Feature Selection through Dynamic Mesh Optimization

Rafael Bello, Amilkar Puris, and Rafael Falcón

Universidad Central de Las Villas (UCLV)
Santa Clara 54830, Villa Clara, Cuba
rbellop@uclv.edu.cu

Abstract. This paper introduces the Dynamic Mesh Optimization meta-heuristic, which falls under the evolutionary computation techniques. Moreover, we outline its application to the feature selection problem. A set of nodes representing subsets of features makes up a mesh which dynamically grows and moves across the search space. The novel methodology is compared with other existing meta-heuristic approaches, thus leading to encouraging empirical results.

Keywords: 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 [12]. These methods search throughout the space of feature subsets aiming to find the best subset among 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. An 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 [5]. 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 [8] [1] [2] [13] [14]. A

J. Ruiz-Shulcloper and W.G. Kropatsch (Eds.): CIARP 2008, LNCS 5197, pp. 348–355, 2008.

reduct is a minimal subset of attributes which preserves the partition over a universe [11]. This very study elaborates on the role played by reducts in feature selection and reduction.

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. 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) [7], Ant Colony Optimization (ACO) [6] and Particle Swarm Optimization (PSO) [9] 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 [8], [1], [2], [13], [14] and [15]. In [3] and [4], a new approach to feature selection based on the ACO and PSO methodologies are presented. The chief thought is the split of the search process accomplished by the agents (ants or particles) into two stages, such that ants are commanded in the first stage to find partial solutions to the problem, which in turn are afterwards used as initial states during the upcoming phase. The application of the two-step approach to the feature selection problem yields 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.

Bearing this in mind, we come up with a new meta-heuristic named Dynamic Mesh Optimization (DMO), which falls under the umbrella of the evolutionary computation techniques. A set of nodes portraying potential solutions of an optimization problem set 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 bore by the DMO procedure in the context of feature selection is studied.

The paper is structured as follows: Section 2 elaborates on the proposed optimization approach whereas an empirical analysis taking into account other existing meta-heuristics can be found at Section 3. Conclusions and further work are outlined in Section 4.

2 An Overview of the DMO Meta-heuristic

The essentials of the DMO method are the creation of a mesh of points in the N-dimensional space wherein the optimization of $f(x_1, x_2, \ldots, x_n)$ is 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 exists 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 depicted as binary vectors $n(x_1, x_2, \ldots, x_n)$ 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.

In each 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 former 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 function is displayed by expression (2), which is anchored in the "quality of the classification" measure from rough set theory.

2.1 Rough Set Theory

Due to the limited space available for presenting the paper, we prefer to skip the essentials of rough set theory which can be found anywhere in literature and directly target the "quality of classification" measure, which describes the percentage of objects correctly classified into the set of classes $Y = \{Y_1, Y_2, \ldots, Y_k\}$ by using information coming from attributes in B. The partition Y is induced by the decision attribute and every Y_i is a disjoint set of objects. So, putting all this into a numerical expression, we have:

$$\gamma_B(Y) = \frac{\sum_{i=1}^{k} \mid B_* Y_i \mid}{\mid U \mid} \tag{1}$$

The notion of *reduct* is the foundation of the rough-set-aided feature selection. A reduct is a minimal subset of features $B \subseteq A$ such that $IND(B) = IND(A)$, where $IND(X)$ is called an *inseparability relation*, i.e., the inseparability relation of the system is preserved. Hence, $\gamma_B(Y) = \gamma_A(Y)$ and in consistent decision systems the equality $\gamma_B(Y) = \gamma_A(Y) = 1$ holds.

The evaluation function of the mesh nodes is displayed in expression (2) and is the same used in [15] and [4]. It considers the number of attributes included in the feature subset R represented by node n and the quality of the classification associated to R. The goal is to maximize the function below.

$$Eval(n) = \alpha \cdot \gamma_R(DS) + \beta \cdot \frac{N - \mid R \mid}{N} \tag{2}$$

2.2 Generation of Nodes in DMO

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.

During the mesh expansion in each cycle, a weight w is defined using expression (3) as in [14], [15] and [4].

$$w = (w_0 - 0.4) \times \frac{M - j}{M + 0.4} \tag{3}$$

where $w_0 = 1.4$ is fixed as the initial weight and j is the current iteration number.

2.3 The Algorithm Unfolded

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 settled 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 evaluation function values both at 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 value. The threshold r determining how every component n_i^* is set is calculated by expression (4).

$$r = 1 - 0.5 \frac{Eval(n)}{Eval(ne)} \tag{4}$$

$f(n, ne, r)$: For each component n_i: If $Random() < r$ then $n_i^* = ne_i$ otherwise $n_i^* = n_i$

Notice from (4) 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 function g is introduced. Needless to say that ng represents the global optimum found thus far by the algorithm.

$$r = 1 - 0.5 \frac{Eval(n)}{Eval(ng)} \tag{5}$$

$g(n, ng, r)$: 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 (3) assures that the expansion declines all over 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, those having the lowest and greatest norm 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 governing this sort of node generation can be found next:

For each node nl in the lower boundary (those with lower norm):
$h(nl, w)$: 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 greater norm):
$h(nu, w)$: 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 its components set to 1. Finally, Algorithm 1 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.

3 Empirical Results

The conducted experimentation included 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 of the methods. Concerning ACO, the Ant Colony System (ACS) model was picked for benchmarking, for it reported the most encouraging outcomes in [1]. The ACS-RST-FS (ACS + RST to feature selection) algorithm [1] and its improved version, the TS-ACS-RST-FS (two-stage approach) [3] were selected for performing the experiments.

As to the parameter setting, we stuck to the guidelines provided in the aforementioned works, i.e. $\beta = 5$, $q_0 = 0.9$, $maxCycles = 21$ and the number of ants depending on the number of features, thus complying with the rules stated in

Algorithm 1. The DMO meta-heuristic

```
Randomly generate N_i nodes to build the initial mesh
Evaluate all the mesh nodes
repeat
    for each node n in the mesh do
        Find its K-nearest neighbors
        n_best ← the best of its neighbors
        if n_best is better than n then
            Generate a new node by using function f
        end if
    end for
    for each initial node in the current mesh do
        Generate a new node by using function g
    end for
    repeat
        Select the most outward node of the mesh
        Generate a new node by using function h
    until MeshSize = N
    Select the best N_i nodes of the current mesh and set up the next mesh
until CurrentIteration = M
```

[1]. Regarding the second algorithm (TS-ACS-RST-FS), the factor r was set to 0.3 and the number of ants (m) is increased as $m_{i+1} = 2.1\, m_i$

Moving on to the particle based optimization, each individual was shaped as a binary vector whose length matches the number of attributes in the system. The algorithms portrayed in [4] were used for running tests. They are built upon the discrete PSO formulation devised by Kennedy and Eberhart in [10]. The parameters associated with the PSO-RST-FS (PSO + RST to feature selection) and TS-PSO-RST-FS (two-step) models were fixed as $c_1 = c_2 = 2$, $maxCycles = 120$ and $swarmSize = 21$. The inertia weight w varies as shown in (3) with $w_0 = 1.4$. For the latter algorithm, the parameter used for computing the quality of the classification in the first stage was set to 0.75 whereas the factor related to the number of cycles in each step was given a value of 0.3; all of these values were chosen after thorough experimentation among a set of possible values.

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 in the PSO-based models) and the computations lasted for 90 iterations.

Table 1 reports the experimental results obtained after applying the above methods over the BreastCancer, Heart and Dermatology data sets coming from the UCI Repository. For the sake of brevity, we omit the details about the input data. Each table entry holds the average number of reducts found, the average length (number of attributes) of the reducts in addition to the length

Table 1. The performance of some meta-heuristic approaches over three data sets

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%

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. The information depicted at the table is representative of the results achieved when using a larger number of data sets. They show up that the DMO model is one of the best in terms of number of reducts and shortest reduct found, which is consistent with the fact that it allows for a broader exploration over the search space. On the other hand, the two-step approaches are good at finding the shortest reducts.

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 lower the number of times expression (2) is computed, the more time-consuming 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 particles in PSO or, analogously, the number of nodes to be generated in DMO), the ACO-based models evaluate this function a greater number of times, viz $P \times Q \times k$ (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, the magnitude orders of the DMO (2503) and PSO-based models (2542 & 2568) are significantly lower when compared to the ones reported by the ACO methods (17222 & 13487).

4 Conclusions

In this paper, a novel evolutionary method coined as "Dynamic Mesh Optimization" (DMO) is presented. The performance of DMO in feature selection was compared with the ones reported in literature for several ACO- and PSO-based meta-heuristics. Based on the empirical evidence collected, we can conclude that the proposed model carries out an exploration of the search space in a similar way to the former method but consuming a computational time analogous to the latter one, which ends up in a good tradeoff between the different criteria considered during the experimental phase.

References

1. Bello, R., Nowé, A.: Using ACO and Rough Set Theory to Feature Selection. In: WSEAS Trans. on Information Science and Applications, pp. 512–517 (2005)
2. Bello, R., Nowé, A.: A Model Based on Ant Colony System and Rough Set Theory to Feature Selection. In: Genetic and Evolutionary Computation Conference (GECCO 2005), pp. 275–276 (2005)
3. Bello, R., Puris, A., Nowé, A., Martínez, Y., García, M.: Two-Step Ant Colony System to Solve the Feature Selection Problem. In: Martínez-Trinidad, J.F., Carrasco Ochoa, J.A., Kittler, J. (eds.) CIARP 2006. LNCS, vol. 4225, pp. 588–596. Springer, Heidelberg (2006)
4. Bello, R., Gómez, Y., Nowé, A., García, M.: Two-Step Particle Swarm Optimization to Solve the Feature Selection Problem. In: Proc. of 7th Int'l Conf. on Intelligent Systems Design and Applications, pp. 691–696. IEEE Computer Society, Washington (2007)
5. Dash, M., Liu, H.: Consistency-based Search in Feature Selection. Artificial Intelligence 151, 155–176 (2003)
6. Dorigo, M., Stützle, T.: Ant Colony Optimization. Cambridge University Press, New York (2005)
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Boston (1989)
8. Jensen, R., Shen, Q.: Finding Rough Set Reducts with Ant Colony Optimization. In: Proceedings of 6th Annual UK Workshop on Computational Intelligence, pp. 15–22 (2003)
9. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proc. of IEEE Int'l. Conf. on Neural Networks, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
10. Kennedy, J., Eberhart, R.C.: A Discrete Binary Version of the Particle Swarm Optimization Algorithm. In: Proc. of Int'l Conf. on Systems, Man and Cybernetics, pp. 4104–4109. IEEE Service Center, Piscataway (1997)
11. Komorowski, J., Polkowski, L., Skowron, A.: Rough Sets: a Tutorial. In: Pal, S., Skowron, A. (eds.) Rough-Fuzzy Hybridization: A New Trend in Decision-Making. Springer, New York (1999)
12. Kudo, M., Sklansky, J.: Comparison of Algorithms that Select Features for Pattern Classifiers. Pattern Recognition 33, 25–41 (2000)
13. Wang, X., Yang, Y., Peng, N., Teng, X.: Finding Minimal Rough Set Reducts with Particle Swarm Optimization. In: Slezak, D., Yao, J., Peters, J., Zizrko, W., Xiaohua, H. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3641, pp. 451–460. Springer, Heidelberg (2005)
14. Wang, X., Yang, J., Jensen, R., Liu, X.: Rough Set Feature Selection and Rule Induction for Prediction of Malignancy Degree in Brain Glioma. Computer methods and Programs in Biomedicine 83, 147–156 (2006)
15. Wang, X., Yang, J., Teng, X., Xia, W., Jensen, R.: Feature Selection Based on Rough Sets and Particle Swarm Optimization. Pattern Recognition Letters 28, 459–471 (2007)