of potential reducts found; for instance, if PerUsed=75%, this means that only features which belong to at least the 75% of the potential reducts will receive a signaling in their associated bit within **UsedFeatures**. On the other hand, the **NotUsedFeatures** vector highlights all features belonging to at most PerNotUsed of potential reducts; for instance, if PerNotUsed=30% this means that only features which are included in at most the 30% of the potential reducts become signalized in **NotUsedFeatures**.

By means of the **UsedFeatures** and **NotUsedFeatures** vectors, the optimal swarm of the first stage is modified to give rise to the startup swarm of the second stage in the following way. Each particle $\mathbf{X_i}$ in the optimal swarm is replaced by the vector **UsedFeatures** or is modified by using the vector

NotUsedFeatures in a random way: if $rand() \leq 0.5$ then replace else modify. Modify means that all features included in **NotUsedFeatures** are reset in the particle encoding.

Greater values of the inertia weight during both the first and processing steps help to find good seeds to build the initial swarm for the second stage. The entire description of the two-step approach can be found in Algorithm 3.

Algorithm 3 TS-PSO-RST-FS

```
procedure TS-PSO-RST-FS( )
    *****STAGE 1*****
    Generate initial population by setting the $\mathbf{X_i}$ and $\mathbf{V_i}$ vectors
    repeat
        Compute $\mathbf{X_{pbest}}$ for each particle
        Compute $\mathbf{X_{gbest}}$ for the swarm
        $PR = PR \cup \{R\}$ such that $\gamma_R(Y) \geq \phi$
        Update velocity and position of every particle by (9) and (12)
    until $k = nc_1$
    *****POST PROCESSING STAGE*****
    Compute UsedFeatures and NotUsedFeatures using $PR$
    currentSwarm ← last swarm of stage 1
    for each particle $\mathbf{X_i}$ in currentSwarm do
        if rand() $\leq$ 0.5 then
            $\mathbf{X_i}$ ← UsedFeatures
        else
            Modify $\mathbf{X_i}$ by resetting all features in NotUsedFeatures
        end if
    end for
    *****STAGE 2*****
    repeat
        Compute $\mathbf{X_{pbest}}$ for each particle
        Compute $\mathbf{X_{gbest}}$ for the swarm
        $Reducts = Reducts \cup \{R\}$ such that $\gamma_R(Y) = \gamma_A(Y)$
        Update velocity and position of every particle by (9) and (12)
    until $k = nc_2$
    Output the set $Reducts$
end procedure
```

The algorithms PSO-RST-FS and TS-PSO-RST-FS were executed by using the following parameters: NC= 120, $c_1 = c_2 = 2$, population size = 21 and $\alpha = 0.54$. In the case of the TS-PSO-RST-FS algorithm, $ratio_Q = 0.75$, $ratio_C = 0.3$, PerUsed=66% and PerNotUsed=30%.

In the two-step approach, the values of the ratios have important bearing over the desired outcome. A low value of $ratio_Q$ yields many low-quality potential reducts, consequently the **UsedFeatures** and **NotUsedFeatures** vectors include useless information about the features; therefore, the effect of using **UsedFeatures** and **NotUsedFeatures** to modify the swarm is poor.

On the other side, a value near to one produces subsets close to the definition of reducts in the first stage. As to $ratio_C$, a low value allows to perform a greater quantity of cycles in the second stage from the modified swarm.

The two algorithms were tested and compared using six data sets from UCI Repository. Each algorithm was executed six times on every data set and the average results are offered in Table 3. The performances obtained were compared in terms of the average length of the resulting reduct set (columns 4 and 6) and the number of times in which the algorithm found the minimal reducts (columns 5 and 7). The length of a reduct is defined by the number of features in it.

Table 3: Results obtained with the proposed PSO-based approaches. Columns 4 and 6 display the average length of the reducts while columns 5 and 7 show the number of times the algorithm found minimal reducts.

Data set name	Features	Instances	PSO	PSO	TS-PSO	TS-PSO
(1)	(2)	(3)	(4)	(5)	(6)	(7)
Breast cancer	9	699	4.95	6	4.6	6
Heart	13	294	7.97	4	6.8	6
Exactly	13	1000	6	6	6	6
Credit	20	1000	12.4	4	10.3	5
Dermatology	34	358	15.3	3	12.6	5
Lung	56	32	15.6	3	12.8	5

These results are very interesting because they shed light on the fact that the two-step PSO approach ends up with shorter reducts than the PSO-RST-FS algorithm. So, the certainty of finding minimal length reducts increases by using the TS-PSO-RST-FS method.

5.2 Ant Colony Optimization

Another very popular swarm intelligence technique is Ant Colony Optimization (ACO). ACO is a generic strategy (meta-heuristic) [9] used to guide other heuristics in order to obtain superior solutions than those generated by local optimization methods. In the early ACO model, a colony of artificial ants cooperates to look for good solutions to discrete optimization problems. Artificial ants are simple agents that incrementally build a solution by adding components to a partial solution under construction.

Ant System (AS) [9] is considered as the first ACO algorithm and was introduced using the Travel Salesman Problem (TSP). In TSP, we have a set of N fully connected cities $c_1, \ldots, c_n$ by edges (i, j). Edges have associated pheromone trails τ_{ij} which denote the desirability of visiting city j directly from city i. Also, the function $\eta_{ij} = 1/d_{ij}$ indicates the heuristic desirability of going from i to j, where d_{ij} is the distance between cities i and j. Initially, ants are randomly associated to cities. In the successive steps, ant k applies a random proportional rule to decide which city to visit next according to (17):

$$p_{ij}^k = \frac{(\tau_{ij})^\alpha \cdot (\eta_{ij})^\beta}{\sum\limits_{l \in N_i^k} (\tau_{il})^\alpha \cdot (\eta_{il})^\beta} \qquad \text{if } j \in N_i^k \tag{17}$$

where N_i^k is the neighborhood of the k-th ant while α and β are two parameters that point out the relative importance of the pheromone trail and the heuristic information, respectively. After all ants have built their tours, the values τ_{ij} are updated in two different ways. First, τ_{ij} values decrease because of the evaporation ($\tau_{ij} = (1-\rho) \cdot \tau_{ij}$). The ρ parameter is meant to prevent unlimited pheromone accumulation along the edges. Second, all ants reinforce the value of τ_{ij} on the edges they have passed on in their tours ($\tau_{ij} = \tau_{ij} + Inc_{ij}$), where Inc_{ij} is the amount of pheromone deposited by all ants which included edge (i,j) in their tour. Usually, the amount of pheromone deposited by the k-th ant is equal to $1/C_k$, where C_k is the length of the tour of ant k.

Some direct successor algorithms of Ant Systems are: Elitist AS, Rank-based AS and MAX-MIN AS [9]. A more different ACO approach is Ant Colony System (ACS) which employs the following pseudo-random proportional rule to select the next city j from city i:

$$j = \begin{cases} \arg\max\limits_{l \in N_i^k} \{\tau_{ij} \cdot (\eta_{il})^\beta\}, & \text{if } q < q_0 \\ \\ \text{random selection as in (17), otherwise} \end{cases} \tag{18}$$

where q is a random variable uniformly distributed in [0,1] and $0 \leq q_0 \leq 1$, controls the amount of exploration. In ACS, ants have a local pheromone trail update which is defined as ($\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \rho \cdot \tau_{ij}(0)$) and is applied after crossing an edge (i,j), where $\tau_{ij}(0)$ represents the initial pheromone value. Furthermore, a global pheromone trail update ($\tau_{ij} = (1-\rho) \cdot \tau_{ij} + \rho \cdot Inc_{ij}$) is executed only by the best-so-far ant.

The feature selection problem is an example of a tough discrete optimization problem which can be represented as a graph problem; this is why the ACO model is well suited to solve it.

For this study we used the ACS-RST-FS according to results showed in [1] and [2]. Let $A = \{a_1, a_2, \ldots, a_{nf}\}$ be a set of features. One can think of this set as an undirected graph in which nodes represent features and all nodes are connected by bidirectional links. Pheromone values τ_i are associated to nodes a_i. The amount of pheromone is a function of the dependency of the feature associated to that node to all other features. The pheromone stands for the absolute contribution of that feature to a reduct.

The solution consists of reducts which have to be gradually constructed by the system agents (ants). Initially the ants are distributed over the nodes of the graph and each one stores an empty subset which has to become a candidate reduct. The behavior of a single ant can be described as follows. In the first step, the ant is assigned to one of the nodes, from which it will move to some other node in the network. By doing so, the ant performs a forward selection in which it expands its subset step-by-step by adding new features. To select the next node to visit, the ant looks for all features which are not yet included in the subset and selects the next one according to the ACS rule. On the one hand, it is drawn by the pheromone the other ants have already put down in the graph and, on the other hand, by the heuristic function. We have confined ourselves to choose the standard quality of classification (see expression 1) as the heuristic function for our problem. It is used too for determining whether the candidate subset is a reduct or not. Over time, the quality of the subsets constructed by the ants will improve, which is supported by the monotonicity property of classical RST; these converge to nearly optimal reducts.

The initial deployment of the ants during each cycle (iteration) is governed by the following rules. Recall that m is the population size (number of ants) whereas nf is the number of features present in the data set.

1. If $m <$ nf then perform a random initial distribution of ants.
2. If $m =$ nf then one ant is assigned to each feature.
3. If $m >$ nf then assign the first m ants according to (2) and the remaining ones as in (1).

The process of finding the candidate reduct sets B happens in a sequence of cycles $NC = 1, 2, \ldots$ In each cycle, all ants build their own set B_k. The process stop criterion is met (PSC = true) once the maximal number of cycles has been exceeded $NC > NC_{max}$. Each ant k keeps adding one feature at a time to its current partial set B_k until $\gamma_{B_k}(Y) = \gamma_A(Y)$. This is known as the ant stopping criterion (ASC_k=true). The population size is envisioned as a function of the number of features $m = f(\text{nf})$ where round(x) denotes the closest integer to x.

$\mathbf{R_1}$: If nf < 19 then $m =$ nf
$\mathbf{R_2}$: If $20 \leq$ nf ≤ 49 then if 2/3 nf ≤ 24 then m=24 else m=round(2/3 nf)
$\mathbf{R_3}$: If nf > 50 then if nf/2 ≤ 33 then m $= 33$ else m $=$ round(nf/2)

The above rules are the direct outcome of a thorough experimental analysis conducted with the purpose in mind of setting the population size on the basis of the number of features describing the data set.

Now we are ready to present the ACS-RST-FS approach in a more formal way. Let us look at Algorithm 4.

A new approach in Ant Colony Optimization for solving the feature selection problem was introduced in [3]. The two-step ACO algorithm is also based on the idea of splitting the process of finding reducts into two stages. The algorithm dynamically constructs candidate feature subsets during the first stage which shall be afterwards used as starting points for each ant's

Algorithm 4 ACS-RST-FS

```
 1: procedure ACS-RST-FS( )
 2:     PSC ← false, NC ← 1
 3:     Calculate $\tau_i(0)$, $i = 1, \dots$,nf (random initial values for trail intensity)
 4:     repeat
 5:         Each ant $k$ is assigned to an attribute $a_i, \forall k \in \{1, \dots, m\}$ and $B^k \leftarrow \{a_i\}$
 6:         $ASC_k \leftarrow$ false $\forall k \in \{1, \dots, m\}$
 7:         repeat
 8:             for $k \leftarrow 1$ to $m$ do
 9:                 if $ASC_k =$ false then
10:                     Select new feature $a_i^*$ according to (1)
11:                     $B_k = B_k \cup \{a_i^*\}$
12:                     $\tau_i \leftarrow (1 - \xi) \cdot \tau_i + \xi \cdot \tau_i(0)$          ▷ $i$ is the index of $a_i^*$
13:                     Update $ASC_k$                ▷ Did ant $k$ complete a reduct $B_k$?
14:                 end if
15:             end for
16:         until $ASC_k =$true $\forall k \in \{1, \dots, m\}$
17:         $B_k^* \leftarrow$ best $B_k$   ▷ Now that all ants have finished, select the best reduct
18:         for each $a_i \in B_k^*$ do
19:             $\tau_i \leftarrow (1 - \rho) \cdot \tau_i + \rho \cdot \gamma_B^k(Y)$          ▷ update global pheromone trail
20:         end for
21:         For each feature $i$ do $\tau_i = \frac{\tau_i}{\sum_{j=1}^{n} \tau_j}$
22:         NC ← NC + 1
23:         Update PSC
24:     until PSC = true
25: end procedure
```

own candidate feature subset in the second stage. The number of cycles, the number of ants and the desired quality of the subsets are degrees of freedom of the model related to each stage. We use the same ratio r that affects the three aforementioned parameters. For instance, suppose we are interested in carrying out 100 cycles as the overall algorithm's execution and we will use 30 ants for generating B_k subsets with the maximum possible quality of classification ($NC_{max} = 100, m = 30, \gamma_B(Y) = 1$). Setting $r = 0.3$ means that the first stage will last only 30 iterations, involving 9 ants and will settle for reducts whose quality would be 0.3 or above. Being this so, the values of these parameters during the second phase will be $NC_{max} = 70$, $m = 21$ and the algorithm will look for subsets with the maximum possible quality of classification. The workflow of activities of the TS-ACS-RST-FS proposed approach is depicted in Algorithm 5.

Of course, any other alternative ACO-based implementation can be used rather than the ACS-RST-FS algorithm. An important issue in this approach is to study which is the most suitable value for ratio r. High values of r (near to 1) cause the two-step algorithm to obtain candidate subsets close to the

Algorithm 5 TS-ACS-RST-FS

```
procedure TS-ACS-RST-FS( )
    Compute the population size ($m$) on the basis of the number of features (nf)
    Compute the quality of classification using (1) and $B = A$
    STAGE 1
    Calculate parameter values in the first stage as follows:
    $NC_{max1} = r \cdot NC_{max},\ m_1 = r \cdot m,\ \gamma_{B1}(Y) = r \cdot \gamma_B(Y)$
    Run the ACS-RST-FS approach
    CS ← output of ACS-RST-FS                    ▷ CS holds the candidate reducts
    STAGE 2
    Calculate parameter values in the second stage as follows:
    $NC_{max2} = NC_{max} - NC_{max1},\ m_2 = m - m_1,\ \gamma_{B2}(Y) = \gamma_B(Y)$
    Run the ACS-RST-FS approach but assign in each cycle a random subset
    from CS as initial subset for each ant
end procedure
```

definition of reducts in the first stage, therefore ants in the second step swiftly find reducts but using very limited iterations and a scarce number of search agents (ants). On the contrary, if the ratio value is low, the quality of the candidate feature subsets computed during the first stage is poor yet there are more ants to work for a larger number of cycles in the second stage. We have developed an experimental study which is concerned with this tradeoff.

The following values for the ratio parameter have been proposed $r \in \{0.2, 0.3, 0.36, 0.5, 0.6, 0.8\}$ and the impact of each of these values over the number of reducts obtained, their length as well as the computational time needed to produce the output has been observed. Table 4 reports the average results achieved after 20 iterations. A synthetic repository comprised of 20 objects which are described by 16 features provides the data for conducting the experiments. The maximum number of cycles is 21.

Table 4: Results obtained with the proposed ACO-based approaches. The last two columns portray the average number of reducts, the average length of the reducts and the computational time (in seconds), these three indicators separated by backslash.

Algorithm	NC_{max1}	NC_{max2}	m_1	m_2	$\beta = 5, q_0 = 0.9$	$\beta = 1, q_0 = 0.3$
ACS	–	–	–	–	46.7/3.95/228	123/4.19/274
TS-ACS $r = 0.2$	4	17	3	13	32.7/4.2/82	76.3/4.2/89.9
TS-ACS $r = 0.3$	6	15	5	11	43.3/4.1/53	71.3/4.2/64
TS-ACS $r = 0.36$	8	13	6	10	38.7/3.9/39	67.3/4.1/47
TS-ACS $r = 0.5$	10	11	8	8	29.7/3.8/32	43.3/4.1/44
TS-ACS $r = 0.6$	13	8	10	6	20.33/3.8/41	37/4.2/49
TS-ACS $r = 0.8$	17	4	13	3	9/3.8/82	10.67/4.2/97

We can see that $r = 0.3$ bears the best results. This setting implies a number of reducts similar to ACS-RST-FS but only in the 23% of the time. Similar results can be witnessed across other data sets. For instance, in Table 5 we display the results using the Breast Cancer database from UCI Repository. The result here is not surprising, since the value $r = 0.3$ provides a good balance between both stages; a higher number of ants and cycles in the second stage allows the algorithms to perform a larger exploration of the search space departing from initial subsets with an acceptable quality.

Another point worthwhile stressing is that the time complexity of TS-ACS-RST-FS is very low. In light of this, we propose a second idea: to increase the number of ants in order to bring about a greater exploration of the search space. In Table 6 the same data set than in Table 4 was used but now the population size is increased by the factors 1.33, 1.5, 1.8 and 2.1, respectively. In columns 4 and 5 the relationship between each alternative and the ACS-RST-FS benchmark algorithm is reported in terms of the amount of reducts achieved and the computational time needed. For instance, when the number of ants is $1.8m$, the TS-ACS-RST-FS approach gets 120% of reducts with respect to the number of reducts computed via ACS-RST-FS only in 52% of the time required by the latter one (for $\beta = 5$ and $q_0 = 0.9$). Here we have set $r = 0.3$ because this value accomplishes the most encouraging results throughout several experimental studies. In the case of $\beta = 1$ and $q_0 = 3$, the TS-ACS-RST-FS method reached the same number of reducts (99%) but only using 69% of the CPU time than its counterpart, the ACS-RST-FS model.

Table 5: A comparison between ACS and several configurations of the two-step ACO approach using $r = 0.3$, $NC_{max1} = 6$ and $NC_{max2} = 15$

Algorithm	m_1	m_2	$\beta = 5, q_0 = 0.9$	$\beta = 1, q_0 = 0.3$
ACS ($m = 16$)	–	–	46.7/228	123/274
TS-ACS ($m = 16$)	5	11	92%/23%	58%/23%
TS-ACS ($m' = 1.33m = 21$)	6	15	96%/31%	81%/37%
TS-ACS ($m' = 1.5m = 24$)	7	17	109%/38%	83%/42%
TS-ACS ($m' = 1.8m = 29$)	9	20	120%/52%	89%/55%
TS-ACS ($m' = 2.1m = 34$)	10	24	126%/66%	99%/69%

Table 6 sketches a similar study using the Breast Cancer database. These results are very interesting because the two-step ACO approach enables us to obtain the same or an even greater number of reducts in less time than ACS-RST-FS, hence the feasibility of splitting the search process is empirically confirmed once again.

Table 6: A comparison between ACS and several configurations of the two-step ACO approach using $NC_{max1} = 6$ and $NC_{max2} = 15$

Algorithm	m_1	m_2	$\beta = 5, q_0 = 0.9$	$\beta = 1, q_0 = 0.3$
ACS ($m = 9$)	–	–	46.7/228	123/274
TS-ACS ($r = 0.2$)	2	7	60%/34%	70%/49%
TS-ACS ($r = 0.3$)	3	6	109%/31%	73%/37%
TS-ACS ($r = 0.36$)	3	6	105%/25%	77%/31%
TS-ACS ($r = 0.5$)	4	9	100%/22%	73%/26%
TS-ACS ($r = 0.6$)	5	4	65%/13%	50%/20%
TS-ACS ($r = 0.8$)	7	2	33%/27%	31%/26%
TS-ACS ($r = 0.3, m' = 1.8m = 16$)	5	11	102%/58%	98%/74%
TS-ACS ($r = 0.3, m' = 2.1m = 19$)	6	13	124%/67%	103%/83%

6 Dynamic Mesh Optimization in Feature Selection

We want to elaborate now on a novel optimization technique called "Dynamic Mesh Optimization" (DMO) [5] which follows some patterns already present in earlier evolutionary approaches but provides a unique framework for managing both discrete and continuous optimization problems.

The essentials behind the DMO method is the creation of a mesh of points in the multi-dimensional space wherein the optimization of the objective function is being carried out. The mesh endures an expansion process toward the most promising regions of the search space but, at the same time, becomes finer in those areas where there exist points that constitute local ends of the function. The dynamic nature of the mesh is given by the fact that its size (number of nodes) and configuration both change over time. When it comes to the feature selection problem, nodes can be visualized as binary vectors $\mathbf{n} = (n_1, n_2, \ldots, n_N)$ of N components, one per attribute, with the component $n_i = 1$ if the i-th attribute is being considered as part of the solution or zero otherwise. This is the same representation adopted in the previously discussed evolutionary approaches.

At every cycle, the mesh is created with an initial number of nodes. Subsequently, new nodes are generated until an upper boundary in the number of nodes is reached. The mesh at the next cycle is comprised of the fittest nodes of the mesh in the current iteration. Along the search process, the node carrying the best value of the objective (evaluation) function so far is recorded, so n_g denotes the global end attained up to now by the search algorithm.

In the case of the feature selection problem, the evaluation (fitness) function for the DMO meta-heuristic is expression (11), which attempts to achieve a tradeoff between the classificatory ability of a reduct and its length.

The dynamic nature of our proposal manifests in the generation of (i) the initial mesh; (ii) intermediate nodes oriented toward the local optima; (iii) intermediate nodes in the direction of the global optimum and (iv) nodes aiming at expanding the dimensions of the current mesh.

The model gives rise to the following parameters: (i) $N_i \rightarrow$ size of the initial mesh, (ii) $N \rightarrow$ maximum size of the mesh across each cycle ($N_i < N$) and (iii) $M \rightarrow$ number of cycles.

The DMO method is defined in the following manner:

STEP 1. Generate the initial mesh for each cycle: At the beginning of the algorithm's execution, the initial mesh will be made up of N_i randomly generated nodes while in the remaining iterations, the initial mesh is built upon the selection of the best (in terms of evaluation measure) N_i nodes of the mesh in the preceding cycle.
STEP 2. Node generation toward local optima: The aim of this step is to come up with new nodes laid in the direction of the local optima found by the algorithm.

For each node **n**, its K-nearest neighbor nodes are computed (the Hamming distance is a suitable option for the FSP). If none of the neighbors surpasses **n** in fitness function value, then **n** is said to be a local optimum and no nodes are begotten out of it in this step. Conversely, suppose that node **ne** is "better" than **n** and the rest of its neighbors. In this case, a new node arises somewhere between **n** and **ne**.

The proximity of the newly generated node **n*** to the current node **n** or to the local optimum **ne** is contingent upon a factor r which is calculated based on the fitness function values at both nodes **n** and **ne**. Each component of **n*** takes either the value of n_i or ne_i according to a rule involving a stochastic component. The threshold r determining how every component n_i^* must be fixed is calculated as in (19).

$$r = 1 - 0.5\,\frac{\text{Eval}(n)}{\text{Eval}(ne)} \tag{19}$$

$f(\mathbf{n}, \mathbf{ne}, r)$: For each component n_i: If random() $< r$ then $n_i^* = ne_i$ otherwise $n_i^* = n_i$

Notice from (19) that the lower the ratio between Eval(n) and Eval(ne), the more likely it is that n_i^* takes the value of the i-th component of the local optimum.
STEP 3. Node generation toward global optimum: Here the idea is the same as in the previous step but now r is computed differently and a new function g is introduced. Needless to say that **ng** represents the global optimum found thus far by the algorithm.

$$r = 1 - 0.5\,\frac{\text{Eval}(n)}{\text{Eval}(ng)} \tag{20}$$

$g(\mathbf{n}, \mathbf{ng}, r)$: For each component n_i: If random() $< r$ then $n_i^* = ng_i$ otherwise $n_i^* = n_i$
STEP 4. Mesh expansion: In this step, the mesh is stretched from its outer nodes using function h, i.e. using nodes located at the boundary of the initial

mesh in each cycle. The weight w depicted in (13) assures that the expansion declines along the search process (i.e., a bigger expansion is achieved at the early cycles and it fades out as the algorithm progresses). To determine which nodes lie in the outskirts of the mesh, we turn to the norm of a vector. Those nodes exhibiting the lowest and greatest norm values are picked. Remark that, in this step, as many outer nodes as needed are selected so as to fill out the maximum mesh size N. The rules regulating this sort of node generation can be found next:

For each node **nl** in the lower boundary (those with the lowest norm): $h(\mathbf{nl}, w)$: For each component n_i: If random() $< w$ then $n_i^* = 0$ otherwise $n_i^* = nl_i$

For each node **nu** in the upper boundary (those with the greatest norm): $h(\mathbf{nu}, w)$: For each component n_i: If random() $< w$ then $n_i^* = 1$ otherwise $n_i^* = nu_i$

In the context of feature selection, the norm of a node (vector) is the number of components set to one. Algorithm 6 outlines the workflow of the DMO approach. It is also worth remarking that no direct search algorithm guarantees to find the global optimum no matter how refined the heuristic search might be.

Algorithm 6 The DMO meta-heuristic

```
1: procedure D(M)O
2:     Randomly generate $N_i$ nodes to build the initial mesh
3:     Evaluate all the mesh nodes
4:     repeat
5:         for each node $n$ in the mesh do
6:             Find its K-nearest neighbors
7:             $n_{best} \leftarrow$ the best of its neighbors
8:             if $n_{best}$ is better than $n$ then
9:                 Generate a new node by using function $f$
10:            end if
11:        end for
12:        for each initial node in the current mesh do
13:            Generate a new node by using function $g$
14:        end for
15:        repeat
16:            Select the most outward node of the mesh
17:            Generate a new node by using function $h$
18:        until $MeshSize = N$
19:        Select the best $N_i$ nodes of the current mesh and set up the next mesh
20:    until $CurrentIteration = M$
21: end procedure
```

7 A Comparative Study

The conducted experimentation embraces a comparison between DMO and existing ACO- and PSO-based approaches. The chosen criteria were the number and length of the reducts found as well as the computational time required by every method.

Concerning ACO, the Ant Colony System (ACS) model was picked for benchmarking following the advice in [1] and [2], for it reported the most encouraging outcomes. As to the parameter setting, we stuck to the guidelines provided in the aforesaid studies, i.e. $\beta = 5, q_0 = 0.9, NC_{max} = 21$ and the population size (number of ants) depending on the number of features as in the previously enunciated rules.

Regarding the TS-ACS-RST-FS approach, the value of the ratio r used for determining the number of ants, number of cycles and threshold of the quality of the classification in each stage was set to 0.3 whereas the number of ants m is increased 2.1 times, i.e. $m' = 2.1m$

Moving on to the PSO-driven approaches' configuration, each individual was shaped as a binary vector whose length matches the number of attributes in the system. The parameters associated with the PSO-RST-FS were fixed as $c_1 = c_2 = 2, maxCycles = 120$ and swarmSize = 21. The inertia weight w keeps its dynamic character as reflected in (13). As to the TS-PSO-RST-FS method, the factor used to calculate the quality of the classification in the first stage ($ratio_Q$) takes 0.75 while the parameter involved in the computation of the number of cycles ($ratio_C$) for each phase was set to 0.3.

The configuration of the DMO-RST-FS (DMO + RST to feature selection) has been defined as follows: a mesh with 30 nodes is used, 9 of them regarded as initial nodes (which means that it is necessary to generate 21 nodes per cycle, just the same number of particles than in the PSO-based models) and the computations lasted for 90 iterations.

Table 7 reports the experimental results obtained after applying the above methods over the Breast Cancer, Heart and Dermatology data sets coming from the UCI Repository. Each table entry holds the average number of reducts found, the average length (number of attributes) of the reducts in addition to the length of the shortest reduct and the number of times it was found with regards to the number of runs performed by the algorithm. Every algorithm was executed six times per data set. From the information in Table 7 we notice, for instance, that the DMO-RST-FS algorithm discovered 18.3 reducts on average for the Breast Cancer data set, the reducts having average length of 5.1 and the shortest reduct found is composed of four attributes, having a reduct of such length always (100%) been found throughout the different runs of the algorithm.

From the outlook of the computational cost, one may notice that the DMO-RST-FS, TS-PSO-RST-FS and PSO-RST-FS algorithms have a very similar performance. This is clearly understood if we keep in mind that the greater the number of times expression (1) is computed, the more time-consuming

Table 7: Quality of the reducts found by different evolutionary algorithms. First datum is the average number of reducts found, followed by their average length, the length of the shortest reduct and, finally, the percentage of times a reduct of the same length was found throughout the different executions of the algorithm.

Method	BreastCancer	Heart	Dermatology
DMO-RST-FS	18.3/5.1/4,100%	14.8/8.29/6, 83%	179.5/20.9/9,50%
TS-PSO-RST-FS	11/4.6/4,100%	3/6.8/6,100%	39.3/12.6/9, 50%
PSO-RST-FS	14/4.95/4,100%	6/7.97/6,67%	78.2/15.3/9, 50%
TS-ACS-RST-FS	12.7/4.74/4,100%	7/7/6,67%	249/13/9,33%
ACS-RST-FS	11.75/4.94/4,100%	14.3/7.53/6,100%	300/14.17/10,66%

the algorithm turns into. While PSO-based and DMO approaches compute this indicator roughly $P \times Q$ times (P being the number of cycles and Q the number of agents engaged in the search process, viz particles in PSO and nodes in DMO), the ACO-based models evaluate this function a far greater number of times, i.e. roughly $P \times Q \times k$ (Q being the number of ants and k the average length of the reducts found, since every time an ant adds a node to the solution, it must evaluate all possible alternatives at hand, namely, all attributes still not considered so far).

Regarding the average amount of times the fitness function was calculated by all approaches under discussion, Table 8 presents the corresponding magnitudes.

Table 8: Average number of evaluations of the fitness function in each algorithm

Algorithm	Avg. number of times
DMO-RST-FS	2530
TS-PSO-RST-FS	2542
PSO-RST-FS	2968
TS-ACS-RST-FS	17222
ACS-RST-FS	13487

8 Conclusions

An study on the performance of several evolutionary techniques for tackling the feature selection problem has been outlined. The common denominator has been the pivotal role played by rough set theory in assessing the quality of a feature subset as a prospective reduct of the system under consideration.

Therefore this criterion has been successfully incorporated to the fitness function of all the studied algorithms and the preliminary results allow to confirm the feasibility and efficiency of this sort of techniques for attribute reduction.

Moreover, the introduction of the two-step search paradigm for the swarm intelligence methods translated into a substantial reduction of the computational time needed to find the reducts of the information system.

Under empirical evidence we can also conclude that the Dynamic Mesh Optimization approach explores the search space in a similar way to the algorithms based on the Ant Colony Optimization model but with a computational cost very close to that of Particle Swarm Optimization.

Acknowledgment The authors would like to thanks VLIR (Vlaamse InterUniversitaire Raad, Flemish Interuniversity Council, Belgium) for supporting this work under the IUC Program VLIR-UCLV.

References

1. Bello R, Nowe A (2005) Using ACO and rough set theory to feature selection. In: WSEAS Trans. on Information Science and Applications 512–517
2. Bello R, Nowe A (2005) A model based on ant colony system and rough set theory to feature selection. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'05) 275–276
3. Bello R, Puris A, Nowe A, Martínez Y, García M (2006) Two-step ant colony system to solve the feature selection problem. In: Martínez-Trinidad JF, Carrasco JA, Kittler J (eds.) CIARP 2006. LNCS 4225. Springer, Heidelberg
4. Bello R, Gómez Y, Nowé A, García M (2007) Two-step particle swarm optimization to solve the feature selection problem. In: Proc. of 7th Int'l Conf. on Intelligent Systems Design and Applications, 691–696. IEEE Computer Society, Washington DC
5. Bello R, Puris A, Falcón R (2008) Feature selection through dynamic mesh optimization. Accepted for oral presentation at the 13th Iberoamerican Congress on Pattern Recognition (CIARP 2008), Havana, Cuba
6. Blake CL, Merz CJ (1998) UCI repository of machine learning databases http://www.ics.uci.edu/~mlearn/MLRepository.html.
7. Caballero, Y, Bello R (2006) Two new feature selection algorithms with rough sets theory. In: Bramer M (eds) Artificial Intelligence in Theory and Practice. Springer, Boston
8. Dash M, Liu H (2003) Consistency-based search in feature selection. Artificial Intelligence 151:155–176
9. Dorigo M, Stützle T (2005) Ant colony optimization. Cambridge University Press, New York
10. Goldberg, DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley Longman Publishing Co., Boston
11. Jensen R, Shen Q (2003) Finding rough set reducts with ant colony optimization. In: Proceedings of 6th Annual UK Workshop on Computational Intelligence, 15–22

12. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proc. of IEEE Int'l. Conf. on Neural Networks, 1942–1948. IEEE Service Center, Piscataway
13. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm optimization algorithm. In: IEEE International Conference on Neural Networks, 4104–4108.
14. Komorowski J, Polkowski L, Skowron A (1999) Rough sets: a tutorial. In: Pal S, Skowron A (eds): Rough-Fuzzy Hybridization: A New Trend in Decision-Making. Springer-Verlag, New York
15. Kudo M, Sklansky J (2000) Comparison of algorithms that select features for pattern classifiers. Pattern Recognition 33:25–41
16. Larrañaga P, Etxeberria R, Lozano JA, Peña JM (2000) Optimization in continuous domains by learning and simulation of Gaussian networks. In: Wu AS (ed) Proceedings of the 2000 Genetic and Evolutionary Computation Conference, 201–204
17. Mitchell TM (1997) Machine learning. McGraw Hill
18. Mühlenbein H, Mahnig T, Ochoa A (1999) Schemata, distributions and graphical models on evolutionary optimization. Journal of Heuristics 5(2):215–247
19. Pawlak Z (1982) Rough sets. International Journal of Information & Computer Sciences 11:341–356
20. Piñero P, Arco L, García M, Caballero Y (2003) Two new metrics for feature selection in pattern recognition. LNCS 2905, 488-497. Springer-Verlag, Berlin Heidelberg
21. Polkowski L (2002) Rough sets: mathematical foundations. Physica-Verlag, Berlin
22. Stefanowski J (2004) An experimental evaluation of improving rule-based classifiers with two approaches that change representations of learning examples. Engineering Applications of Artificial Intelligence 17:439–445
23. Tay FE, Shen L (2002) Economic and financial prediction using rough set model. European Journal of Operational Research 141:641–659
24. Wang X, Yang Y, Peng N, Teng X (2005) Finding minimal rough set reducts with particle swarm optimization. In: Slezak D, Yao J, Peters J, Ziarko W, Xiaohua H (eds): RsFsDGrC 2005. LNCS 3641, 451–460. Springer, Heidelberg
25. Wang X, Yang J, Jensen R, Liu X (2006) Rough set feature selection and rule induction for prediction of malignancy degree in brain glioma. Computer Methods and Programs in Biomedicine 83:147–156
26. Wang X, Yang J, Teng X, Xia W, Jensen R (2007) Feature selection based on rough sets and particle swarm optimization. Pattern Recognition Letters 28:459–471
27. Wroblewski, J (1995) Finding minimal reducts using genetic algorithms. In: Wang PP (ed) Proceedings of the 2nd Annual Joint Conference on Information Sciences, 186-189
28. Xing H, Xu L (2001) Feature space theory -a mathematical foundation for data mining. Knowledge-based systems 14:253-257
29. Xu ZB, Liang J, Dang C, Chin KS (2002) Inclusion degree: a perspective on measures for rough set data analysis. Information Sciences 141:227-236
30. Yuan S, Chu FL (2007) Fault diagnostics based on particle swarm optimization and support vector machines. Mechanical Systems and Signal Processing 21:1787–1798