A Hybrid Particle Swarm Optimization Applied for Multi-Label Classification Problem

Tiago Amador Coelho
Department of Mathematical Sciences
Universidade Estadual de Feira de Santana
tiago@uefs.br

Ahmed Ali Abdalla Esmin
Department of Computer Sciences
Universidade Federal de Lavras
ahmed@dcc.ufla.br

Wagner Meira Júnior
Department of Computer Sciences
Universidade Federal de Minas Gerais
meira@dcc.ufmg.br

***Abstract*—Multi-label learning first arose in the context of text categorization, where each document may belong to several classes simultaneously. In this paper, we propose a hybrid approach, called Multi Label K-Nearest Michigan Particle Swarm Optimization (ML-KMPSO). It is based on two strategies. The first strategy is the Michigan Particle Swarm Optimization (MPSO), which breaks the multi-label classification task into several binary classification problems. The second strategy is ML-KNN, which is complementary and takes into account the correlations among classes. We evaluated the performance of ML-KMPSO using two real-world benchmark datasets: Yeast gene functional analysis and natural scene classification. The experimental results show that ML-KMPSO produced results that match or outperform well-established multi-label learning algorithms.**

***Index Terms*—Multi-label classification, Particle swarm optimization, Data mining**

I. Introduction

Real applications such as semantic scene classification, protein function classification, text categorization, and music categorization, among others, are increasingly require multi-label classification methods. Problems with two or more classes are considered special cases of multi-label classification, where each instance has only a single label.

In a classification problem, the examples are associated with a single label Y of a label set $\mathcal{Y}$ and $|\mathcal{Y}| > 1$. When $|\mathcal{Y}| = 2$, the problem is called a binary classification problem, and when $|\mathcal{Y}| > 2$, it is called a multi-class classification problem. However, there are several real problems where data may belong to more than one class simultaneously ($|Y| \geq 2$) and the classes are not disjoint, in this case, the problem is called a multi-label classification (MLC) problem. This generality of multi-label problems makes it more difficult to solve.

An intuitive method to solve this problem is to decompose it into multiple independent binary classification problems, but such method does not consider the correlations between the different labels of each instance and the expressive power of such a system may be weak. Another method to solve this problem is to adapt single label classification algorithms for multi-label classification problems, such as multi-label decision trees [2], multi-label kernel methods [3], multi-label neural networks [4] and Multi-Label k-Nearest Neighbor (ML-KNN) [5]. In this case, the results are still not good enough and this issue remains open and is attracted much attention from researchers.

In this paper, we propose a hybrid approach, ML-KMPSO. It is based on two strategies. The first strategy is the Michigan Particle Swarm Optimization (MPSO), which breaks the multi-label classification task into several binary classification problems, but it does not take into account the correlations among the various classes [6], [7]. The second strategy is ML-KNN [5], which is complementary and takes into account the correlations among classes.

In practice ML-KMPSO works as follows. First, for each test instance, it identifies its KNNs in the training set and calculates their prior probabilities. Then, the classification model is generated by MPSO and a set of expert classification particles are identified. Based on statistical information gained from the labeled sets of these neighbor instances, i.e., the number of neighbor instances including the MPSO classification particles belonging to each possible class, maximum a posteriori (MAP) [8] principle is utilized to determine the label set for that test instance (the classification result).

The performance of the ML-KMPSO is tested through two different multi-label learning problems, more specifically Yeast gene functional analysis and natural scene classification. The experimental results show that ML-KMPSO produced results that match or outperform well-established multi-label learning algorithms.

The rest of the paper is organized as follows: Section 2 discusses the multi-label classification problem, Section 3 provides an overview of PSO, Section 4 presents the PSO and MPSO classification algorithms, Section 5 describes the proposed approach, Section 6 the experiments are presented and discussed, and Finally, the conclusions and future works are given in Section 7.

II. Multi-Label Classification (MLC)

Research on multi-label classification was initially motivated by the challenge present in text categorization, as many labels may be associated with the same document [8].

In a multi-label problem, each instance is associated with a set of labels Y, where Y $\subseteq \mathcal{Y}$, being $\mathcal{Y}$ a set of labels.

In [9], the classification problem is defined as follows: Since $\mathcal{X}$ a training set, $\mathcal{Y} = \{1, 2, ..., k\}$ a label set. Given a training set of the form $\langle x_i, Y_i \rangle$, $x_i \in \mathcal{X}$, $Y_i \in 2^{|\mathcal{Y}|}$, where $2^{|\mathcal{Y}|}$ are all possible combinations of $\mathcal{Y}$. The goal of the learning system is to output a multi-label classifier $h : \mathcal{X} \rightarrow 2^{|\mathcal{Y}|}$ that maximizes a function $f(x)$, such that $f(x)$ return values of $2^{|\mathcal{Y}|}$ with the smallest error. The difficulty of defining the error in multi-label is that various combinations of labels are possible.

In most cases, the multi-label approach induces an ordering of the possible labels of a given instance according to $f(x, l_n)$. thus, we can formally define $rank_f(x, l)$ as $rank_f$ of label l for instance x, such that if $f(x, l_1) \leq f(x, l_2)$ then $rank_f(x, l_1) \leq rank_f(x, l_2)$.

Both [10] and [8] grouped the methods to solve the multi-label problem into two groups:

- The first category is more intuitive, is decomposition of the problem into multiple independent binary classification problems;
- The second category is algorithm adaptation for multi-label classification problems.

Figure 1 shows the methods that are found in the literature for the multi-label problem as organized hierarchically by Carvalho et.al [11]. Next we discuss some of these methods.

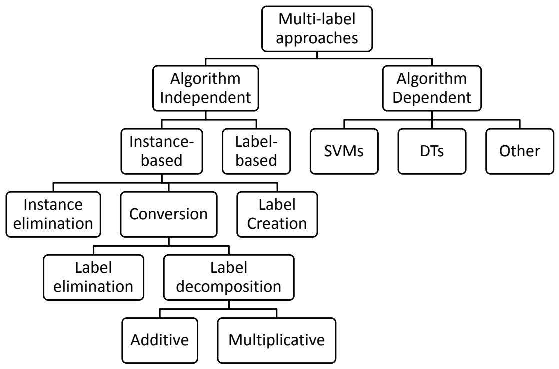

Fig. 1. Methods used in Multi-Label Classification Problems [11]

One well-known approach for multi-label learner is BoosTexter. It was proposed by Schapire and Singer [1] and maintains a set of weights on every instance-label pairs of the training set and those pairs that are hard/easy to predict correctly will get incrementally higher/lower weights.

The multi-label kernel method RANK-SVM proposed by Elisseeff and Weston [3], maximizes the sum of the margins for all categories simultaneously, ranking the relevant categories of each instance in the training set higher than the irrelevant categories.

Zhang and Zhou [5] proposed an instance-based multi-label lazy learning approach named ML-KNN, derived from the traditional KNN algorithm. In detail, first it identifies, for each unseen instance, the K nearest neighbors in the training set. Then, based on statistical information gained from the labeled sets of these neighbor instances, it employs the maximum a posteriori (MAP) principle to determine the label set for the test instance.

We propose a hybrid approach to solve the MLC problems, which puts together the algorithm adaptation and the algorithm independent class as seen in Figure 2.

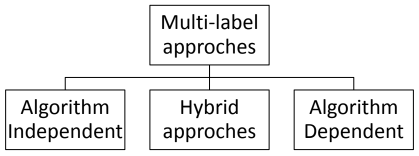

Fig. 2. New Configuration of Methods used in Multi-Label Classification Problems

III. Particle Swarm Optimization Algorithm

The Particle Swarm Optimization algorithm (PSO) is a population-based optimization method first proposed by Kennedy and Eberhart [12]. The PSO technique finds the optimal solution using a population of particles. Each particle represents a candidate solution to the problem. PSO basically simulates bird flocking in a two-dimension space, where each particle represents a candidate solution to the problem [12], [13]. We may define the PSO as follows:

- Each individual particle i has the following properties: A current position in search space, x_i, a current velocity, v_i, and a personal best position in search space, y_i.
- The personal best position, y_i, corresponds to the position in search space where particle i presented the smallest error as determined by the objective function f, assuming a minimization task.
- The global best position denoted by $\breve{y}$ represents the position yielding the lowest error amongst all the y_i.

During each iteration of the algorithm, every particle in the swarm is updated using equations 1 and 2. The velocity update step is:

$$\begin{aligned} v_{i,j}(t+1) &= wv_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + \\ & c_2 r_{2,j}(t)[\breve{y}_j(t) - x_{i,j}(t)] \end{aligned} \tag{1}$$

where c_1 and c_2 are two positive constants, r_1 and r_2 are two random numbers within the range [0,l], and w is the inertia weight.

The current position of the particle is updated to obtain its next position:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2}$$

Equation 1 consists of three parts. The first part is the current speed of the particle, which shows its current state; the second part is the cognition term, which expresses the ”best experience” of the particle itself; the third part is called social term, and reflects the information sharing among the swarm. These three parts together determine the particle's space searching ability. The first part has the ability to balance the whole and search a local part. The second part causes the swarm to have a strong ability to search the whole and avoid local minimum. The third part reflects the information sharing among the particles, leading the particles towards known good solutions. Under the influence of these three parts, the particles may reach an effective and best position.

Equations 3 and 4 define how the personal and global best values are updated at time t, respectively. It is assumed that the swarm consists of s particles (thus $i \in 1..s$).

$$y_i(t+1) = \begin{cases} y_i(t) & f(y_i(t) \leq f(x_i(t+1))) \\ x_i(t+1) & f(y_i(t) > f(x_i(t+1))) \end{cases} \tag{3}$$

$$\breve{y}(t) = min(f(y), f(\breve{y}(t))) \tag{4}$$
$$y \in y_0(t), y_1(t), ..., y_s(t)$$

The algorithm consists of executing iteratively the update equations presented. The pseudo-code for the basic PSO algorithm:

```
1) Initialize swarm
2) While maximum iterations or minimum error criteria is not
   attained
     a) For each particle
          i) Update the particle velocity (Eq. 1)
         ii) Update the particle position (Eq. 2)
        iii) Update the particle best known position (Eq. 3)
         iv) Update the swarm best known position (Eq.4 )
```

Fig. 3. PSO pseudo code

In the next section we describe the adaptation of PSO for the classification problem.

IV. The PSO and MPSO Classification Algorithm

Recently, some work has already been done to adapt PSO to classification problems and most of it concerns rule-based classifiers. In [14], PSO is used to extract induction rules to classify data; the standard PSO algorithm is executed several times, generating a single rule each time and using only unclassified patterns for subsequent iterations. In [15], [16] and [17], the standard PSO is used for rule extraction for only discrete classification problems. In [18] the PSO is used to generate clusters.

In a standard PSO approach, a potential solution is encoded in each particle. The information that has to be encoded is the set of prototypes and the prototype classes. There is another method, called Michigan PSO (MPSO) that uses the Michigan approach. The method MPSO results are quite competitive compared to other methods. In the MPSO approach, a member of the population does not encode the whole solution to the problem, but only part of it. The whole swarm is the potential solution to the problem [6], [14]. The advantages of the Michigan approach against the standard PSO approach are: a) scalability and computational cost, as particles have much lower dimension; and b) flexibility and reduced assumptions, as the number of prototypes in the solution is not fixed [6], [14].

The pseudo code of the MPSO method is shown in Figure 4 and described in detail in [6], [14]. The equation changes in comparison with the standard PSO are the introduction of a repulsion force and the use of a dynamic definition of the neighborhood of a particle. When moving, each particle selects another one from what it is called a ”non-competing” set as a leader for attraction, and a second one from a competing set as a leader for repulsion.

```
1) Load training patterns
2) Initialize swarm
3) Insert N particles of each class in the training patterns
4) Until max. number of iterations reached or success rate is 100
     a) Calculate which particles are in the competing and non-
        competing sets of particles for every class
     b) For each particle
          i) Calculate Local Fitness
         ii) Calculate Social Adaptability Factor
        iii) Find the closest particle in the non-competing set for
             the particle class (attraction center)
         iv) Find the closest particle in the competing set for the
             particle class (repulsion center).
          v) Calculate the particle's next position based on its
             previous velocity, previous best position, attraction
             center and repulsion center
     c) Move the particles
     d) Assign classes to the patterns in the training set using
        the nearest particle
     e) Evaluate the swarm classification success
     f) If the swarm gives the best success so far, record the
        current positions of the particles as ”current best swarm”
5) Delete, from the best swarm found so far, the particles that can
   be removed without a reduction in the classification success
   value.
6) Evaluate the swarm classification success over the validation
   set and report result
```

Fig. 4. MPSO pseudo code

During each iteration, both neighborhoods are defined dynamically and take into account the particles classes. In this case, particles cooperate with particles from different classes and compete with particles from the same class. The MPSO uses the concept of local fitness. A single particle is measured as ”good” if it classifies correctly patterns in its surroundings. The particle fitness calculation does not take into account how the rest of the patterns are classified [6].

The MPSO is very efficient generating a swarm of specialized particles in pattern-recognition and is may be easily adapted to different classes of problems [6], [7]. In this paper, we present an adaptation of MPSO to solve the MLC problem, called the ML-KMPSO. As the name indicates, ML-KMPSO is derived from both ML-KNN and MPSO.

V. ML-KMPSO

The proposed Multi Label K-Nearest Michigan Particle Swarm Optimization (ML-KMPSO) is based on two strategies. The first strategy is the Michigan Particle Swarm Optimization (MPSO), which breaks the multi-label classification task into several binary classification problems. The second strategy is ML-KNN, which is complementary and takes into account the correlations among various classes.

Figure 5 provides the pseudo code for the ML-KMPSO algorithm and it is works as follows. Given a training data set, a test data set, the number of neighbors (k) and the quantity of particles to be produced for each label (Q). First, it calculates the prior probabilities from the training set, then it generates the classification model using MPSO (step 2) and identifies a set of expert classification particles. Note that in step 5 of MPSO (see Figure 4), unused particles are marked for removal from the solution (swarm) starting with the particle with the worse local fitness value. If this action doesn't reduce the swarm classification accuracy, the particle is definitely removed. Finally, according to statistical information gained from the labeled sets of these neighbour instances, i.e., the number of neighbor instances including the MPSO classification particles belonging to each possible class, it uses the maximum a posteriori (MAP) [8] principle to determine the label set for the test instance, which becomes the classification result.

```
1) Compute the prior probabilities of all labels using training set
2) Compute swarm with MPSO (see Figure 4)
3) Merge expert classification particles with training set
4) Compute the posterior probabilities of all labels using k
   neighbours using merged data
5) Determine the label set for each test instance using MAP
```

Fig. 5. Summary of ML-KMPSO pseudo code

In the next section we evaluate the performance of the ML-KMPSO using two different multi-label learning problems: Yeast gene functional analysis [3] and natural scene classification [19]. The experimental results are compared with ML-KNN [8] [5], BoosTexter [1] and Rank-SVM [3] methods.

VI. Experimental Evaluation

A. Datasets

Two commonly-used multi-label datasets where used in the experiments: the Yeast dataset [3] and the Scene dataset [19].

Yeast [3] is a biological dataset formed by micro-array expression data and phylogenetic profiles for 2417 genes, where each gene is associated with a set of 14 functional classes (Figure 6) of the Yeast *Saccharomyces cerevisiae*. Elisseeff and Weston [3] preprocessed the Yeast data set and, in order to make it easer, only the known structures of the functional classes are used.

Scene [19] is a natural scene image dataset where each image is categorized into semantic classes such as beach, sunset, foliage, field, mountain, and urban. Table I shows the detailed description of the images in the dataset.

Table II shows some statistics about the datasets, such as the numbers of examples in the training and test sets, number of numeric attributes, label cardinality, and label density. Label cardinality is the average number of labels per example, while label density is the same number divided by the number of possibles labels.

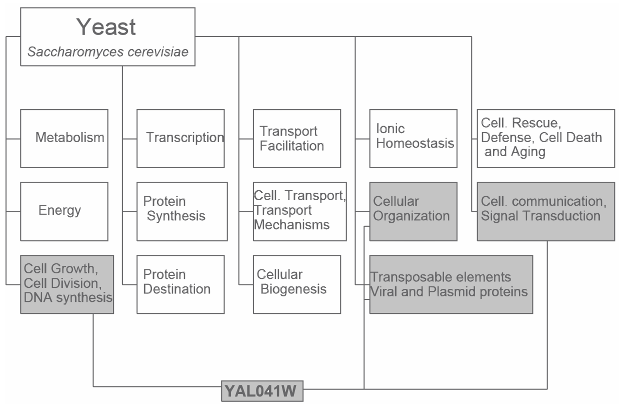
Yeast
Saccharomyces cerevisiae
Metabolism
Transcription
Transport Facilitation
Ionic Homeostasis
Cell. Rescue, Defense, Cell Death and Aging
Energy
Protein Synthesis
Cell. Transport, Transport Mechanisms
Cellular Organization
Cell. communication, Signal Transduction
Cell Growth, Cell Division, DNA synthesis
Protein Destination
Cellular Biogenesis
Transposable elements Viral and Plasmid proteins
YAL041W

Fig. 6. First level of the hierarchy of the gene functional classes. There are 14 classes. [3]

TABLE I
Summary of the natural scene image data set

Class	Total
Desert	340
Mountains	268
Sea	341
Sunset	216
Trees	378
Desert + mountains	19
Desert + sea	5
Desert + sunset	21
Desert + trees	20
Mountains + sea	38
Mountains + sunset	19
Mountains + trees	106
Sea + sunset	172
Sea + trees	14
Sunset + trees	28
Desert + mountains + sunset	1
Desert + sunset + trees	3
Mountains + sea + trees	6
Mountains + sunset + trees	1
Sea + sunset + trees	4
Total	2000

TABLE II
Multi-label Statistics for the Datasets Used

		Attributes		Distinct	Label	Label
Dataset	Examples	Numeric	Labels	Subsets	Cardinality	Density
Yeast	2417	103	14	198	4.327	0.302
Scene	2000	294	5	15	1.24	0.248

B. Evaluation Methodology

For the sake of evaluating the performance of the learning methods, we choose commonly-used criteria in multi-label classification, as proposed by Schapire and Singer in [1]. Given a set of multi-label instances $S = \{(x_1, Y_1), ..., (x_P, Y_P)\}$, a learned ranking function $f(\cdot,\cdot)$ and the corresponding multi-label classier $h(\cdot)$:

(A) *HammingLoss* evaluates how many times an instance-label pair is mispredicted.

$$HL_S(h) = \frac{1}{P}\sum_{i=1}^{P}\frac{|h(x_i)\triangle Y_i|}{|Y|} \tag{5}$$

where $\triangle$ stands for the symmetric difference between two sets, predict labels ($h(x_i)$) and correct labels (Y_i).

(B) *One-error* evaluates how many times the top-ranked label was not in the set of possible labels.

$$OE_S(f) = \frac{1}{P}\sum_{i=1}^{P}\left[[\texttt{arg max } f(x_i, y)] \notin Y_i\right] \quad (6)$$

where $y \in \mathcal{Y}$. The term $[\texttt{arg max } f(x_i, y)]$ represents the maximum value of the function $f()$. The smaller the value of $One - Error_S(f)$, the better the performance.

(C) *Coverage* is defined as the distance to cover all possible labels assigned to a sample x. It is loosely related to precision at the level of perfect recall.

$$Co_S(f) = \frac{1}{P}\sum_{i=1}^{P}\texttt{max } rank_f(x_i, y) - 1 \quad (7)$$

where $y \in \mathcal{Y}$. The smaller the value of coverage is, the better the performance is.

(D) *Ranking Loss* evaluates the average fraction of label pairs that are reversely ordered for the instance

$$RL_S(f) = \frac{1}{P}\sum_{i=1}^{P}\frac{1}{|Y_i||\overline{Y}_i|}|(y_1, y_2)|f(x_i, y_1) \not< f(x_i, y_2), \quad (8)$$
$$(y_1, y_2) \in Y_i \times \overline{Y}_i$$

(E) *Average Precision* is the average precision taken for all the possible labels and it can evaluate algorithms as a whole. It measures the average fraction of labels ranked above a particular label $y \in Y_i$ which is actually in Y_i. The best performance is reached when average precision is equal to 1. The larger the value of average precision is, the better performance is.

$$AP_S(f) = \frac{1}{P}\sum_{i=1}^{P}\frac{1}{|Y_i|} \times \quad (9)$$
$$\sum_{y \in Y_i}\frac{|\{y'|rank_f(x_i, y') \not< rank_f(x_i, y'), y' \in Y_i\}|}{rank_f(x_i, y)}$$

C. Experimental Results

To ensure more reliable results, all experiments were performed using 10-fold cross-validation. In this validation method, the set of examples is divided into 10 folds, each iteration of the algorithm uses nine folds for training and one fold for testing.

Table III presents the comparison of ML-KPMSO with others algorithms for the multi-label classification problem (ML-KNN, and BoosTexter Rank-SVM) for the dataset Yeast. The result is presented using all the evaluation metrics that were mentioned in subsection VI-B. For each evaluation criterion, down arrow indicates that the smaller the better while up arrow indicates that the higher value is the better.

As seen in this table, the ML-KMPSO performed better in all the five metrics when compared to BoosTexter and Rank-SVM. Considering ML-KNN, it was better just for the one-meter error, and was similar for most of the remaining metrics.

TABLE III
RESULT ON YEAST DATA

Metrics	Algorithms			
	ML-KNN	BoosTexter	Rank-SVM	ML-KMPSO
HL $\downarrow$	0.194 ± 0.010	0.220 ± 0.011	0.207 ± 0.013	0.198 ± 0.008
OE $\downarrow$	0.230 ± 0.030	0.278 ± 0.034	0.243 ± 0.039	0.225 ± 0.045
Co $\downarrow$	6.275 ± 0.240	6.550 ± 0.243	7.090 ± 0.503	6.355 ± 0.232
RL $\downarrow$	0.167 ± 0.016	0.186 ± 0.015	0.195 ± 0.021	0.172 ± 0.009
AP $\uparrow$	0.765 ± 0.021	0.737 ± 0.022	0.749 ± 0.026	0.762 ± 0.012

The ranking of the best methods for each metric is shown in Table IV.

TABLE IV
PERFORMANCE BETWEEN EACH MULTI-LABEL LEARNING ALGORITHM ON THE YEAST DATA

Metrics	Algorithms
HL	ML-KNN > ML-KMPSO > Rank-SVM > BootTexter
OE	ML-KMPSO > ML-KNN > Rank-SVM > BootTexter
Co	ML-KNN > ML-KMPSO > BootTexter > Rank-SVM
RL	ML-KNN > ML-KMPSO > BootTexter > Rank-SVM
AP	ML-KNN > ML-KMPSO > Rank-SVM > BootTexter

The results for the Scene dataset are in Table V. ML-KPMSO was very promising, being the best considering the Hamming Loss and Coverage metrics, second in One-Error and third in Ranking Loss and Average Precision. The improvements associated with Coverage (Co) and Hamming Loss (HL) are a consequence of the low cardinality and density of the dataset, which favors MPSO that comprises several expert classifiers and is able to predict accurately a larger number of tuples (samples x_i - labels Y_i) On the other hand, Rank-SVM provided the worst results.

TABLE V
RESULT ON SCENE DATA

Metrics	Algorithms			
	ML-KNN	BoosTexter	Rank-SVM	ML-KMPSO
HL $\downarrow$	0.169 ± 0.016	0.179 ± 0.015	0.253 ± 0.055	0.153 ± 0.010
OE $\downarrow$	0.300 ± 0.036	0.311 ± 0.041	0.491 ± 0.135	0.311 ± 0.036
Co $\downarrow$	0.939 ± 0.100	0.939 ± 0.092	1.382 ± 0.381	0.872 ± 0.274
RL $\downarrow$	0.168 ± 0.024	0.168 ± 0.020	0.278 ± 0.096	0.173 ± 0.013
AP $\uparrow$	0.803 ± 0.027	0.798 ± 0.024	0.682 ± 0.093	0.796 ± 0.018

TABLE VI
PERFORMANCE BETWEEN EACH MULTI-LABEL LEARNING ALGORITHM ON THE SCENE DATA

Metrics	Algorithms
HL	ML-KMPSO > ML-KNN > BootTexter > Rank-SVM
OE	ML-KNN > ML-KMPSO > BootTexter > Rank-SVM
Co	ML-KMPSO > ML-KNN > BootTexter > Rank-SVM
RL	ML-KNN > BootTexter > ML-KMPSO > Rank-SVM
AP	ML-KNN > BootTexter > ML-KMPSO > Rank-SVM

The experimental results show that the proposed algorithm outperforms other approaches such as Rank-SVM , BootTexter and matches ML-KNN. We believe that further calibration to adjust the parameters of MPSO in the context of the proposed approach should enable better results than the ML-KNN.

VII. Conclusion

This paper presented a hybrid approach to solve MLC problems called ML-KMPSO, which is an adaptation of the Michigan Particle Swarm Optimization for Multi-label Problem. The paper has compared the ML-KMPSO to others well-established multi-label methods (ML-KNN, RankSVN and BoosTexter) and the experimental results using two real-world dataset empirically show that our proposal outperforms or at least matches those methods, indicating that it is very promising method.

As future work we plan to perform more experiments towards adjusting parameters and to fully evaluate the efficacy of ML-KMPSO.

Acknowledgment

We would like to thank CNPq and FAPEMIG (Brazilian agencies) for partial financial support.

References

[1] R. E. Schapire and Y. Singer, "BoosTexter: A Boosting-based System for Text Categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.1666

[2] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[3] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *In Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 681–687.

[4] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 1338–1351, 2006.

[5] M. Zhang and Z. Zhou, "A k-nearest neighbor based algorithm for multi-label classification," in *GrC*, 2005, pp. 718–721.

[6] A. Cervantes, I. M. Galván, and P. I. Viñuela, "Michigan particle swarm optimization for prototype reduction in classification problems," *New Generation Comput.*, vol. 27, no. 3, pp. 239–257, 2009.

[7] A. Cervantes, I. M. Galván, and P. Isasi, "Ampso: a new particle swarm method for nearest neighborhood classification," *Trans. Sys. Man Cyber. Part B*, vol. 39, pp. 1082–1091, October 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1656796.1656797

[8] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, July 2007. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2006.12.019

[9] T. Li, C. Zhang, and S. Zhu, "Empirical studies on multi-label classification," in *ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 86–92.

[10] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*, 2008.

[11] A. de Carvalho and A. Freitas, "A tutorial on multi-label classification techniques," in *Foundations of Computational Intelligence Volume 5*, ser. Studies in Computational Intelligence, A. Abraham, A.-E. Hassanien, and V. Snael, Eds. Springer Berlin / Heidelberg, 2009, vol. 205, pp. 177–195.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, 1995, pp. 1942–1948 vol.4.

[13] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," 1995, pp. 39–43. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=494215

[14] T. Sousa, A. Silva, and A. Neves, "Particle swarm based data mining algorithms for classification tasks," *Parallel Comput.*, vol. 30, pp. 767–783, May 2004. [Online]. Available: http://portal.acm.org/citation.cfm?id=1016308.1016321

[15] Z. Wang, X. Sun, and D. Zhang, "Classification rule mining based on particle swarm optimization," in *RSKT*, 2006, pp. 436–441.

[16] A. A. A. Esmin, R. A. Coelho, and S. Matwin, "A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data," 2013. [Online]. Available: http://dx.doi.org/10.1007/s10462-013-9400-4

[17] Z. Wang, X. Sun, and D. Zhang, "A pso-based classification rule mining algorithm," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, ser. Lecture Notes in Computer Science, D.-S. Huang, L. Heutte, and M. Loog, Eds. Springer Berlin / Heidelberg, 2007, vol. 4682, pp. 377–384.

[18] A. A. A. Esmin, D. L. Pereira, and F. De Araujo, "Study of different approach to clustering data by using the particle swarm optimization algorithm," in *Evolutionary Computation 2008 CEC 2008 IEEE World Congress on Computational Intelligence IEEE Congress on*, 2008, pp. 1817–1822.

[19] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, September 2004.