# Evolutionary Multilabel Hyper-Heuristic Design

Alejandro Rosales-Pérez*, Andrés E. Gutiérrez-Rodríguez*, José C. Ortiz-Bayliss*,
Hugo Terashima-Marín*, Carlos A. Coello Coello†
* Tecnologico de Monterrey, School of Engineering and Science
Monterrey, Nuevo Leon, 64849, Mexico
Email: {arosalesp, aegr82, jcobayliss, terashima}@itesm.mx
† CINVESTAV-IPN, Evolutionary Computation Group
Mexico City, 07360, Mexico
Email: ccoello@cs.cinvestav.mx

***Abstract*—Nowadays, heuristics represent a commonly used alternative to solve complex optimization problems. This, however, has given rise to the problem of choosing the most effective heuristic for a given problem. In recent years, one of the most used strategies for this task has been the hyper-heuristics, which aim at selecting/generating heuristics to solve a wide range of optimization problems. Most of the existing selection hyper-heuristics attempt to recommend only one heuristic for a given instance. However, for some classes of problems, more than one heuristic can be suitable. With this premise, in this paper, we address this issue through an evolutionary multilabel learning approach for building hyper-heuristics. Unlike traditional approaches, in the multilabel formulation, the result could not be a single recommendation, but a set of potential heuristics. Due to the fact that cooperative coevolutionary algorithms allow us to divide the problem into several subproblems, it results in a natural approach for dealing with multilabel classification. The proposed cooperative coevolutionary multilabel approach aims at choosing the most relevant patterns for each heuristic. For the experimental study included in this paper, we have used a set of constraint satisfaction problems as our study case. Our experimental results suggest that the proposed method is able to generate accurate hyper-heuristics that outperform reference methods.**

## I. Introduction

A number of heuristic methods have been proposed for dealing with complex optimization problems. In spite of their success, to date there is no single method that has the best performance on all problems, and this is, indeed, not possible in general, according to the well-known *No Free Lunch Theorem* [1]. Thus, when facing an optimization problem, the user has to choose the heuristic that best solves it. This, however, can be a challenging task not only due to the wide umbrella of possibilities, but also due to the lack of guidelines that indicate in which cases a specific heuristic is expected to be better than others. This fact has been the main motivation for developing a research area devoted to automating the design of algorithms.

In the field o f o ptimization, o ne o f t he m ost recent trends for automating the design of algorithms is the one that considers using hyper-heuristics. The term *hyper-heuristic* was initially used to describe heuristics for choosing heuristics [2], [3]. Nowadays, this definition also includes the automatic generation of heuristics. In this sense, hyper-heuristics operate at a higher level of generality by working with the low level heuristics rather than working directly with the solution of the problem [2], [4], [5].

Our research focuses on hyper-heuristics for heuristic selection. In this approach, the strategy can access a pool of available heuristics and selectively applies the one that reaches a peak performance on the problem at hand. Traditionally, hyper-heuristic approaches rely on machine learning techniques to learn a model of the problem, which in essence constitutes the hyper-heuristic. For example, the last years have witnessed how various machine learning strategies –such as lifelong learning [6], reinforcement learning [7] and artificial neural networks [8]– have been successfully applied to produce selection hyper-heuristics for different problem domains. One way of doing this learning is through the so-called supervised classification. The idea is that, given a set of training instances, we first identify the best heuristic for each instance and later, a hyper-heuristic is built with the information of the instance and the associated heuristic.

Note however, that different heuristics may show a similar performance for a particular instance –a phenomenon that occurs in many problem domains. This represents a situation that is usually ignored by most of the hyper-heuristic approaches: the fact that two or more heuristics can be equally good or bad at a certain time [9]. Based on our experience, we believe that preserving such information about the set of all acceptable heuristics may be beneficial for the hyper-heuristic process.

Based on the previous discussion, this paper proposes a novel approach for heuristic selection that aims at keeping the information of specific subsets of heuristics that perform well for a particular instance. A natural way for doing this is through multilabel classification [10], which is a kind of supervised classification featured based on predicting a set of responses, which are heuristics in our case. In this regard, we have studied the use of evolutionary algorithms because they have been found to have a competitive performance on different supervised learning tasks [11]–[14]. More specifically, we used a cooperative coevolutionary algorithm [15], [16] to learn the set of most relevant patterns for each heuristic. The contributions of this paper are summarized as follows:

- The formulation of hyper-heuristics as supervised multilabel classifiers. To the best of our knowledge, this work is the first attempt to produce hyper-heuristics that recommend a subset of the available heuristics when they are invoked.
- A cooperative coevolutionary method to learn the most relevant patterns for each single heuristic.
- An experimental study to validate the proposed method.

With the aim to assess the effectiveness of our proposal, we have adopted a suite of constraint satisfaction problems (CSPs) as a study case. Our experimental results give evidence of the suitability of the proposed method as it is able to outperform traditional approaches.

The remainder of this paper is organized as follows. Section II presents the basic concepts required to understand the rest of the paper. Section III describes the proposed method to deal with the design of hyper-heuristics. In Section IV, we present our experimental study and the analysis of the obtained results. Finally, Section V summarizes the main conclusions of the paper and provides some possible paths for future research.

## II. Preliminaries

This section introduces basic concepts related to coevolutionary algorithms, multilabel classification, and constraint satisfaction problems, which are used as a study case for the proposed hyper-heuristic approach.

### *A. Coevolutionary Algorithms*

A coevolutionary algorithm is an evolutionary algorithm which is able to manage two or more populations simultaneously [17]. An important characteristic of these algorithms is that they allow to split the problem into different parts and assign a different population to each subproblem. Each population focuses its efforts on solving one specific part of the problem.

In a coevolutionary model, the interaction of individuals from different populations delivers the solution for the problem. Depending on the type of interaction, two different kinds of coevolutionary algorithms can be described [18]:

- **Competitive coevolutionary algorithms** [19]. The individuals of each population compete against each other. In this sort of coevolution, the fitness value of an individual decreases as the result of an increment in the fitness value of its adversaries. Competitive coevolution is normally adopted for game-like problems.
- **Cooperative coevolutionary algorithms** [15], [16]. Each population evolves individuals representing a part of the solution. A complete solution is composed by joining individuals from all the populations. Therefore, the fitness value of an individual is the result of its collaboration with other individuals from other populations.

In this paper, we focus on cooperative coevolution because this approach allows decomposing the problem into several subproblems, which can result in a natural way of handling multilabel learning problems.

### *B. Multilabel Classification*

Unlike traditional binary and multiclass classification problems –where each instance belongs to only one class label, in multilabel classification an instance may be associated with a set of labels [10]. Formally, given a dataset $\mathcal{D} = \{(\mathbf{s}_i, Y_i)\}$, where $\mathbf{s}_i = [s_1, \ldots, s_d]$ is an instance in a $d$-dimensional feature space, $Y_i \subseteq \{l_1, \ldots, l_L\}$ is the label set associated with $\mathbf{s}_i$, and $i = \{1, \ldots, m\}$ is the number of samples. The task is to learn a multilabel classifier $h : \mathbf{S} \rightarrow \mathcal{Y}$ from $\mathcal{D}$ which predicts the label set of unseen samples.

Two approaches have been widely applied to multilabel data learning: data transformation and method adaptation [10]. The first one is based on the idea of applying transformation techniques to produce multiple single label datasets from the multilabel dataset. A single label classifier is learned for each single label problem and the outputs are combined. The second approach seeks to adapt existing classification techniques, such that they generate a set of labels instead of only one.

Many existing studies in multilabel classification rely on simplifying the original problem. Nonetheless, taking into account the information about the label correlation represents one of the main concerns in this type of classification [20]–[23]. In this regard, several methods for data transformation have been proposed. Perhaps, the most well-known is the binary relevance method [24], which consists of transforming the multilabel problem into $L$ single label problems, where a classifier is trained for each single label problem. This method assumes that the labels are fully independent. A second mechanism is to implicitly incorporate the dependency information into the learning process. One way of doing this is to treat the sets of labels as a unit, and the dependency among the labels is implicitly embedded in the training process. A similar approach is based on a chain of classifiers [25], which consists of introducing the predicted class label for one classifier into the data given as input for the next one. The latter one is specially used in the present study. Thus, the different possible correlations in terms of the performance for the different single heuristics can be represented in a multilabel formulation. The heuristics considered for this study are related to those designed for solving the constraint satisfaction problem, which is detailed in the next section.

### *C. Constraint Satisfaction Problems*

The hyper-heuristic method described in this paper can be applied to different combinatorial optimization problems. To validate the proposed approach, we apply it here to solve the CSP. We selected this problem mainly because of its many practical applications [26], [27].

The CSPs considered for this investigation are defined by a set of variables and the constraints among them. Each variable can be assigned a value from a finite domain. To solve a CSP, we are requested to assign a value to every variable in the problem such that their values satisfy all the constraints [28]. CSPs are usually solved by traversing a depth-first search tree. At each node in the search tree, the algorithm must select an unassigned variable and one suitable value from its corresponding domain. If the current assignment of the variables breaks at least one constraint, the search backtracks and changes the value of a previously assigned variable, and continues the search from there.

At each node, the variable to assign, as well as the value used for the assignment, are usually selected by heuristics. In this paper, we have included five different heuristics for variable selection, which are described as follows:

- **Domain (DOM)**. DOM selects the variable with the fewest remaining values in its domain [29].
- **Degree (DEG)**. DEG selects the variable with the largest degree [30], where the degree of a variable is the number of constraints where such a variable participates.
- **Kappa (K)**. K selects first the variable that minimizes the $\kappa$ value of the resulting instance [31]:

$$\kappa = \frac{-\sum\limits_{c \in C} \log_2(1 - p_c)}{\sum\limits_{x \in X} \log_2(d_x)} \tag{1}$$

  where $p_c$ represents the tightness of constraint $c$ (the proportion of forbidden pairs of values in the constraint) and $d_x$ stands for the domain size of variable $x$.
- **WDEG**. WDEG attaches a weight to every constraint in the problem and increases it when its respective constraint fails during the search [32], [33]. The weighted degree of a variable is calculated as the sum of the weights of the constraints in which the variable is currently involved. WDEG selects the variable with the largest weighted degree.
- **Domain over degree (DOM/DEG)**. It is the result of the combination of DOM and DEG into a single heuristic. DOM/DEG tries first the variable with the smallest quotient of the domain size over the degree of the variables [34].

Once a variable has been selected by using any of the previously described heuristics, the first available value in the domain of the selected variable is assigned to such a variable.

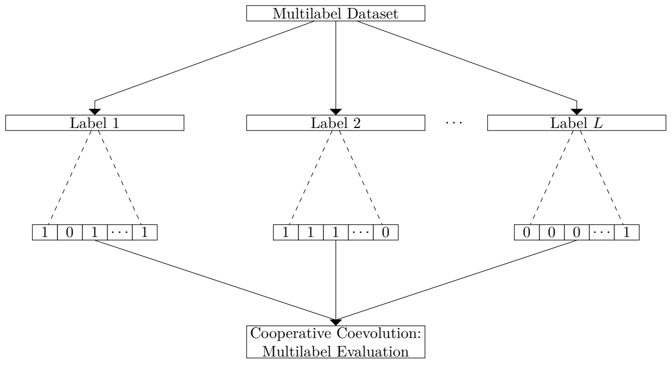

Fig. 1: Population scheme for the evolutionary multilabel learning

## III. Evolutionary Multi-Label Hyper-Heuristics Design

In this section, we propose a coevolutionary algorithm to learn a set of patterns from the training data, which can be used as a set of rules in the hyper-heuristic design. The coevolutionary algorithm takes into account not only the performance for each label, but also a measure of the degree of disagreement among the correlated samples. The rest of this section describes the proposed approach.

### A. Initialization

As usually happens with evolutionary algorithms, the learning process starts with an initial set of solutions. These solutions are generated from the training set and aim at selecting a subset of training samples to be used by the $k$-nearest neighbor ($k$-NN) classifier as the hyper-heuristic. For doing so, potential solutions are encoded using a binary representation, as follows:

$$\mathbf{x} = [b_1, \dots, b_m] \tag{2}$$

where $b = \{0, 1\}$ indicates whether a sample is used or not and $m$ is the dataset size.

Since in a multilabel classification problem there is more than one label, a coevolutionary approach is adopted. In the coevolutionary formulation, a subpopulation of instances is used for each label. Thus, all subpopulations share the same basic chromosome definition, as it is shown in Fig. 1.

By using this representation scheme, all the individuals will define a subset of the original dataset, with each focused on selecting the most relevant subset of samples for each label. Each individual symbolizes a reduced subset which will be employed as a training set by the $k$-NN classifier.

### B. Evolutionary Operators

The subpopulations are evolved separately. The individuals in each subpopulation are subject to two operations: crossover and mutation. Since a binary representation is used, we adopt Uniform Crossover. In this operator, each

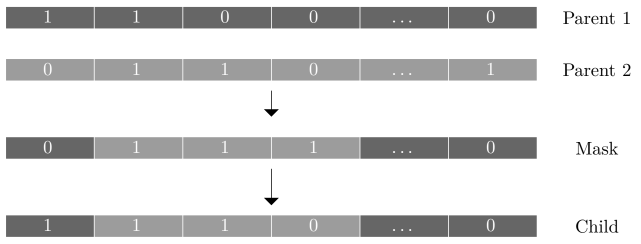

Fig. 2: Uniform crossover operation

bit is compared between two parents. The bits are swapped with a fixed probability. This operator is graphically depicted in Figure 2.

We also adopt bit-flip mutation. In this operator, for each bit, a random uniform probability is generated and those positions where the random number is less than the mutation probability are flipped. Finally, a binary tournament selection is adopted to choose the parents to be used in the reproduction step.

## C. Fitness Function

In the optimization problem for multilabel learning, two objectives are considered: (1) minimize the error rate of a particular label and (2) minimize the Hamming score in the multilabel problem.

A chain of classifiers is adopted for the multilabel classification. This means that a total of $L$ classifiers are required, one per each label in the dataset. A characteristic of the chain of classifiers is that each classifier is linked to the rest through the labels of the previous classifiers. Using this principle, an instance is classified as follows:

$$\forall i \in \{1, \ldots, L\} : y_i^* = KNN\left(\mathcal{S}^*, \mathcal{Z} \cup (y_1, \ldots, y_{i-1})\right) \quad (3)$$

where $KNN$ is a function that returns the predicted label, $\mathcal{S}^*$ is the reduced training set, $\mathcal{Z}$ is the set of samples to be classified, and $y_i$ is the $i^{th}$ label.

The full solution for the multilabel classifier is built considering an individual for the current subpopulation and the best individuals for the others. Therefore, an individual for each subpopulation is needed to compute the fitness value. As we have previously stated, two measures are considered: the error rate of the current label ($err_{cl}$) and the Hamming distance of the multilabel prediction ($H$), which are defined as follows:

$$err_{cl} = \frac{1}{m}\sum_{j=1}^{m}\mathcal{L}\left(y_{cl}^{*(j)}, y_{cl}^{(j)}\right) \quad (4)$$

$$H = \frac{1}{m}\sum_{j=1}^{m}\frac{\sum_{i=1}^{L}\mathcal{L}\left(y_i^{*(j)}, y_i^{(j)}\right)}{L} \quad (5)$$

where $\mathcal{L}$ is a loss function that computes the error incurred by the classifier. In our case, we have used the $^0/_1$ loss function, which takes the value of 1 if the prediction differs from the true label and 0, otherwise. At this point, the fitness value of an individual is computed as:

$$Fitness = \alpha \cdot err_{cl} + (1 - \alpha) \cdot H \quad (6)$$

where $\alpha$ is a weighting factor in the interval $[0, 1]$. In this study, this value is set to 0.5.

## D. Cooperative Coevolutionary Algorithm for Multilabel Learning

In this subsection, we describe the cooperative coevolutionary algorithm used for performing multilabel learning. Algorithm 1 describes a basic pseudocode for the proposed approach to perform multilabel learning. Next, we explain in detail each instruction.

**Algorithm 1** CCML

```
1: Generate randomly an initial population, $\mathcal{P}_y$ for each label $y \in \mathcal{Y}$
2: Select individuals for each label
3: Evaluate all the population labels using the fitness function
4: while a stopping criterion is not met do
5:     Select the best solution for each label in the last generation
6:     Create an offspring for each population label
7:     Evaluate individuals for each population label using the fitness function
8:     Select a population for each label using both the parent and the offspring populations
9: end while
```

- Instruction 1 generates an initial population for each label in the multilabel dataset. This step includes the random generation of the individuals using the representation described in subsection III-A.
- The second instruction randomly selects an individual for other population labels to build a complete solution to the problem.
- In instruction 3, the evaluation of the fitness of each individual is performed. This evaluation is done by considering the blocks of complete solutions constructed and using the fitness function defined for this purpose.
- Here is where the iterative coevolutionary process starts.
  - In instruction 5, the best solutions for each population label in the last generation are selected in order to be used in the construction of a complete solution to the multilabel problem.
  - Instruction 6 creates an offspring for each population based on the evolutionary operators described in subsection III-B.
  - Next, the children are evaluated using the fitness function, instruction 7.

  – Instruction 8 constructs the population for the next generation by selecting the best individuals both for the parents and for the offspring populations.

The evolutionary process finishes when a given stopping criterion is satisfied. The final solution is obtained by joining the best individual for each population. With this solution, the final multilabel model is constructed, which would represent the designed hyper-heuristic to be used for solving future problems.

## IV. Experiments and Results

In this section, we describe the configuration for the experiments as well as the results obtained with the proposed method and with respect to other studies.

### A. Experimental Settings

This section describes the instances used as well as the validation process that we adopted.

*1) Benchamrk Instances:* For our experiments, we used a set of benchmark CSPs taken from a public repository.[1] The 482 CSP instances considered for this research are coded in `XCSP` 2.1 format and represented in extension. Among the available instances in the repository, we included a mixture of sets that contain instances from different classes, such as random, quasi random and patterned instances.[2]

Every instance in this research has been characterized by using the following CSP features:

- **Global constraint density**. The global constraint density is calculated as the number of constraints in the instance divided by $n(n-1)/2$, which represents the maximum number of possible bidirectional constraints in the instance (where $n$ stands for the number of variables in the instance).
- **Global constraint tightness**. The local constraint tightness is calculated as the fraction of forbidden pairs of values in a given constraint. The global constraint tightness of a CSP instance is the average of the local constraint tightness among all the constraints.
- **Kappa**. The value of $\kappa$ is used as an estimate of the hardness of an instance in relation to its size [31]. This feature is calculated as depicted in Eq. 1.
- **Global clustering coefficient**. The neighborhood of a variable $x$ is composed by all the variables immediately connected to $x$ by one constraint. The local clustering coefficient is calculated as the constraint density of the neighborhood. Thus, the global clustering coefficient can be defined as the average of the local clustering coefficients among all the variables in the instance. The local clustering coefficient represents the number of constraints among the neighbors of a variable over the variable that is directly connected to (by a constraint).

*2) Validation Process:* We have adopted *hold-out* as our validation procedure. In hold-out validation, the dataset is divided into two disjoint parts, such that one part is used to fit the model parameters and the other one is used to assess performance. For our experiments, 50% of the dataset is used as the training set and the remainder 50% is used as the test set. We decided in favor of hold-out instead of the well-known cross validation due to the fact that stratified sampling from cross validation could not work for multilabel problems.

Two performance measures are adopted to assess the effectiveness of our proposal. The first one is the success rate, which counts the ratio at which a given method is able to solve the set of problems/instances at a given time, which has been fixed to 20 seconds. The second one is related to a measure of the performance of the selected heuristic; in this case, this measure is the time required for the heuristic to solve the CSP.

The statistical evaluation is carried out by a set of non-parametric tests. The Wilcoxon signed rank test [35] is used for pairwise comparisons, and the Aligned Friedman test [36] with Holm's procedure [37] are used for multiple comparisons. These non-parametric tests have been widely recommended for safe and robust comparisons in [38]–[41]. In all cases, the significance level is set to $\alpha = 0.05$.

Regarding the parameters for the evolutionary multilabel method, a population size equals to 80 for each label is considered. The probability of crossover (CR) and mutation (MR) is fixed in 1 and 0.10, respectively. In order to avoid the risk of producing overfitting during the learning step, the stopping criterion is fixed to perform a relative low number of fitness evaluations. The number of evaluations (evals) of the fitness function is set to 2,000. Moreover, the value of $k$ equal to 7 is used for the KNN evaluation. These parameters were experimentally determined. For doing this, we have evaluated, using the two-fold cross validation, the performance of the proposed method under each configuration of $CR = \{0.8, 0.9, 1.0\}$, $MR = \{0.05, 0.10, 0.20\}$, and $k\,\{1, 3, 5, 7, 9\}$.

### B. Experimental Study

In this subsection, we present the experimental study performed to assess the validity of our proposal. The experimental study is divided into three parts: the first one compares the performance of the hyper-heuristic against the single heuristics. The second one aims at comparing the multilabel approach against a traditional multiclass one. Finally, the third one compares with a state-of-the-art method specifically designed for producing hyper-heuristics for CSPs.

*1) Comparing with Single Heuristics:* The aim of this experimental study is to compare the performance of

[1] Available at http://www.cril.univ-artois.fr/~lecoutre/benchmarks.html

[2] The specific sets of instances used in this work can be referred to by the names given in the repository: `2-30-15`, `geom`, `ehi-85`, `25-10-20`, `coloringExt`, `bqwh-15-106` and `bqwh-18-141`

TABLE I: Performance of the proposed approach (EMLHH) and single heuristics.

| Method | Success Rate | Time (ms) |
|---|---|---|
| DOM | 0.8132 | 5742.77 |
| DEG | 0.3444 | 14721.63 |
| Kappa | 0.7759 | 5921.61 |
| WDEG | 0.6639 | 8557.13 |
| DOM/DEG | 0.6515 | 8291.64 |
| EMLHH | 0.9501 | 2328.94 |

TABLE II: Ranks obtained with Aligned Friedman test and adjusted p-values (APV) determined with Holm's procedure.

| Method | Rank | $p$ | APV |
|---|---|---|---|
| DEG | 1110.86 | $< 0.05$ | $< 0.05$ |
| WDEG | 813.11 | $< 0.05$ | $< 0.05$ |
| DOM/DEG | 784.61 | $< 0.05$ | $< 0.05$ |
| DOM | 638.24 | $< 0.05$ | $< 0.05$ |
| Kappa | 596.77 | $< 0.05$ | $< 0.05$ |
| EMLHH | 397.40 | — | — |

TABLE III: Results obtained with EMLHH and standard multiclass classifiers for building hyper-heuristics.

| Method | Success Rate | Time (ms) |
|---|---|---|
| NB | 0.9419 | 2643.12 |
| MLP | 0.8797 | 3789.05 |
| KNN | 0.9378 | 2625.37 |
| J48 | 0.9378 | 2571.95 |
| RF | 0.9129 | 3116.96 |
| EMLHH | 0.9501 | 2328.94 |

TABLE IV: Ranks obtained with the Aligned Friedman test and adjusted p-values (APV) determined with Holm's procedure.

| Method | Rank | $p$ | APV |
|---|---|---|---|
| MLP | 833.45 | $< 0.05$ | $< 0.05$ |
| RF | 732.30 | 0.13 | 0.53 |
| NB | 713.63 | 0.31 | 0.53 |
| KNN | 701.00 | 0.50 | 0.53 |
| J48 | 685.52 | 0.78 | 0.99 |
| EMLHH | 675.10 | — | — |

the hyper-heuristic based on a cooperative coevolutionary algorithm (EMLHH) against the single heuristics. This will allow us to show the benefits of our proposal. Table I shows the results obtained by each heuristic and the proposed method. We report the success rate and the average time spent for each heuristic/hyper-heuristic to solve the problem. It is worth noting that for comparative purposes, in the time measure, when considering the multilabel approach, one random label is selected from those recommended by EMLHH.

We perform a statistical analysis to determine if such differences are statistically significant. The Aligned Friedman test and Holm's procedure are used to this end. The results of these tests are shown in Table II

Based on the results and statistical tests, the following can be stressed:

- Hyper-heuristics outperform all single heuristics in terms of the success rate. This means that for most problems, the hyper-heuristic is able to find a solution to a CSP instance.
- Due to the evolutionary process, the learning of the hyper-heuristic can be computationally expensive. Nevertheless, once this has been built, the time to give a recommendation is usually small.
- The time required to solve the problem for the hyper-heuristic is lower than that required for the best heuristic (DOM).
- The results have been supported by statistical tests, which have shown that the hyper-heuristic is able to improve with a statistically significant difference to all the heuristics considered in this study.
- Selecting the most suitable heuristic for each problem provides better solutions than working with a single one.

*2) Comparing with a Multiclass Approach:* The main goal of this section is to compare the performance of the proposed method (EMLHH) against the traditional multiclass approach. To this end, we have trained several common multiclass classifiers using the same dataset. The set of classifiers are Naïve Bayes (NB), Multilayer Perceptron (MLP), K-nearest neighbors (KNN), a decision tree (J48), and Random Forest (RF). It is worth noting that in those samples for which more than one heuristic performs well, only one is randomly chosen.

Table III reports the obtained results both for EMLHH and the multiclass classifiers. We report both the average success rate of the single heuristics and the resulting time when such hyper-heuristic is used to solve the problem.

Table IV reports the results of the statistical tests performed over this set of experiments.

Based on these results, the following can be highlighted:

- The success rate of EMLHH is superior to the one obtained by common multiclass approaches. This is due to the fact that, for similar problems, different heuristics can achieve similar performance, which makes harder for a single multiclass approach to correctly predict the assigned target.
- When comparing with respect to the time required to solve the problem, we can notice a similar behavior. The time for solving the problems is, on average, lower with the proposed approach than with standard ones.
- For most of the reference methods, there is no statistically significant difference, but the success rate for the hyper-heuristic is better. This means that, on average, for most cases, EMLHH is able to find a solution using our available time budget.
- The lack of a statistical significant difference is due to the fact that multiclass classifiers can recommend a heuristic with a similar performance to the one assigned as the target, which further confirms our initial premise that has motivated this study: different heuristics can perform well for a given problem.

TABLE V: Reported results using the proposed approach (EMLHH) and a reference method (EHH) [8].

| Case | EHH | EMLHH |
|---|---|---|
| Worst | 7972.51 | 2688.20 |
| Median | 7220.88 | 2328.94 |
| Best | 5964.96 | 2293.80 |

The use of coevolution for breaking the multilabel classification problem into several single label problems has been found to be quite beneficial, allowing EMLHH to produce a more accurate selection of the single heuristic and keeping the information of those having a similar performance.

The cooperation between individuals of different populations allows us to select the most relevant patterns for each label at the time that it removes the samples that can be considered as noisy and irrelevant for the labels. This can be noted in the improvement in the performance measures.

*3) Comparing with a Reference Method:* The goal of this section is to show the benefits of our proposed approach with respect to a classical evolutionary method. To this aim, we have compared with EHH, described in [8]. This generator runs a genetic algorithm to produce rules of the form `state` $\rightarrow$ `heuristic`. By following this methodology, we generated three hyper-heuristics specifically designed for the sets of instances and heuristics considered in this investigation.

Table V shows the results obtained for each method. We compare in terms of the required time for each method to solve the CSPs. Since evolutionary algorithms work with a randomly generated population, we have run the methods using ten different seeds, and we report the best, median, and the worst case for each one.

We have applied the Wilcoxon Signed Rank test in order to validate if the difference between both methods is statistically significant. This test has revealed with a $p < 0.05$ that, indeed, the performance difference between both approaches is statistically significant.

If we analyze the results shown in Table V, we can note that EMLHH outperforms EHH. Our approach is able to significantly reduce the time spent in solving the problem.

## V. Conclusions and Future Work

This paper introduced an evolutionary multilabel approach to design hyper-heuristics, which is based on the idea that, for some problems, different heuristics can have a similar performance. Through an experimental study, we have shown that the use of a cooperative scheme allows our approach to select the most relevant patterns for each label.

The results achieved by the proposed method in the experimental study have shown that it is able to offer a significant improvement when it is compared to the standard multiclass approach and an evolutionary method designed to generate hyper-heuristics for solving CSPs. These results have been contrasted with statistical tests, which have confirmed our hypothesis that the use of a multilabel approach can lead to the design of more effective hyper-heuristics. Moreover, the CSPs have been used as a study case; nonetheless, this approach attempts to be general and it can be applied to other domains without requiring major modifications.

As a result of our experimental study, we can also point out some interesting paths for future research. For instance, the fitness function is based on an aggregated formula that combines both error rate per label and the Hamming score. A natural extension is to formulate it as a multi-objective optimization problem by explicitly and simultaneously optimizing these two criteria. Moreover, exploring different learning techniques, such as neural networks in the cooperative coevolutionary method is another interesting path. Finally, we would like to test the proposed method in other domains in order to assess its applicability to other types of problems.

## Acknowledgements

This research was partially supported by CONACyT project no. 241461 and ITESM Research Group with Strategic Focus in intelligent Systems. The last author gratefully acknowledges support from CONACyT project no. 221551.

## References

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[2] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[3] P. Cowling, G. Kendall, and E. Soubeiga, *A Hyperheuristic Approach to Scheduling a Sales Summit.* Springer Berlin Heidelberg, 2001, pp. 176–190.

[4] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, *Hyper-Heuristics: An Emerging Direction in Modern Search Technology.* Boston, MA: Springer US, 2003, pp. 457–474.

[5] P. Ross, *Hyper-Heuristics.* Springer US, 2005, pp. 529–556.

[6] K. Sim, E. Hart, and B. Paechter, "A lifelong learning hyper-heuristic method for bin packing," *Evol. Comput.*, vol. 23, no. 1, pp. 37–67, Mar. 2015.

[7] F. Alanazi and P. K. Lehre, *Limits to Learning in Reinforcement Learning Hyper-heuristics.* Cham: Springer International Publishing, 2016, pp. 170–185.

[8] J. C. Ortiz-Bayliss, H. Terashima-Marín, and S. E. Conant-Pablos, "Combine and conquer: an evolutionary hyper-heuristic approach for solving constraint satisfaction problems," *Artificial Intelligence Review*, vol. 46, no. 3, pp. 327–349, 2016.

[9] J. H. Moreno-Scott, J. C. Ortiz-Bayliss, H. Terashima-Marín, and S. E. Conant-Pablos, "Experimental matching of instances to heuristics for constraint satisfaction problems," *Intell. Neuroscience*, vol. 2016, pp. 37:37–37:37, Jan. 2016.

[10] F. Herrera, S. Ventura, R. Bello, C. Cornelis, A. Zafra, D. Sánchez-Tarragó, and S. Vluymans, *Multiple Instance Learning.* Springer International Publishing, 2016, pp. 17–33.

[11] S. Bandyopadhyay, U. Maulik, C. A. Coello Coello, and W. Pedrycz, "Guest editorial: Special issue on advances in multiobjective evolutionary algorithms for data mining," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 1–3, 2014.

[12] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems Man and Cybernetics Part C–Applications and Reviews*, vol. 38, no. 3, pp. 397–415, May 2008.

[13] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. Coello Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 4–19, 2014.

[14] ——, "Survey of multiobjective evolutionary algorithms for data mining: Part II," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 20–35, 2014.

[15] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, ser. PPSN III. Springer-Verlag, 1994, pp. 249–257.

[16] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[17] R. P. Wiegand, "An analysis of cooperative coevolutionary algorithms," Ph.D. dissertation, Fairfax, VA, USA, 2004.

[18] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, 2nd ed. Springer, US, 2007.

[19] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, no. 1, pp. 1–29, 1997.

[20] E. Alvares-Cherman, J. Metz, and M. C. Monard, "Incorporating label dependency into the binary relevance framework for multi-label classification," *Expert Systems with Applications*, vol. 39, no. 2, pp. 1647 – 1655, 2012.

[21] K. Dembszynski, W. Waegeman, W. Cheng, and E. Hüllermeier, "On label dependence in multilabel classification," in *ICML Workshop on Learning from Multi-label data*. Ghent University, KERMIT, Department of Applied Mathematics, Biometrics and Process Control, 2010.

[22] L. Tenenboim-Chekina, L. Rokach, and B. Shapira, "Identification of label dependencies for multi-label classification," in *Working Notes of the Second International Workshop on Learning from Multi-Label Data*, 2010, pp. 53–60.

[23] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 999–1008.

[24] T. Gonçalves and P. Quaresma, "A preliminary approach to the multilabel classification problem of portuguese juridical documents," in *Portuguese Conference on Artificial Intelligence*. Springer, 2003, pp. 435–444.

[25] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, pp. 254–269, 2011.

[26] J. Berlier and J. McCollum, "A constraint satisfaction algorithm for microcontroller selection and pin assignment," in *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, march 2010, pp. 348–351.

[27] S. V. Bochkarev, M. V. Ovsyannikov, A. B. Petrochenkov, and S. A. Bukhanov, "Structural synthesis of complex electrotechnical equipment on the basis of the constraint satisfaction method," *Russian Electrical Engineering*, vol. 86, no. 6, pp. 362–366, 2015.

[28] R. Dechter, "Constraint networks," in *Encyclopedia of Artificial Intelligence*. Wiley, 1992, pp. 276–286.

[29] P. W. Purdom, "Search rearrangement backtracking and polynomial average time," *Artificial Intelligence*, vol. 21, pp. 117–133, 1983.

[30] R. Dechter and I. Meiri, "Experimental evaluation of preprocessing algorithms for constraint satisfaction problems," *Artificial Intelligence*, vol. 38, no. 2, pp. 211–242, 1994.

[31] I. P. Gent, P. Prosser, and T. Walsh, "The constrainedness of search," in *Proceedings of AAAI'96*, 1999, pp. 246–252.

[32] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais, "Boosting systematic search by weighting constraints," in *European Conference on Artificial Intelligence (ECAI'04)*, 2004, pp. 146–150.

[33] C. Lecoutre, F. Boussemart, and F. Hemery, "Backjump-based techniques versus conflict-directed heuristics," in *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, ser. ICTAI '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 549–557.

[34] A. Arbelaez, Y. Hamadi, and M. Sebag, "Online heuristic selection in constraint programming," in *Symposiumon Combinatorial Search (SoCS)*, 2009, pp. 1–8.

[35] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[36] J. L. Hodges and E. L. Lehmann, "Rank methods for combination of independent experiments in analysis of variance," *The Annals of Mathematical Statistics*, vol. 33, no. 2, pp. 482–497, 1962.

[37] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.

[38] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, pp. 1–30, 2006.

[39] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3 – 18, 2011.

[40] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[41] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.