RCMDE-GSD: Building Global Hierarchical Classifiers Using Differential Evolution for Predicting Gene Ontology Terms

Rafael Abud Menezes
Post-Graduate Program in Informatics - PPGIa
Pontificia Universidade Catolica do Parana - PUCPR
Brazil
rafael.abud@gmail.com

Julio Cesar Nievola
Post-Graduate Program in Informatics – PPGIa
Pontificia Universidade Catolica do Parana – PUCPR
Brazil
nievola@ppgia.pucpr.br

***Abstract*— In this work, we present a method to the building of a global hierarchical classification of the proteins functions for a structure of classes represented by a DAG (Directed Acyclic Graph), called RCMDE-GSD (Rule Construction Method Using Differential Evolution-Global Single DAG). Here, we compare RCMDE-GSD with hAntMiner, a method based on ACO (Ant Colony Optimization) algorithm and with HLCS, a method based on Learning Classifier Systems. In all the experiments, RCMDE-GSD outperformed or had similar results with at least one of the other two algorithms. Using Kruskal Wallis test, we conclude that the difference between the three algorithms was nonsignificant.**

Keywords*— *Hierarchical classifiers; DAG; classes.

I. Introduction (*Heading 1*)

The building of computational hierarchical classifiers by computers algorithms is actually one of the most researched areas in Bioinformatics [1]. Computers can perform the task of predicting the proteins' functions mainly because the hierarchical relationship among the classes can be represented by computational structures.

In order to ease the task of classification, the classes of the proteins are described in [2] [3] [4] [24]. The taxonomy can be represented by a DAG, as in [2] or by a tree, as in [3] [4] and [24]. A DAG (Directed Acyclic Graph) is a graph with directions and no cycles [25]. In fig. 1, the item (a) shows the function of a protein represented by the hierarchy in a tree and the item (b) shows the function of a protein represented by the hierarchy in a DAG (R stands for the hierarchy root).

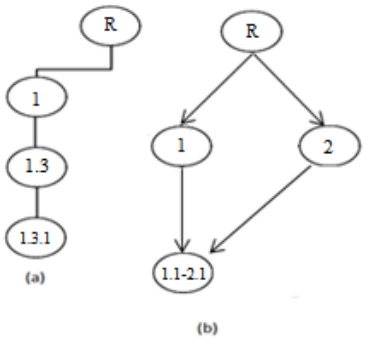

Figure 1 Representation of the function of a protein by a hierarchy in a tree (a) and in a DAG (b).

Developing an algorithm representing the classes' dependencies by a DAG is more complex than developing an algorithm representing the classes' dependencies by a tree.

The prediction of classes with dependency among them is more complex than the prediction of classes without it. This occurs mainly due to two reasons. The first is that the function to be predicted can belong to any level of the structure of classes. The second is that a function performed by an object in a given level of the structure of classes depends of the function that the object performs at prior levels. The task of classifying proteins is also complicated by the large number of possible functions of the proteins and the fact that each protein can perform more than one function in the same organism.

The difficulties cited and the fact that the discovery of the functions of proteins can help directly the research in health sciences encouraged the development of computational methods [5] [6][7] with the main objective of constructing hierarchical classifiers of proteins.

In this work, a method called RCMDE-GSD (Rule Construction Method using Differential Evolution-Global Single DAG), based on the Differential Evolution (DE)[10], is presented. The main objective of the method is the building of a single label global hierarchical classifier with the structure of classes represented by a DAG. DE was developed to work with optimization problems and continuous values and has been used to search the optimal solution in areas as Graphical Computing [11], electromagnetics [12] and economy [13]. Since it has been used successfully in problems with continuous values, the usage of DE in a problem with discrete values is our main motivation.

In section II the classifier is described. In section III, DE is described and in section IV the RCMDE-GSD is presented. In section V we describe the main works related with RCMDE-GSD. In section VI the experiments are described and in section VII the results of the experiments are presented. In section VIII the conclusions and future work are described.

II. CLASSIFIERS

Computationally, classifiers perform the task of predicting the functions of objects. Rules, semantic networks, decision trees, neural networks, can represent the knowledge that composes a classifier etc. The rules can be written in a IF THEN form. The main advantage of this type of rules is that they can

be read more easily by humans than the other types of rules and the other types of knowledge representation. In the task of classifying proteins, this advantage is important because biologists, during the decision-making process, will use the classifiers built probably.

Beyond the knowledge representation, there are other elements of the classifier to be defined. When the instances of the dataset used have classes related hierarchically with each other, the classifier used for classifying this object is called hierarchical.

A hierarchical classifier can be built into two approaches. The local approach follows one of three possible architectures. In the local per level approach, used in [1] and [5], the prediction of the function of an object is made using one level of the structure at a time. Furthermore, the local approach can be local per node and local per parent node.

In the local per node approach, one classifier is built for each level. In the local per parent node approach one classifier for each leaf node is built [21].

In the usage of local per level and local per node approaches, an object can be classified inconsistently. An example of an inconsistent classification can be seen in the usage of a dataset in which a new instance can belong to the classes 1.1, 1.4 or 2.1 (where the classes 1.1 and 1.4 are child classes of the class 1 and the class 2.1 is child class of the class 2). During the definition of the class in which the instance belongs to in the first level of the structure of classes (class 1 or 2), the classifier can define that the new instance belongs to the class 2. During the definition of the class in which the instance belongs in the second level of the structure of classes (classes 1.1, 1.4 or 2.1), the classifier can define that the new instance as belonging to class 1.4, making an inconsistent prediction, since the instance that belongs to the class 1.4 in the second level belongs to the class 1 in the first one. This inconsistency can be avoided with the top-down approach [21].

Nevertheless, in the usage of the three local approaches, if the classifier makes a mistake in a level, the error is propagated to the levels below [21]. The HEHRS method [8] tries to mitigate this disadvantage.

In the global approach, the prediction is made considering the entire structure of classes, as in [5], [6] and [7]. Due to it, the building of a global classifier is more complex than the building of local one [21].

In relation to the number of classes predicted by the rules, the classifier can be single label, as in [5] and [6], and multi-label, as in [9]. While the single label paradigm, classifiers predict one class for each instance, the multi-label classifier can predict more than one class for each instance.

III. DIFFERENTIAL EVOLUTION

The Differential Evolution is a little and simple mathematical model of the process of evolution [14]. DE was created with the main objective of solving optimization problems. The algorithm is composed by a population of N individuals (vectors) that evolve until G generations are formed. Each individual evolves through the mutation and crossover mechanisms.

In this way, in a generation g-1, for each individual I, one individual M_I is formed through mutation. Afterwards, from the crossover of I and M_I, an individual N_I is formed. So, in order to form the generation g (g >= 2), the better evaluated among I and N_I is chosen. This process is called selection [14].

In order to perform the mutation, a strategy has to be chosen. This choice depends on the problem being solved. According to [17], DE, as the majority of the population based algorithms, tries to explore every regions of the search space (exploration) and to find the optimal solution or some solution near from the optimal quickly (exploitation). Some strategies privilege the exploration of the state space, as DE/rand/1 (described in equation 1) and DE/rand/2 (described in equation 2).

$$M_I = I_1 + F * (I_2 - I_3) \quad (1)$$

where I_1 is the first randomly chosen individual, I_2 is the second randomly chosen individual, I_3 is the third randomly chosen individual and F is the scale factor. The value of F depends on the problem being solved. [14] suggests F = 0.4.

$$MiI = I_1 + F * (I_2 - I_3) + F * (I_4 - I_5) \quad (2)$$

where I_4 is the fourth randomly chosen individual and I_5 is the fifth randomly chosen individual.

The strategies that privilege the exploration of the state space are useful and advantageous in some problems, but disadvantageous in other ones. For example, trying to solve some kinds of problems, DE presents better results than other methods that have also the objective to solve optimization problems, as PSO (Particle Swarm Optimization) using DE/rand/1 strategy. Therefore, in other problems, DE can find the solution slowly or can stagnate in one region of the state space [15].

Some strategies were created with the objective of solving the mentioned problems caused by the strategies DE/rand/1 and DE/rand/2. These strategies use the best evaluated individual of the population. One of them is called DE/current-to-best/1 [16] or DE/target-to-best/1 [17] and is described in equation 3:

$$M_I = I + F * (B_I - I) + F * (I_1 - I_2) \quad (3)$$

where B_I is the better evaluated individual of the population.

In some situations, the DE/current-to-best/1 strategy also presents disadvantages, making the algorithm converge quickly to a local maximum [16] [17]. Some strategies have been proposed with the objective to solve this problem [16] [17].

After the mutation (in order to form N_I), I and M_I undergo crossover, that can be binomial or exponential [18]. Although the binomial crossover has been proposed in the original article [10], the exponential is more used [18]. The differences between them are discussed in [18].

A. DE in the building of classification rules

DE can be used to create rules into two approaches. These approaches are the same used in the GA´s. In the Michigan approach each individual represents one rule, while in the Pittsburgh, each individual represents a set of rules [21][22].

In the Michigan approach, the rules of the final generation can be used to compose the classifier. In this case, the algorithm can converge to a maximum local, generating similar rules and with no interaction between them [21][22]. This fact is undesirable for a classifier, since the rules must cover all the state space and consequently cover instances with any features.

The undesirable fact mentioned occurs since rules are constructed from the same instances of the dataset. Due to it, one possible solution is the creation of one rule at each execution of the algorithm [21] [22], with the instances covered by it been excluded from the dataset, as done in [9] and [19].

In the Pittsburgh approach, the redundancy of the individuals is not a problem, since the rules that compose each of these individuals are not redundant. This approach has the disadvantage of generating individuals syntactically [21] large [22] and complex [21] and with hard manipulation [22].

IV. RCMDE-GSD

The sections II and III describe the main subjects related with the understanding of the RCMDE-GSD. Thus, the RCMDE-GSD builds classifiers (described in the section II) using DE (described in the section III).

The main objective of RCMDE-GSD is to build a single label global hierarchical classifier with the structure of classes represented by a DAG. The classifiers are created by RCMDE-GSD and formed by rules, which are extracted from a dataset with labelled instances. In this work, the classifiers are composed by rules because they are more expressive and easy to understand than other forms of knowledge expression. These two facts are desirable because the classifiers built by RCMDE-GSD are formed in order to assist the biologists in decision-making.

Here, the classifier that creates the rules is called *Cr*. The *Cr* is initialized with no rules and filled by *Gr* (the rules generator, which is described in section IV.A) with one rule (*r*) at a time.

The following variables store information used in the method: *Neb* (number of instances at a given moment in the dataset), *Ecr* (number of examples covered by rule r), *Nrn* (number of rules generated with *Ecr* = 0) e *Max* (maximum number of *Nrn* used as a stopping criterion of the RCMDE-GSD).

When *r* is added to *Cr*, the instances covered by it are removed from the dataset. Thus, *Neb* is subtracted from *Ecr* (Fig. 2).

Thus, a rule is created considering only the instances not covered by the rules currently in *Cr*.

If a rule is added to the classifier and this rule has not covered any example in the dataset, a unit is added to *Nrn*. Since *Nrn* stores the number of consecutive times that a rule with *Ecr*=0 is generated, its value is set to zero when a rule with *Ecr* > 0 is added to *Cr* (Fig. 2).

There are two stopping criteria in RCMDE-GSD. The first is the number of examples in the dataset (Neb = 0) and the second is the number of consecutive rules generated with no instances covered by it (*Nrn* = *Max*).

Gr is an algorithm based on DE and described in the next section.

A. *The rules' generator*

In the first subsection there is a description of the variables that store information about Gr, while in Section IVA.2 there is a description of how the procedure works.

1) *The variables of Gr*

Gr is composed of a population of *N* individuals. According to this population, the parameters *J* (minimal number of gens by individual), *K* (maximal number of gens by individual), *G* (number of generations) can be defined. Whereas the exponential crossover is used, the variable *CrossR* (crossover rate) also can be defined.

```
BEGIN
    WHILE (Neb != 0 && Nrn != Max)
    {
        Gr creates rule r;
        r is added to Cr;
        IF (Ecr = 0)
        {
            Nrn = Nrn + 1;
        }
        ELSE
        {
            Neb = Neb + Ecr;
            Nrn= 0;
        }
    }
END
```

Figure 2 Pseudocode of RCMDE-GSD.

K's value must be less than or equal to the number of attributes of the instances of the dataset. In [17], it is suggested that the number of individuals should be ten times the number of genes of an individual. Since the number of genes varies from *J* to *K*, *N* is calculated as indicated by equation 4:

$$N = ((J + K)/2) * 10 \quad (4)$$

G is a free parameter that depends on the problem that is being solved.

Whereas each individual in *Gr* is represented by a rule in the format IF conditions THEN class, DE is used here in Michigan approach.

A_I and T_I are used for each individual *I*, with values randomly chosen inside the problem's domain. A_I ($1 <= A_I <= K$) represents the number of genes (antecedents) of an individual (rule) and T_I is the set of these genes. $T_I \in T_A$, where T_A is the set of attributes of the instances presents in the dataset.

Thus, the A_I*-1* first genes of an individual represent the antecedents of the rule, while A_I*-th* represents the consequent (the class of the rule represented by the individual).

2) *How Gr works*

The individuals of the first generation are randomly created within the problem domain. The genes of each individual assume the value of one of the correspondent attribute of a randomly chosen instance of the dataset. This is used to force the individual to cover at least one instance in the dataset. The

instance is randomly chosen in order to diversify the population of the first generation.

The next generations are formed as in the original DE. The mutation strategy used here is the same used in DEGL/SAW [17] with the exponential crossover.

The gene that represents the consequent (A_I-th) cannot be formed as in the original DE because in DE this operation is done through mathematical operations and in RCMDE-GSD the consequent is represented by a categorical class. Due to it, here the consequent of a rule is the class of one of the examples covered by it. A rule covers an example if the value of each antecedent is equal to the correspondent attribute of the instance.

B. The evaluation of the rules

In order to evaluate the rules (individuals), the instances that it covers are used.

Thus, the rules are evaluated according to the equation 5.

$$F = NCP/(1 + NCN) \quad (5)$$

where *NCP* is the number of instances covered by the rule in which the corrected class has at least one common ancestor with the predicted class. *NCN* is the number of instances covered by the rule in which the corrected class has no common ancestors with the predicted class.

NCN is incremented by 1 to avoid the denominator being 0.

V. RELATED WORKS

The method presented in [6] uses LCS (Learning Classifier Systems) algorithm in order to construct a single label global hierarchical classifier to the structure of classes represented by a tree and by a DAG. The algorithms described in [5] and [6] use the annotations described in [2] and [3] in their experiments.

Using ACO (Ant Colony Optimization) algorithm, the method presented in [9] build global hierarchical classifiers multi-label with the hierarchy of classes represented by a tree or a DAG. In the experiments, the annotations described in [4] and the ontology described in [24] are used.

Also based on ACO, the method hAnt-Miner is presented in [19]. The main objective of hAnt-Miner is to build a single label global hierarchical classifier with the hierarchy of classes represented by a DAG. hAnt-Miner uses one colony to build the antecedents and one colony to build the consequent. Both colonies have equal number of ants [19].

In [6], the authors have adapted the Classifiers Systems to a hierarchical classification problem. Thus, HLCS builds IF THEN rules based on the examples of the datasets, as the method described here.

These two methods were selected to be compared with RCMDE-GSD because they have the objective of constructing global hierarchical classifiers with the hierarchy of classes represented by a DAG, as RCMDE-GSD.

The method proposed in [17] tries to mitigate the drawbacks of the strategy DE/target-to-best, as the cited in section III. According to [17], DE/target-to-best favor exploitation only. In order to try to mitigate this drawback, [17] proposes a new algorithm, called DEGL. In DEGL, instead of creates one individual during the mutation, two individuals are created, one based on the local neighbourhood and other based on the global one. [17] proposes four versions of DEGL, one of them called DEGL/SAW, used for mutation in RCMDE-GSD.

VI. EXPERIMENTS

In the first experiment, RCMDE-GSD is compared with hAnt-Miner and with HLCS. In the second, RCMDE-GSD is compared only with hAnt-Miner.

The measures used to evaluate the algorithm in both experiments are the hierarchical precision (*hP*), the hierarchical recall (*hR*) and the *hF* measure. The three measures were proposed by [25].

The standard precision, recall and F measures are not suitable for hierarchical classification. The standard recall defines the proportion of examples that belong to the positive classes and were correctly classified. The standard precision defines the probability of a true prediction is correct. For hierarchical classifiers, according to [25], the hierarchical measures based on the notion of distance are broadly used; however, they have two main drawbacks. The first is that they can not be easily used with DAG's. The second is that misclassifications in different levels of the structure of classes are considered the same. It is an undesirable feature, because the classification of an object which belongs to the fifth level of the structure of classes, for example, is more difficult to be performed than a classification of an object which belongs to the first level. Therefore, a mistake in the first case should be less penalized by the classifier than a mistake in the first situation. Due to these drawbacks, the measures of hierarchical precision and hierarchical recall were proposed by [25]. In this case, the instance belongs to a class and to all of its ancestors [25].

In order to describe the *hR* and *hP* measures, two variables are used. *C* is composed by the predicted class and its ancestors. *Cx* is composed by the correct class (the class to which the example x belongs) and its ancestors.

The *hP* measure is calculated as described in (9):

$$hP = \left(\sum_{x=1}^{P} \frac{A}{B}\right)/P \quad (9)$$

,where *P* is the number of instances presents in the test dataset, *A* is the number of common elements between *C* and *Cx* and *B* is the number of elements presents in *C*.

The *hR* measure is calculated as described in (10):

$$hR = \left(\sum_{x=1}^{P} \frac{A}{E}\right)/P \quad (10)$$

,where *E* is the number of elements presents in *Cx*.

The *hF* measure is calculated as described in:

$$hF = (2 * hP * hR)/(hP + hR) \quad (11)$$

In section VI.A the datasets used in the two the experiments are described, while in section VI.B the values used in the variables of RCMDE-GSD are described.

A. The datasets

The datasets used in the experiments store information about ion channels, a transmembranar protein present in any living cell [19].

The datasets are divided in two groups (Table I). The preparation of the datasets is described in [19].

Table I Features of the datasets used in the experiments.

Group	Name	Nº of attributes	Nº of instances
ds1	AA	22	177
	Interpro	92	177
	IntAct	2096	147
ds2	AA	22	1631
	Interpro	155	478

The name of each dataset was chosen based on the information used to classify the instances. In the AA datasets, the instances were classified based on information about the composition of the aminoacids. In the Interpro datasets the instances were classified based on information about the Interpro pattern to classify the instances. In the IntAct dataset, the instances were classified based on information about IntAct pattern.

The first experiment was run with the AA and Interpro datasets. The second experiment was run with the IntAct dataset. The classes of the instances present in the dataset and their relationships are described using the Gene Ontology [2]. It is important the usage of the ontology during the classification due to the fact that the ancestors of all the classes are required to the calculation of the measures hP, hR and hF.

B. The values of the variables

In this work, the experiments were run with $J = 5$ and $K = 9$. These numbers were chosen since humans can understand approximately 7 pieces of information (chunks) [20]. Thus, the rules build by RCMDE-GSD have had 7±2 antecedents.

We have tried to avoid building of rules with less than 5 antecedents because in complexes problems, rule with less than 5 antecedents doesn't represent adequately the knowledge extracted from the dataset.

Whereas the strategy proposed by [17] was used in order to create the N_I individuals, all the variables related with the creation of N_I had the same values used in [17]. Thus, $N = 70$, neighbourhood radius = 7 (N / 10), *CrossR* = 0.9 and $F = \alpha = \beta$. Based on preliminaries results, $F = 0.2$ e $G = 100$.

VII. RESULTS

In table II, the results of the three algorithms in the first experiment are presented. The results which RCMDE-GSD that outperformed or had similar results with at least one of the two other algorithms are in bold.

RCMDE-GSD outperformed or had similar results with the other two algorithms in all the results presented. In the usage of AA dataset in ds1 group, RCMDE-GSD had similar results with hAntMiner independently of the measure used. Using the Interpro dataset of the ds1 group and the three measures RCMDE-GSD outperformed HLCS. Using *hR* measure RCMDE-GSD had similar results with hAntMiner.

In the usage of AA dataset in the ds2 group, RCMDE-GSD had similar result with hAntMiner when *hR* and *hF* measure were used and outperformed it when *hP* was used. Using the Interpro dataset of the ds2 group, in the usage of *hP* measure, RCMDE-GSD overcame HLCS and had similar result with hAntMiner. Using the *hF* measure, RCMDE-GSD outperformeds HLCS.

Table II Results of the experiments of RCMDE-GSD, hAnt-Miner and HLCS in the first experiment.

RCMDE-GSD				
Group	***Name***	***hP***	***hR***	***hF***
ds1	AA	**0.56±0.12**	**0.57±0.12**	**0.56±0.09**
	Interpro	**0.58±0.05**	**0.83±0.05**	**0.68±0.04**
ds2	AA	**0.62±0.04**	**0.63±0.04**	**0.62±0.03**
	Interpro	**0.81±0.02**	**0.58±0.02**	**0.68±0.02**
hAnt-Miner				
Group	***Name***	***hP***	***hR***	***hF***
ds1	AA	0.56±0.06	0.55±0.06	0.56±0.06
	Interpro	0.82±0.04	0.81±0.04	0.81±0.04
ds2	AA	0.63±0.02	0.59±0.02	0.61±0.01
	Interpro	0.83±0.01	0.75±0.01	0.79±0.01
HLCS				
Group	***Name***	***hP***	***hR***	***hF***
ds1	AA	0.84±0.02	0.64±0.03	0.73±0.02
	Interpro	0.54±0.03	0.65±0.02	0.59±0.02
ds2	AA	0.86±0.04	0.58±0.03	0.69±0.01
	Interpro	0.64±0.04	0.61±0.03	0.63±0.02

The difference between the results of the three method was statistically non-significant. The statistical test Kruskal-Wallis was used.

In table III, the results of RCMDE-GSD in the second experiment are presented.

Table III Results of the experiments of RCMDE-GSD and hAntMiner in the second experiment.

RCMDE-GSD				
Group	***Name***	***hP***	***hR***	***hF***
ds1	IntAct	0.67±0.03	**0.88±0.02**	**0.74±0.02**
hAntMiner				
Group	***Name***	***hP***	***hR***	***hF***
ds1	IntAct	0.77±0.04	0.54±0.03	0.63±0.03

In the second experiment, using the IntAct dataset RCMDE-GSD outperformed hAntMiner using *hR* and *hF* measures

VIII. CONCLUSIONS AND FUTURE WORKS

RCMDE-GSD showed satisfactory results, since it outperformed or had similar results with one of the two algorithms used for comparison in all the experiments. The results presented shown that RCMDE-GSD is an alternative to other methods for helping the decision-making process in the hierarchical classification of proteins with the hierarchy of classes represented by a DAG.

RCMDE-GSD showed satisfactory results mainly due to three facts. The first is that the method finds the most adapted rule ate each set of instances. The second is that DE creates rules in all the regions of the state space. The third is that the rules evaluate considering the correct prediction (when the rule covers and instance and its class have ancestors in common with the predicted class) and the wrong predictions (when the rule covers an instance and its class have no ancestors in common with the predicted class).

As future work, we suggest the building of the global version multi-label of RCMDE. We also suggest the building of the single label and multi-label local version of RCMDE.

ACKNOWLEDGMENT

The authors would like to thank CNPq and Fundação Araucaria for the partial grants that help develop this work.

REFERENCES

[1] Salama, K.M, Freitas, A.A., "ACO-Based bayesian network ensembles for the hierarchical classification of ageing-related proteins," In Proceedings of the 11th European conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, 2012, pp. 80-91.

[2] Ashburner, M. et al., "Gene Ontology: tool for the unification of biology," in Nature genetics, Vol. 25 number 1, 2000,pp 25-29.

[3] Webb, E. C., "Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes," in Academic Press, number 6, 1992.

[4] Horn, F. et al., "GPCRDB information system for G protein-coupled receptors," in Nucleic acids research, 2003, Vol. 31 number 1, pp. 294-297.

[5] Sillla Jr., C.N., Freitas,A.A., "A Global-Model Naive Bayes Approach to the Hierarchical Prediction of Protein Functions," in Proceeding ICDM '09 Proceedings of the 2009 Ninth IEEE International Conference on Data Mining,2009, pp. 992-997.

[6] Romão, L.M., Nievola, J.C., "Predicting GPCR and Enzymes Function with a Global Approach Based on LCS," In Proceedings of the 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE), 2012, pp.11-13.

[7] Romão, L.M., Nievola, J.C., "Hierarchical Classification of Gene Ontology with Learning Classifier Systems," in Advances in Artificial Intelligence, Lecture Notes in Computer Science, Vol. 7637, 2012,pp.120-129.

[8] Holden, N. and Freitas, A.A., "Hierarchical Classification of Protein Function with Ensembles of Rules and Particle Swarm Optimization," in Journal Soft Computing - A Fusion of Foundations, Methodologies and Applications - Special Issue on Evolutionary and Metaheuristics based Data Mining (EMBDM), 2008,Vol. 13(3), pp. 259, 272.

[9] Otero, F.E.B, Freitas, A.A., "Johnson, C.G., A Hierarchical Multi-Label Classification Ant Colony Algorithm for Protein Function Prediction," in Memetic Computing, 2010, Vol. 2 number 3, pp. 165-181.

[10] Storn, R. and Price, K., "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces", in International Computer Science Institute, Berkeley. CA, Tech. Rep., 1995, TR-95–012.

[11] Veroneses, L.P., Krohling, R.A., "Differential Evolution Algorithm on the GPU with CUDA", in IEEE World Congress on Computational Intelligence, 2010, pp. 18-23.

[12] Rocca, P., Massa, A., "Differential Evolution as applied to electromagnetics," in Advances, comparisons and applications, 6th European Conference on Antennas and Propagation, 2011, pp. 26-20.

[13] Zao, Z., Wang, J., Zhao, J., Su, Z., "Using a Grey model optimized by Differential Evolution algorithm to forecast the per capita annual net income of rural households in China," in Omega, 2012, Vol. 40 number 5, pp. 525-532.

[14] Feoktistov, V., Differential Evolution: In Search Solutions, Optimization And Its Applications, Springer, Vol. 5.

[15] Vesterstrom, J. and Thomsen, R., "A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems," in Evolutionary Computation, Vol. 2, 1987, pp. 1980-1987.

[16] Islam, S.M., Das, S., Ghosh, S., Roy, S. and Suganthan, P. N. "An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization," in Systems, Manufacturing and Cybernetics, Vol. 42 number 2, 2012, pp. 482-500.

[17] Das,S. and Abraham, A. and Chakraborty, U. K. and Konar, A., "Differential Evolution Using a Neighborhood-Based Mutation Operator," in Evolutionary Computation, Vol. 13 number 3, 2009, pp. 526-553.

[18] Zaharie, D., "A Comparative Analysis of Crossover Variants in Differential Evolution," in Proceedings of International Multiconference on Computer Science and Information Technology, 2007, pp. 171-181.

[19] Otero, F.E.B, Freitas, A.A., Johnson, C.G., "A Hierarchical Classification Ant Colony Algorithm for Predicting Gene Ontology Terms," in Evolutionary Computation. Machine Learning and Data Mining in Bioinformatics, Lecture Notes in Computer Science, Vol. 5483, 2009, pp. 68-79.

[20] Miller, G.A., "The Magical Number Seven, Plus or Minus Two Some Limit on Our Capacity for Processing Information, " in Psychological Review, 1995,Vol. 101 number 2, pp. 343-352.

[21] Freitas, A.A., "A survey of evolutionary algorithms for data mining and knowledge discovery," in Advances in evolutionary computing, 2011, pp.819-845.

[22] Giordana A., Neri F., "Search Intensive Concept Induction," In Evolutionary Computation, 1995,Vol. 3 number 4, pp. 375-416.

[23] Ruepp A, et al.," The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes," in Nucleic Acid Research, 2004, Vol. 32 number 18, pp. 5539–5545

[24] Kiritchenko, S. Matwin, S. and F. Famili, "Functional annotation of genes using hierarchical text categorization," in Proc. of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology, 2005.

[25] Wang, L. "Directed Acyclic Graph," in Encyclopedia of Systems Biology, pp 274-274, 2013.