Hindawi
Complexity
Volume 2018, Article ID 4963139, 13 pages
https://doi.org/10.1155/2018/4963139

Research Article

A Scalable Genetic Programming Approach to Integrate miRNA-Target Predictions: Comparing Different Parallel Implementations of M3GP

Stefano Beretta,[1,2] Mauro Castelli,[3] Luis Muñoz,[4] Leonardo Trujillo,[4] Yuliana Martínez,[4] Aleš Popovič,[3,5] Luciano Milanesi,[2] and Ivan Merelli[2]

[1]*DISCo, Università degli Studi di Milano-Bicocca, Milan, Italy*
[2]*Istituto di Tecnologie Biomediche, Consiglio Nazionale delle Ricerche, Segrate, Italy*
[3]*NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal*
[4]*Tree-Lab, Posgrado en Ciencias de la Ingeniería, Instituto Tecnológico de Tijuana, Tijuana, BC, Mexico*
[5]*Faculty of Economics, University of Ljubljana, Kardeljeva Ploščad 17, SI-1000 Ljubljana, Slovenia*

Correspondence should be addressed to Mauro Castelli; mcastelli@novaims.unl.pt

Received 28 February 2018; Accepted 29 May 2018; Published 4 July 2018

Academic Editor: Dimitrios Vlachakis

There are many molecular biology approaches to the analysis of microRNA (miRNA) and target interactions, but the experiments are complex and expensive. For this reason, in silico computational approaches able to model these molecular interactions are highly desirable. Although several computational methods have been developed for predicting the interactions between miRNA and target genes, there are substantial differences in the results achieved since most algorithms provide a large number of false positives. Accordingly, machine learning approaches are widely used to integrate predictions obtained from different tools. In this work, we adopt a method called multidimensional multiclass GP with multidimensional populations (M3GP), which relies on a genetic programming approach, to integrate and classify results from different miRNA-target prediction tools. The results are compared with those obtained with other classifiers, showing competitive accuracy. Since we aim to provide genome-wide predictions with M3GP and, considering the high number of miRNA-target interactions to test (also in different species), a parallel implementation of this algorithm is recommended. In this paper, we discuss the theoretical aspects of this algorithm and propose three different parallel implementations. We show that M3GP is highly parallelizable, it can be used to achieve genome-wide predictions, and its adoption provides great advantages when handling big datasets.

1. Introduction

MicroRNAs (miRNAs) are approximately 22-nucleotide-long, single-stranded RNA molecules encoded in the genomes of plants, animals, and viruses and are capable of interfering with intracellular messenger RNAs (mRNA) [1]. miRNAs are key regulators of gene expression at the posttranscriptional level, but the precise mechanisms underlying their interactions with the respective gene targets are still poorly understood. The effect of the hybridization between a miRNA and its target mRNA is that the expression of the protein coded by the gene is silenced, either by stopping the translation process or by marking the mRNA for degradation.

Since miRNAs are involved in the onset of many different diseases, the study of their interactions with the genome is very important. For instance, several recent reports suggest that miRNA aberrations may be an important factor in the development of cancer [2, 3]. Another study demonstrated that more than 50% of miRNA targeted genes are located in cancer-associated genomic regions or in fragile sites [4], indicating that miRNAs may play an important role in the pathogenesis of many human diseases.

It is also a challenging problem to identify miRNAs on an experimental basis due to their limited expression, which arises from the dynamic behavior of the regulation process and the tissue specificity of their control mechanism. Moreover, taking into account the different mechanisms by which miRNAs exert their role, only the absence of the protein in miRNA-transfected cells represents definitive proof of the miRNA-target interaction. This technique is very complex and costly to achieve, thereby imposing serious constraints on the number of experiments that can be performed.

Therefore, computational predictions represent a very important approach for screening possible targets to be experimentally tested. More precisely, the interactions between a miRNA and its mRNA target sites can be considered from thermodynamic, probabilistic, and evolutionary (or sequence-based) points of view. Several computational tools for predicting miRNA-target sites have been developed in recent years using one or more of the aforementioned aspects [5].

Many works have been done to compare their performance (see [6–8] for details), taking into account the most famous tools: PITA [9], miRSystem [10], miRmap [11], DIANA-microT-CDS [12], CoMir [13], mirWalk [14], and PicTar [15].

Among the most well-known prediction tools for miRNA-target recognition, three of the most adopted ones are miRanda [16, 17], TargetScan [18, 19], and RNAhybrid [20]. miRanda completes three sequential steps: (i) sequence matching to find the maximal local complementarity between a mature miRNA and the putative target site, (ii) free energy calculation to estimate the strength of a potential RNA duplex, and (iii) filtering of predicted targets on the basis of evolutionary conservation. TargetScan is based on two hypotheses: (i) highly conserved miRNAs are more involved in regulation and (ii) membership in large miRNA families leads to a higher number of existing targets. After the matching step (allowing wobble pairs and stopping at the first mismatch encountered), a thermodynamic evaluation of the RNA duplex is performed. Finally, RNAhybrid predicts the target genes based on free energy calculation. It collects the most favorable energetic structures, normalizes them, and then uses estimated p values to determine the significance of each predicted binding site.

We decided to focus on these three tools since, as described before, they employ different approaches to predict the interactions, such as sequence matching, thermodynamic evaluation of the RNA duplex, or free energy calculation. This aspect is fundamental since one of the main goals of this work is to combine results obtained with different approaches. Finally, we would like to point out that some of the other available tools base their prediction on the output of one of the three tools we considered, or employ a similar technique.

Although common guidelines are adopted by the aforementioned methods, differences in the definition of the physical models (and uncertainties concerning the real biological mechanisms) and differences in the formalization of the corresponding algorithms (and implementations) result in quite different miRNA-target predictions. Moreover, by comparing computational results with experimental validations, we can see that these tools produce a large number of false-positive predictions.

The lack of a clear consensus on the predictions achieved by these tools has led to the development of methods to perform meta-analyses of the results by integrating lists of miRNA-target genes predicted by several algorithms. Among such integrated tools, the most used ones are miRGator [21] and ExprTarget [22]. These tools exploit functional analyses and genome annotations to better characterize the identified targets. miRGator also provides miRNA expression profiles by importing expression experiments from the Gene Expression Omnibus databank [23]. Analogously, expression profiles are reported in mESAdb [24] and mirEX [25]. MAGIA [26] returns predictions as unions or intersections of results produced by TargetScan, miRanda, and RNAhybrid. Moreover, it integrates mRNA expression values with miRNA expression scores in order to elucidate inverse correlations, thus hypothesizing about new miRNA-target associations. myMIR [27] implements a pipeline for computing ranked miRNA-target lists, integrating predictions from different tools. This approach also provides functional annotations for characterizing genes targeted by each miRNA, highlighting overrepresented ontological terms.

In a previous work [28], we described the application of a classification technique based on genetic programming, called M3GP, to integrate results from three different miRNA-target prediction tools. More precisely, we considered this as a classification problem and started from a set of positive and negative examples used to train the adopted method. Although the M3GP method has been developed to improve the performance on multiclass problems, it has also been shown to achieve good results with binary problems. The idea behind this method is to improve the standard genetic programming technique in which a population of candidate solutions is evolved by applying genetic operators until an ending criterion is reached. In particular, the idea of M3GP is to define a function that transforms the input data by mapping them into another feature space. The objective of M3GP is to evolve transformations of the input data in such a way that it becomes easier to perform the classification task in the new feature space of the problem.

To assess the performance of the M3GP method on this classification problem, we compared the obtained results with those achieved by other methods which were applied to the same data, showing that M3GP always achieves good accuracy. Another key point relates to the type of models evolved by M3GP relative to other GP techniques. In M3GP, the models are composed of a set of independent subtrees allowing for the individuals to be evaluated in parallel to each other, an approach that is successfully applied in this work.

Indeed, considering the high number of species to analyze (this mechanism can be studied in several species for which the genome is already available), the number of miRNAs identified (in humans, the total is about 2,000), and the number of genes to study (in humans about

20,000), a good strategy to parallelize M3GP is essential. In this work, we discuss some important aspects of this algorithm, proposing three different parallel implementations. In summary, we present the following contributions. First, we show that M3GP can perform competitively in predicting interactions between miRNA and targets. In this task, the advantage of our method is that it does not overfit the training data and presents a minimal amount of variance over multiple runs. In other words, M3GP appears to be quite robust in this domain. Second, from an implementation perspective, we show that the method is highly parallelizable, particularly given the manner in which the output is constructed. The results show that when the search contains large populations with large multidimensional individuals, the best strategy is to parallelize the evaluation of each individual transformation.

The paper is organized as follows: Section 2 provides a description of the data used in the experimental phase where the performance of the classification algorithm in integrating the predictions of miRNA-target sites was evaluated. Section 3 presents the M3GP algorithm and describes its properties. Section 4 discusses the obtained results, also taking other state-of-the-art machine learning techniques into account. Section 5 presents the parallel implementations of the M3GP system, with a detailed discussion of the performance achieved. Section 6 concludes the paper and suggests possible avenues for future research.

2. Background

In this section, we describe the datasets used in this work and how we obtained the data on which the proposed method was tested. We started by downloading from the TargetScan website (http://www.targetscan.org/) the sequences of the miRNA families and of the untranslated regions (UTRs, the genomic loci targeted by miRNA) from 23-way alignment (We have used Release 6.1 of November 2011 to avoid problems in the names of the miRNA sequences with respect to those present in the other datasets we considered for the analysis.) More precisely, the TargetScan database contains the miRNA sequences obtained from miRBase (with the additional information about the miRNA families) and the 3UTR sequences obtained from the UCSC. We decided to use the data from TargetScan to be able to match the identifiers (of both miRNA and UTR sequences) with those used in the study we employed in our experimental analysis. Subsequently, we filtered the information relative to the *Homo sapiens* species and obtained a total of 30,887 UTR and 1,558 miRNA sequences, which were used as the starting point of our analysis.

To identify the miRNA-target interactions, we used three target prediction tools: miRanda [16, 17], TargetScan [18, 19], and RNAhybrid [20]. The results produced by these tools were combined in a matrix; by looking at the positions on the UTR of each predicted interaction and, for each of them, we considered the score corresponding to the obtained prediction. Moreover, to handle the missing predictions of some tools, we replaced the missing values with Not-a-Number (NaN). To deal with such values during the experiments performed, we assigned penalizing scores to them. About the missing data, we conducted some experiments to assess how the choice of the penalization influences the final classification results. Results of this preliminary analysis showed that the value of the penalization score only marginally affects the final results. The overall matrix was composed of 48,121,946 elements (30,887 UTRs $\times$ 1,558 miRNAs) and 3 columns corresponding to the scores of the considered prediction tools (in addition to the other columns containing information about the miRNA-target interaction).

As discussed in [29], one problem when using machine learning methods to address the prediction of miRNA-target interactions is the lack of negative examples or miRNA nontarget pairs. In fact, since having a good training set is crucial when applying these techniques for solving classification problems, the current approaches tend to randomly generate sequences to be used as negative examples that, by the way, could be unrealistic. For this reason, we decided to adopt the results proposed in [29] to populate the dataset used to train the proposed classifier.

In that work, the authors generated two sets of positive and negative miRNA-target examples. The former set (positive examples) was obtained by biologically verified experiments, while the latter examples (negative) were identified from a pooled dataset of predicted miRNA-target pairs. More precisely, the authors selected a set of computationally predicted (by one or more algorithms) targets of miRNA, measuring the tissue specificity for both of them. Then, significantly overexpressed miRNA-mRNA pairs were selected as potential negative examples, and a further expression profiling test was performed to discard those that did not pass this filter. Finally, the thermodynamic stability and seed-site conservation were measured to infer the final negative examples. The downloaded dataset from [29] is composed of 288 (Out of the 289 positive examples available in the dataset, we were not able to find a match for one of them, which was discarded.) positive examples and 286 negative examples of miRNA-target interactions.

To further increase the set of positive examples in the dataset, we downloaded the data from miRTarBase [30], which is a database of experimentally validated miRNA-target interactions. In this way, it was possible to classify the positive examples into positive and experimentally validated examples, only experimentally validated examples, and only positive examples. More specifically, we crossed miRTarBase interactions with the positive examples from [29], in order to make the dataset more robust. We thus labelled the positive examples with three different labels (although the final classification problem is binary), which are (1) for the positive examples from [29] that were also present in the miRTarBase, (2) for the (positive) interactions only found in the miRTarBase, and (3) for those only present in the positive examples from [29], while we labelled with 1 the negative examples (from [29]). In this way, in the binary classification problem encountered in this work, we selected two balanced sets of elements: the first one among the negative examples was labelled with 1, while the second one among those labelled with 1, 2, and 3 since all of them represent "positive examples" (from [29], miRTarBase, or both). To obtain the final matrix, we combined the results of the prediction tools with these classification labels and, for those predicted interactions that are neither negative nor positive (in any of the three sets), we added the label 0. Finally, we refined these results by eliminating redundancies in the two classes of examples which are caused by notational problems (i.e., the same mRNA identified multiple times). Table 1 reports the number of elements in the matrix for each classification type.

Note that the *unknown examples* are elements of the matrix for which we do not have a priori knowledge regarding whether the interaction is true or not. For this reason, these examples could be used to identify new interaction candidates to be biologically validated; that is, those classified as positive by the proposed method could represent novel miRNA-mRNA interactions.

Table 1: Description of the types of samples in our dataset.

Classification	Number of elements
Negative examples	286
Positive and exp. validated examples	179
Only exp. validated examples	286
Only positive examples	6376
Unknown examples	48114996
Total	48121946

3. Method

In this work, we address a binary classification problem in which the two classes refer to the examples described in Section 2. More precisely, the first class refers to negative examples, while the second class refers to positive ones. To simplify the classification process, we take an equal number of elements from both classes (for the positive ones, we combined the three classes of positive examples of Table 1), defining a balanced classification task that is easier to handle for most machine learning classifiers. Figure 1 presents different views of the distribution of all samples used from each class, showing three 2D views, one for each pair of features (prediction tools). As it is possible to notice from these scatter plots, the problem basically shows three clusters of data, but in each of them the classes are multimodal and overlap within the feature space. This view clearly shows that the problem of distinguishing the classes is quite difficult.

In addition to M3GP, we executed several other standard classifiers from the machine learning literature to solve this problem (in order to evaluate the proposed method's performance with respect to the state-of-the-art techniques), as follows:

(1) Euclidean distance classifier (ED)

(2) Mahalanobis distance classifier (MD)

(3) Naive Bayes Classifier (NB);

(4) Support vector machine (SVM) (with Gaussian radial basis function kernel and a default scaling factor of 1)

(5) K-nearest neighbor (KNN) (using $K = 5$ neighbors)

(6) Treebagger classifier (TREE) (using 10 trees)

(7) Multidimensional multiclass genetic programming classifier with multidimensional populations (M3GP) [31, 32]

ED and MD are linear classifiers that assume a Gaussian unimodal distribution for each class, an assumption that is clearly violated in this problem as seen in Figure 1. KNN and NB are well-known and widely used classifiers, mainly for their simplicity, ease of implementation, and competitive performance. However, the NB classifier assumption that each feature dimension can be treated independently is often violated in practice (although this is not the case in the problem studied here). SVM and TREE are state-of-the-art methods that perform strongly in many domains. In particular, the TREE classifier can handle multimodal classes and unbalanced datasets.

3.1. Multidimensional Multiclass GP with Multidimensional Populations: M3GP.

In this section, we introduce the method we previously developed and apply it to the problem of integrating the results of different prediction tools. More precisely, M3GP uses a GP-based search to achieve competitive performance on multiclass problems but also achieves strong results in binary tasks [32].

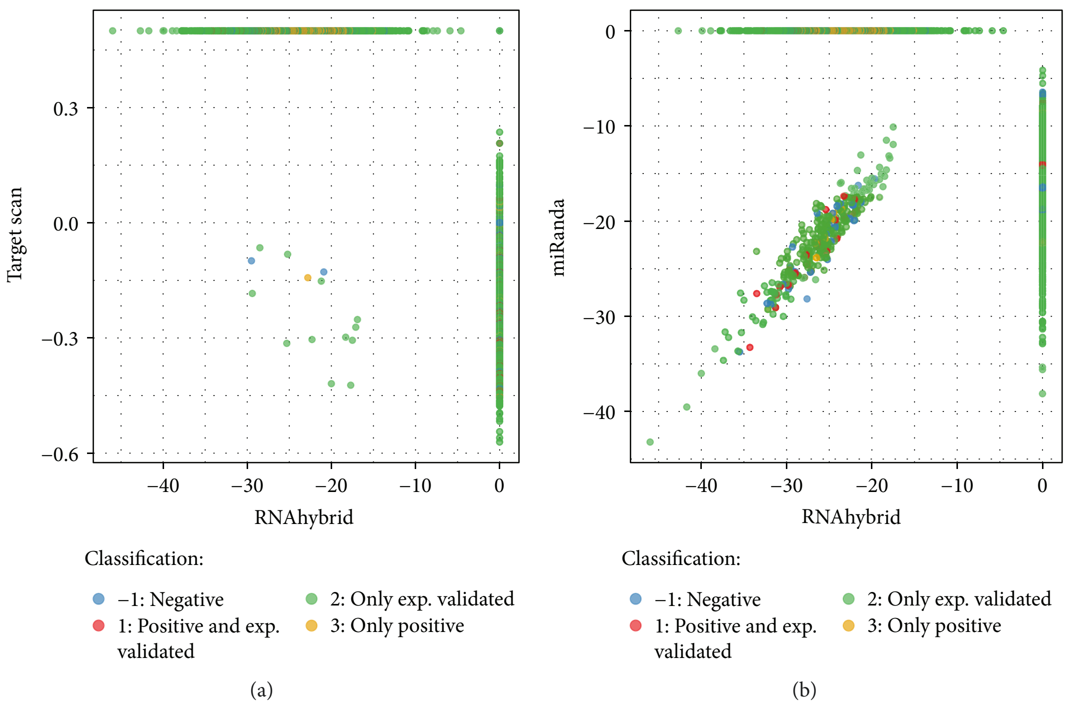
M3GP is based on the multiclass GP with multidimensional populations (M2GP) [31] that searches for a transformation, which is applied to the input data so that the transformed data of each class can be grouped into unique clusters and thus simplify the classification task. In M2GP, the number of dimensions in which the clustering process is performed is completely independent of the number of classes, such that high-dimensional datasets can be easily classified by a low-dimensional clustering, while low-dimensional datasets may be better classified by a high-dimensional clustering.

In order to achieve this, M2GP uses a representation of the solutions that allows performing, for each data point x, the mapping $k(\mathbf{x})\colon \mathbb{R}^p \rightarrow \mathbb{R}^d$, from an input feature space of p dimensions to a new one. The representation is basically the same used for regular tree-based GP, except that the root node of the tree exists only to define the number of dimensions d of the new space. Each branch stemming directly from the root performs the mapping in one of the new d dimensions. In M2GP, candidate solutions are evaluated as follows:

(1) All the p-dimensional samples of the training set are mapped to the new d-dimensional space (each branch of the tree is one of the d dimensions).

(2) In this new space, for each of the M classes in the data, the covariance matrix and the cluster centroid are calculated from the samples belonging to that class.

(3) the *Mahalanobis* distance between each sample and each of the M centroids is calculated. Each sample



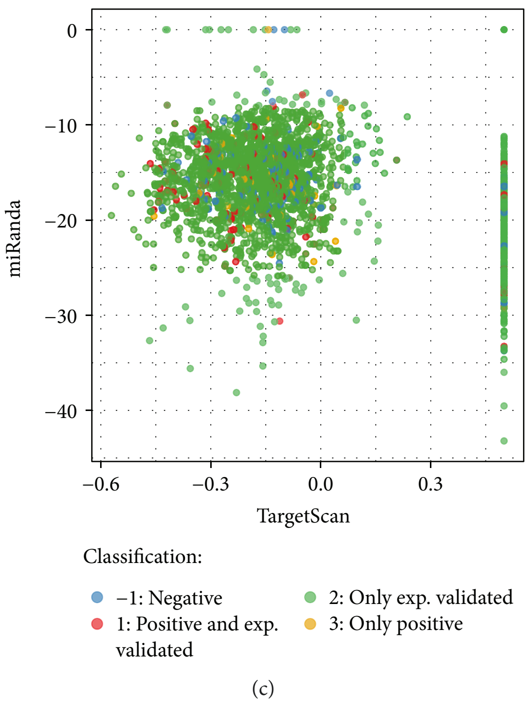


Figure 1: Scatter plots showing the distribution of the data: negative examples are represented by blue dots, while three different colors are used for positive examples. More precisely, red are those samples that are among the positive examples in the dataset from [29] and that are also experimentally validated, green for those only experimentally validated in miRTarBase, and yellow for the ones that are only in the positive examples of the dataset from [29].

is then assigned to the class having the closest centroid. Finally, the fitness function is given by the classification accuracy.

The original M2GP uses a greedy approach to determine how many dimensions the evolved solutions should have. It may happen that, by fixing the number of dimensions at the beginning of the run, the algorithm will be unable to find the best solutions during the search process with respect to those that may be found by using a different number of dimensions. Therefore, in the M3GP approach, a population that may contain transformation of different dimensions is evolved.

During the breeding phase, whenever the chosen genetic operator is a mutation, one of the three following actions is performed with equal probability: (i) a standard subtree mutation, where a randomly created new tree replaces a randomly chosen branch (excluding the root node) of the parent tree; (ii) a randomly created new tree is added as a new branch of the root node, effectively adding one dimension to the parent tree; and (iii) a complete branch of the root node is randomly removed, effectively removing one dimension from the parent tree.

On the other hand, whenever the chosen genetic operator is a crossover, one of the two following actions is performed with equal probability: (i) a standard subtree crossover, where a random node (excluding the root node) is chosen in each of the parents, and the corresponding branches are swapped, and (ii) the swapping of dimensions, where a random complete branch of the root node is chosen in each parent, and swapped between each other, effectively swapping dimensions between the parents. The latter event is just a particular case of the first one, where the crossing nodes are guaranteed to be directly connected to the root node.

M3GP also adds a pruning procedure that removes a random dimension and reevaluates the tree. If the fitness improves, the pruned tree replaces the original one and the procedure goes through the pruning of another dimension. Otherwise, the pruned tree is discarded and the original tree goes through the pruning of a new dimension. The procedure stops after each of the original dimensions was pruned and tested for improvement.

4. Results

As stated above, the problem is posed as a balanced binary classification task where the data for both classes were taken from the sets of positive and negative examples described in Section 2, by eliminating the duplicated entries. More precisely, 212 data samples were selected to represent class 1 (of negative examples labelled with 1) and 212 data samples were selected to represent class 2 (of positive examples, from those labelled with 1, 2, and 3). Subsequently, each of the selected methods for the performance comparison (i.e., ED, MD, NBC, SVM, KNN, TREE, and M3GP) was executed. The whole dataset was split into two partitions, a training set composed of 70% of the data and a test set with the remaining 30%. For each classifier, 30 independent runs were performed using a different random partition of the data in each run.

For M3GP, we used the parameter values specified in Table 2, following the indication of [32]. These values have been shown to produce accurate results, and since a preliminary tuning phase (performed with a grid search) showed no (statistically significant) change in terms of performance with respect to other configurations tested, we relied on these values. This is related to the fact that it has become increasingly clear that GP is very robust to parameter values once a good configuration is determined, as suggested by a recent and in-depth study [33].

Figure 2 provides a detailed view of the results by showing a comparison of the boxplots. In detail, Figure 2(a) shows the error obtained by the classifiers on the training data, while Figure 2(b) shows the error on the test set.

Table 2: Parameters used by M3GP to obtain the results reported in Figure 2.

Parameter	Value
Runs	30
Population size	500 individuals
Generations	100
Initialization	6-depth full initialization with 1 initial dimension
Operator probabilities	Crossover $p_c = 0.5$, mutation $p_\mu = 0.5$
Function set	(+, −, ×, ÷ protected as in [34])
Terminal set	Ephemeral random constants [0,1]
Bloat control	17-depth limit
Selection	Lexicographic tournament of size 5
Elitism	Keep best individual

Again, as anticipated in Section 3, these results confirm that all the considered classifiers achieve a very similar performance on unseen data, even if the training performance is better for some of them. In fact, we can state that SVM, KNN, and TREE overfit the training instances and, thus, their training performance is a poor indicator of performance on unseen instances. To statistically validate the results obtained, statistical comparisons are carried out using a $1 \times N$ formulation where a single control method (M3GP) is compared with N algorithms. We use the Friedman test and the Bonferroni-Dunn correction of the p values for each comparison. In all tests, the null hypothesis (that the medians are equal) is rejected at the $\alpha = 0.05$ significance level. According to the statistical validation, SVM and TREE are the techniques that are able to outperform M3GP on unseen instances. Specifically, the p values returned by the Friedman test are 0.0001 and 0.0016, respectively. All of the remaining techniques perform similarly than M3GP.

While SVM and TREE produce a better performance with respect to M3GP, it is important to highlight an advantage provided by M3GP with regard to the former methods. Specifically, M3GP is able to produce a human-understandable model, combining the problem features in a tree structure that can be interpreted by a domain expert, hence allowing for a better understanding of the relations between the features of the problem. This is a typical property of the large majority of GP-based systems. On the other hand, SVM does not have this property, providing a model that expresses the equation of the hyperplane that divides the two classes of the problem. TREE provides an ensemble of trees, and this makes it inherently different from the other techniques considered. For this technique, it is even difficult to identify what the final model is. For all these reasons, we believe that M3GP is a useful technique for addressing classification problems with two or more classes.

Although the results achieved by the tested methods may look similar, which was expected due to the difficulty of the problem (see Figure 1 for details of the distribution of the input data), we can highlight some aspects of the adopted approach. First of all, M3GP does not seem to be affected by overfitting or a lack of generalization after learning, with

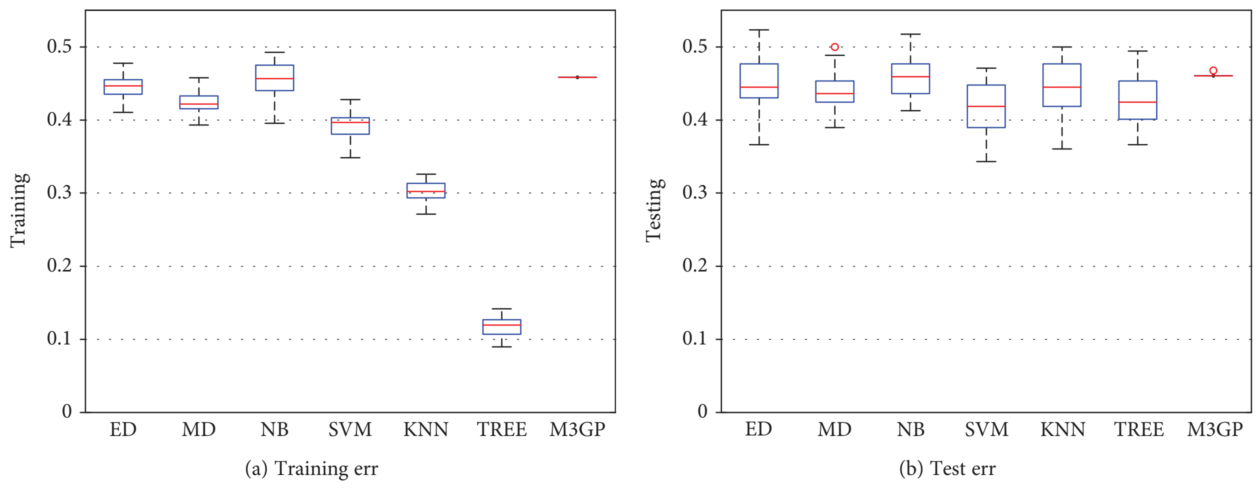


Figure 2: Boxplot reporting the errors of the tested classifiers in the both training (a) and test (b) sets.

an almost equal training and testing performance. Second, the method seems to be quite robust to the training data used, exhibiting the least amount of variance of all the methods. This result is particularly surprising since M3GP is a stochastic search procedure, like most other evolutionary algorithms. Indeed, a common feature of such methods is that their performance tends to vary from run to run, especially when different data partitions are used. This does not happen to M3GP in this domain, making the method an interesting and promising choice for the considered problem given that there can be a lot of noise in the data collection process.

4.1. M3GP Parallelization. Like most population-based search methods, and given the manner in which M3GP operates, the algorithm can be easily implemented in a parallel way by using different methods [35]. However, probably the most direct way to parallelize the algorithm is to evaluate each candidate solution in parallel since the fitness computation of each individual is independent, and this is referred to as the *global model* [36]. In the case of M3GP, this includes the data transformation stage and the results of the MD classification step.

Another approach is to change the basic GP paradigm, which is mostly used as a synchronous and local algorithm. For instance, it is possible to use the evolutionary model proposed in [37] to distribute the evolutionary process across various client machines (either locally or over the web) in a plug-and-play manner, thus increasing the available computational resources by exploiting cloud-based technologies. However, this option is left for possible future work.

Moreover, in the particular case of M3GP, there is another parallelization option. Since the individuals are composed of several independent subtrees joined by the "dummy" root node, it is quite easy to parallelize the evaluation of each individual subtree (dimension). In other words, each feature dimension in an M3GP individual can be evaluated independently.

In summary, three different parallel implementations of M3GP were developed and tested in this work (see Figure 3):

(i) Parallel population evaluation (*Pop Parallel*): this implementation parallelizes the evaluation of each solution as a whole, sending to each parallel worker independent solutions to process. This is a very common approach for evolutionary algorithms where the fitness evaluation of the population is divided to reduce the computational time.

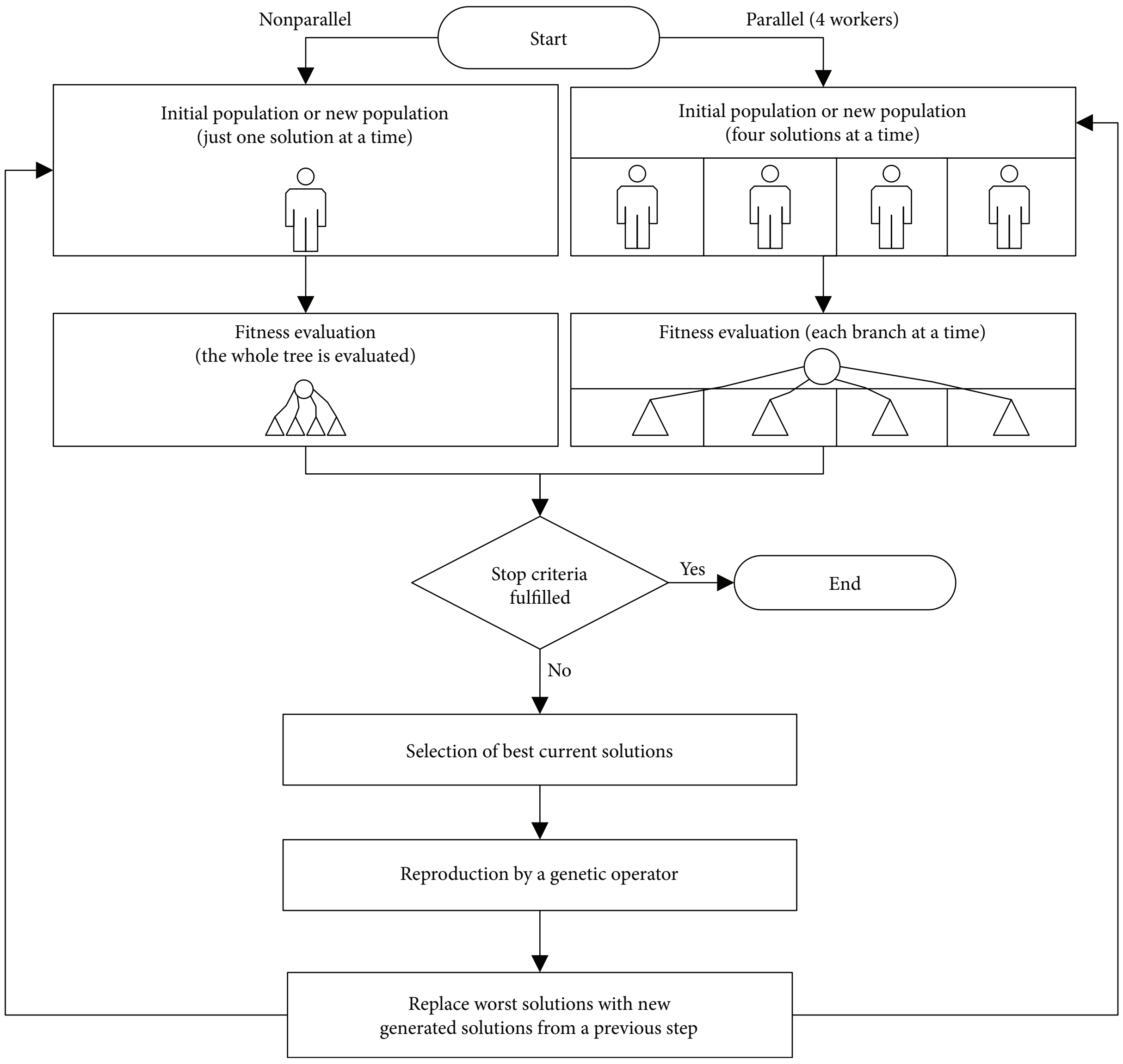
(ii) Parallel fitness evaluation (*Fitness Parallel*): this implementation parallelizes the evaluation of each subtree branch on the main root node, sending individual independent branches (features) to parallel workers. Thus, the evaluation of M3GP is parallelized based on the number of feature dimensions for each candidate solution.

(iii) Full parallel M3GP (*Full Parallel*): in this case, the two previous approaches are combined.

5. Discussion

Before describing the experimental analysis we performed to compare the three different implementations, we discuss some important aspects regarding each. More precisely, we want to highlight advantages and differences of the different approaches, also with respect to the *Non-Parallel* approach, from a computational complexity point of view. In particular, we focus on how the evaluation process can be parallelized. In the following analysis, we consider the computational cost of a single generation and for a fixed initial configuration of the algorithm parameters (see Table 3). The reason for this choice is that we want to focus on the differences in the described parallel approaches, discarding all operations that do not vary among them in our analysis.

The first approach, which is referred to as *Pop Parallel*, consists of evaluating each individual of the current population independently of the others and then joining the results



Start
Nonparallel
Parallel (4 workers)
Initial population or new population (just one solution at a time)
Initial population or new population (four solutions at a time)
Fitness evaluation (the whole tree is evaluated)
Fitness evaluation (each branch at a time)
Stop criteria fulfilled
Yes
End
No
Selection of best current solutions
Reproduction by a genetic operator
Replace worst solutions with new generated solutions from a previous step

Figure 3: Parallelization options for M3GP: an example with four independent threads (or workers). After the start, a nonparallel branch and a parallel one are shown. The latter highlights possible ways to parallelize the M3GP method, that is, (i) to evolve each solution independently, (ii) to compute each branch of the fitness tree separately, or (iii) to combine the two previous approaches.

Table 3: Parameters of M3GP used in all the parallel variants.

Parameter	Value
Runs	30
Population size	10,50,100, and 150 individuals
Generations	1
Initialization	12-Depth full initialization with 1, 5, 10, 20, and 30 dimensions
Operator probabilities	Crossover $p_c = 0.5$, mutation $p_\mu = 0.5$
Function set	$(+, -, \times, \div$ protected as in [34])
Terminal set	Ephemeral random constants [0,1]
Bloat control	17-Depth limit
Selection	Lexicographic tournament of size 5
Elitism	Keep best individual

to create the new population. As anticipated before, this is probably one of the most intuitive ways to apply parallelism to M3GP. Accordingly, the overall computational time in the worst-case scenario, that is, when all individuals require the same time to be evolved, is $(P/t)C$, where P is the population size (i.e., the total number of individuals), t is the number of threads, and $C = \max\{c_i\}$, with $1 \le i \le P$, is the maximum time required to evaluate an individual and c_i is the computational time necessary to evolve the i^{th} individual. Moreover, we also assume that $t \le P$ because having more threads than the number of individuals in the population will not bring any additional benefit to the parallelization since the excess threads will be unused.

Although this approach is very efficient from a theoretical point of view, some considerations must be made. In particular, one drawback is the memory consumption required to store all the individuals processed by the different threads. The number of nodes in the trees, which will impact the amount of memory required to store each tree, can lead to high memory consumption and a possible bottleneck. Unfortunately, in GP it is common for individual models to evolve larger than they need to be, which is referred to as the bloat phenomenon.

On the other hand, the second approach we took for parallelization is the fitness computation, which is split, in the *Fitness Parallel* implementation, among the different threads. In details, the evaluation takes as an input a tree composed of d subtrees, with d the number of dimensions of the transformed feature space of each individual in M3GP. Moreover, let F be the maximum cost necessary to evaluate one of the d subtrees connected to the root, that is, $F = \max\{f_i\}$, with $1 \le i \le d$, where f_i is the cost required to compute the fitness in the ith subtree.

The overall computational time to evaluate a single individual, which is the one required by the *Non-Parallel* approach, is $dF + d$, since in the worst case each of the d subtrees requires F time to be computed, plus an additional d time to combine these results. In any case, each subtree can be computed independently so in the Fitness Parallel implementation the computation is split into t threads. In this case, the cost is $(d/t)F + (d/t)\log t$, assuming that $t \le d$. In fact, by having more threads than dimensions of the search space will only waste the excessive computational power.

In more detail, assuming the worst-case scenario in which the computation of all the subtrees requires F time, we can split the d computations on t threads, leading to a $(d/t)F$ time. Further, we can also take advantage of the parallelism to recombine these results so that this process is executed in a binary-tree structure. In this way, the d results are recursively recombined in pairs, which are distributed in t threads, so that the overall time for recombining the results is $(d/t)\log t$. Finally, as is easy to observe, when $t = 1$, we obtain the same complexity of the *Non-Parallel* approach, that is, $dF + d$.

5.1. Parallel Experiments. M3GP was implemented using the GPLAB toolbox in Matlab [], which is freely available at http://gplab.sourceforge.net. Each parallel M3GP variant was implemented using the parallel computing tools in Matlab. The experiments were carried out on a workstation with an 8-core CPU and 16 GB of RAM.

In our experiments, the goal was to evaluate the relative improvement of the parallel approach in conditions that required a high computational cost. After some preliminary experimental runs, the configuration reported in Table 3 was used to stress the search process and illustrate the benefits of the different parallel approaches. In particular, it is noted that the highest initial tree depth was set to 12 levels, making a total of 2,048 nodes for each dimension in an evolved transformation. This makes the required computational cost of evaluating each individual feature significant, and evaluating each individual is more costly when increasing the total number of dimensions. Moreover, we compared the total computational time using different population sizes and a different number of total initial dimensions in the population. For the sake of comparison, M3GP without any parallelization (referred to as *Non-Parallel*) was run with the same settings as a baseline.

A parallel environment for the experiments was set up to check for time improvements in just one generation. Letting the algorithm run for a complete search will just increase the computational cost linearly since each generation is independent of the previous one, in particular when the current solution is already close to the maximum allowed depth. For a fair comparison, the same random seeds were used for all parallel M3GP variants so that the random tree generation process did not influence the results.

5.2. Comparison of the Different Parallel Approaches. The results are summarized in the plots of Figure 4 where for each run, the time required to evaluate a population for the sequential M3GP and all the three parallel versions (i.e., *Pop Parallel*, *Fitness Parallel*, and *Full Parallel*) is shown. The x-axis in each plot corresponds to the number of dimensions of each individual, while each plot corresponds to a different population size (i.e., 10, 50, 100, and 150).

Running M3GP with just one dimension (one branch below the root node) does not provide any substantial benefit for the parallelization. In this case, with different population sizes, the fastest evaluation was the *Non-Parallel* M3GP. The reason is that in the *Fitness Parallel* version the majority of the time was spent sending data to different workers and then merging the computed results.

A tendency that emerged from the results we obtained and that is easy to observe in all the plots is that increments in the population size do not substantially affect the trend of the total computational time. In fact, it is almost linear in all cases considered, although the (absolute) time required to process bigger populations is greater than that required for processing smaller populations.

On the other hand, by increasing the number of dimensions, the overall workload entailed in evaluating each solution increases. The parallel implementations start to show significant improvements when five dimensions are used, and they continuously improve when more dimensions are added.

This is particularly true of the *Fitness Parallel* implementation, which tends to require less time than the other variants do. In all analyzed cases, only the *Fitness Parallel* variant requires less processing time than the sequential *Non-Parallel* M3GP does. It is evident that when increasing the population size and the number of dimensions, *Full Parallel* and *Pop Parallel* start to perform worse with each increment.

5.3. M3GP Speed-Up. To assess the speed-up of the different M3GP systems we proposed, the implementations were tested with up to 8 threads, and the results are shown in Figures 5 and 6. The values reported are calculated as the ratio between the run time of the *Non-Parallel* version over the parallel ones. Then, by varying the number of threads, we can measure the gain in terms of speed.

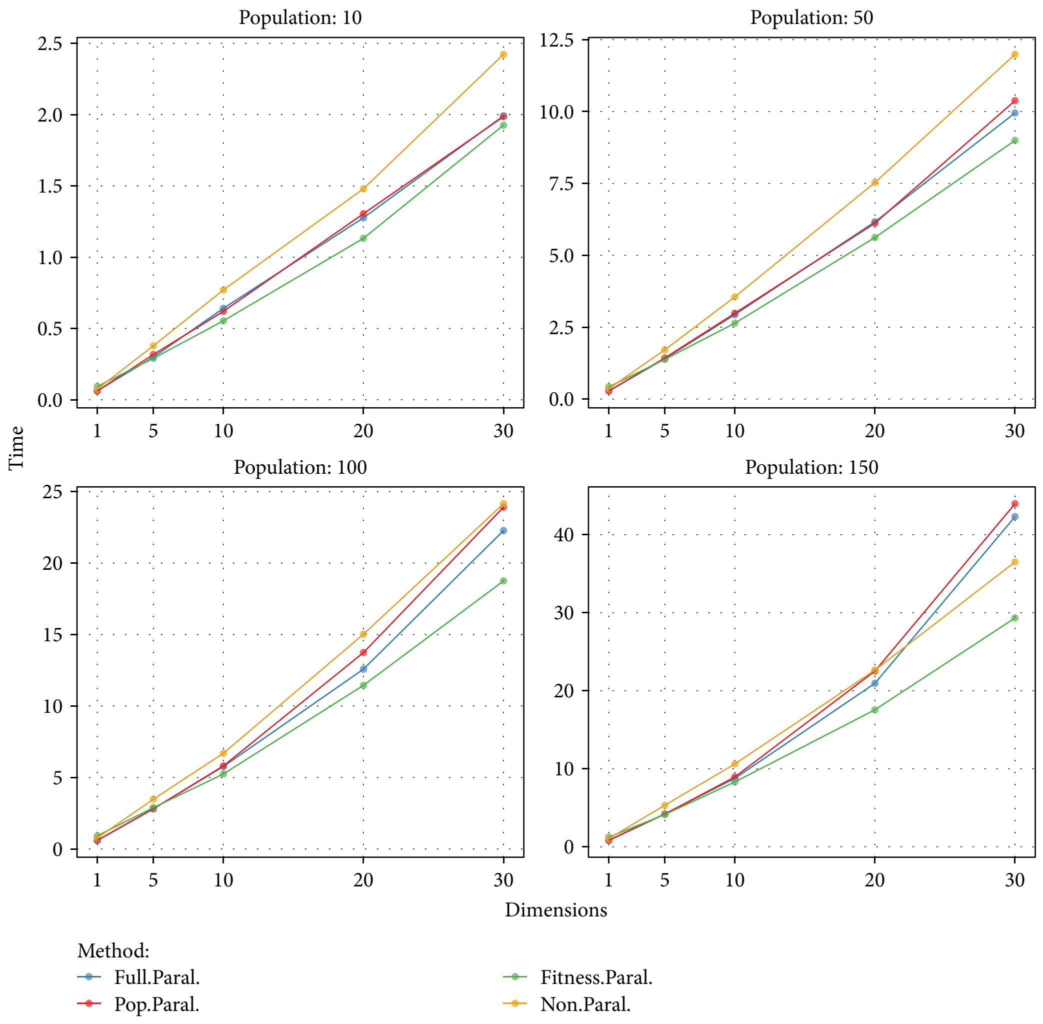


FIGURE 4: Parallel runs with M3GP, time (in minutes) versus dimensions.

In the plots in Figure 5, we reported the speed-up of the three parallel implementations of M3GP for individuals with 10 dimensions, tested over different values of the population size: 10, 50, 100, and 150. There is a performance gain when using 2 or more threads. In fact, when adopting only one thread, we have values lower than 1 because the parallel runs required additional operations, with respect to the *Non-Parallel* one, leading to this loss in performance. The *Full Parallel* and *Pop Parallel* perform best in this setup, with *Parallel Fitness* always performing slightly worse, except for the simplest case with the smallest population. However, it is still necessary to evaluate how this performance generalizes when evolving larger transformations with more dimensions.

Therefore, we performed similar tests by fixing the population size to 100 and varying the number of dimensions in the individual transformations. In Figure 6, we show the plots for 1, 5, 10, 20, and 30 dimensions. Not surprisingly, in the first case with only a single dimension, the *Parallel Fitness* implementation does not take advantage of the increasing number of threads since its parallelization is based on the number of dimensions. In the other cases, we can observe that by increasing the number of dimensions, this implementation improves its speed-up, achieving the best performance when the number of dimensions reaches 30.

Finally, although Matlab allows a very fast and useful way to realize and test the M3GP method (also providing an easy way to implement a parallel algorithm), we think that a more efficient implementation could lead to a significant performance improvement.

6. Conclusions

In this work, we applied a novel GP-based classification method called M3GP to the problem of integrating the results of different miRNA-target prediction tools, namely, miRanda, TargetScan, and RNAhybrid, and classifying them. Moreover, we derived a parallel implementation of this algorithm to reduce the computational cost of the search when evolving large multidimensional transformations.

The results we obtained with the M3GP method are promising and competitive when compared with the ones achieved by the other classifiers we tested (Euclidean distance, Mahalanobis distance, naive Bayes, SVM, KNN, and

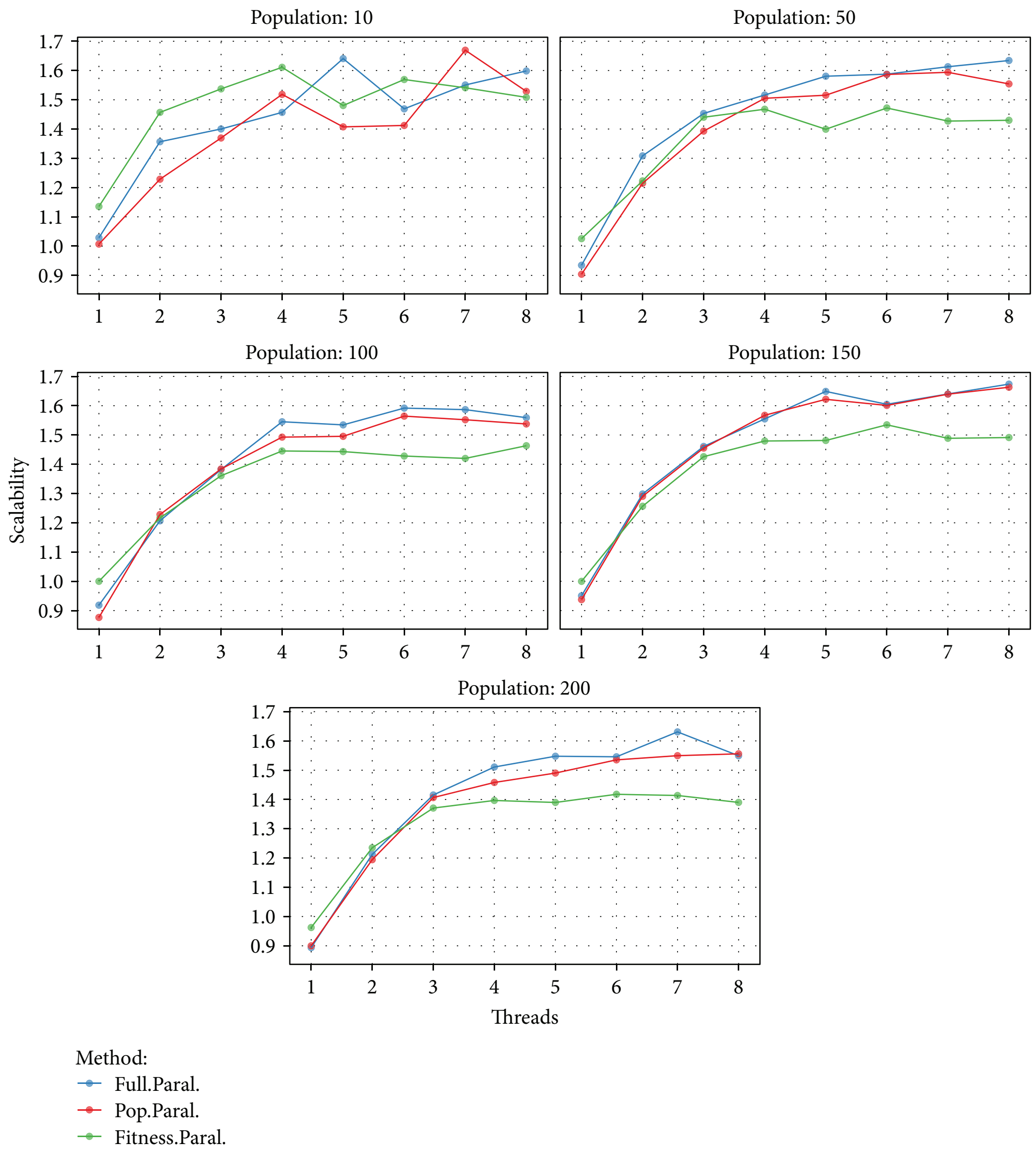


Figure 5: Speed-up plots for solutions having 10 dimensions, with population sizes of 10, 50, 100, and 150 individuals, and using from 1 to 8 threads.

Treebagger). The main advantage in contrast with other machine learning techniques is that the performance of M3GP is more robust than that of the other methods. Specifically, in this domain M3GP shows almost no overfitting or lack of generalization relative to the training performance and is robust to the training set used since its performance variance is almost null.

This makes the M3GP method a good candidate for solving the problem at hand. We also showed that M3GP is easily parallelizable. In fact, like most evolutionary algorithms, M3GP can be parallelized during fitness evaluation, or by assigning each solution to a different thread, or by combining these two approaches. However, it was shown that the best strategy is to parallelize the evaluation of every single individual by sending each tree branch (feature dimension) of the root node to a different thread. This can be done by virtue of the way in which M3GP individuals construct their output response. This approach showed substantial improvements when compared to the sequential approach and to other parallelization options.

One of the possible future research directions will focus on exploiting the parallelization aspects by adopting a more efficient implementation of the M3GP method and also by testing its execution in a distributed environment. Moreover, to further improve the robustness of the results and the strength of the approach, we plan to integrate the results of other miRNA-target prediction tools (in addition to the ones we considered here) but also to employ other datasets, such as [38]. As a final remark, we would like to point out that for future study, one can employ additional target sequences, such as the 5-UTR of the genes and the coding part of the transcripts.

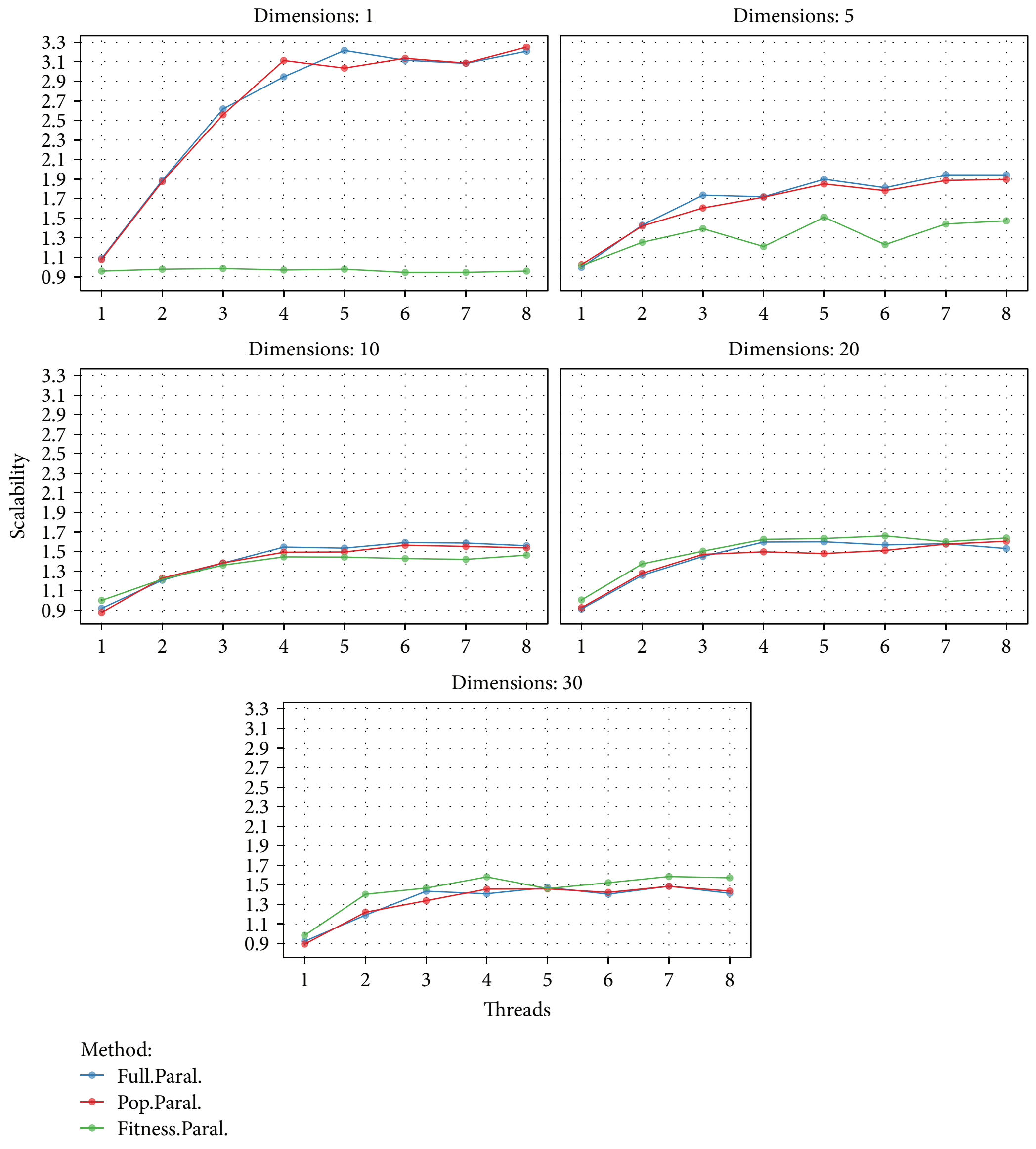


Figure 6: Speed-up plots for a population size of 100, with individuals of 1, 5, 10, 20, and 30 dimensions and using from 1 to 8 threads.

Data Availability

As described in the article, we used data available at the following website: http://www.targetscan.org/.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

Acknowledgments

This research was partially funded by Consejo Nacional de Ciencia y Tecnología (Mexico) Fronteras de la Ciencia 2015-2 Project no. FC-2015-2:944, and the third author was supported by Consejo Nacional de Ciencia y Tecnología graduate scholarship 401223.

References

[1] S. Manvati, K. C. Mangalhara, J. Khan et al., “Deciphering the role of microRNA – a step by step guide,” *Gene Expression Patterns*, vol. 25-26, pp. 59–65, 2017.

[2] Y. Peng and C. M. Croce, “The role of microRNAs in human cancer,” *Signal Transduction and Targeted Therapy*, vol. 1, no. 1, article 15004, 2016.

[3] L. He, X. He, L. P. Lim et al., “A microRNA component of the p53 tumour suppressor network,” *Nature*, vol. 447, no. 7148, pp. 1130–1134, 2007.

[4] M. Leclercq, A. B. Diallo, and M. Blanchette, “Prediction of human miRNA target genes using computationally reconstructed ancestral mammalian sequences,” *Nucleic Acids Research*, vol. 45, no. 2, pp. 556–566, 2017.

[5] C. P. C. Gomes, J. H. Cho, L. Hood, O. L. Franco, R. W. Pereira, and K. Wang, “A review of computational tools in microRNA discovery,” *Frontiers in Genetics*, vol. 4, p. 81, 2013.

[6] X. Fan and L. Kurgan, "Comprehensive overview and assessment of computational prediction of microRNA targets in animals," *Briefings in Bioinformatics*, vol. 16, no. 5, pp. 780–794, 2015.

[7] P. K. Srivastava, T. Moturu, P. Pandey, I. T. Baldwin, and S. P. Pandey, "A comparison of performance of plant miRNA target prediction tools and the characterization of features for genome-wide target prediction," *BMC Genomics*, vol. 15, no. 1, p. 348, 2014.

[8] M. M. Akhtar, L. Micolucci, M. S. Islam, F. Olivieri, and A. D. Procopio, "Bioinformatic tools for microRNA dissection," *Nucleic Acids Research*, vol. 44, no. 1, pp. 24–44, 2016.

[9] M. Kertesz, N. Iovino, U. Unnerstall, U. Gaul, and E. Segal, "The role of site accessibility in microRNA target recognition," *Nature Genetics*, vol. 39, no. 10, pp. 1278–1284, 2007.

[10] T.-P. Lu, C.-Y. Lee, M.-H. Tsai et al., "miRSystem: an integrated system for characterizing enriched functions and pathways of microRNA targets," *PLoS One*, vol. 7, no. 8, article e42390, 2012.

[11] C. E. Vejnar and E. M. Zdobnov, "MiRmap: comprehensive prediction of microRNA target repression strength," *Nucleic Acids Research*, vol. 40, no. 22, pp. 11673–11683, 2012.

[12] M. D. Paraskevopoulou, G. Georgakilas, N. Kostoulas et al., "DIANA-microT web server v5. 0: service integration into miRNA functional analysis workflows," *Nucleic Acids Research*, vol. 41, no. W1, pp. W169–W173, 2013.

[13] C. Coronnello and P. V. Benos, "ComiR: combinatorial micro-RNA target prediction tool," *Nucleic Acids Research*, vol. 41, no. W1, pp. W159–W164, 2013.

[14] H. Dweep et al.M. Alvarez and M. Nourbakhsh, "miRWalk database for miRNA–target interaction," in *RNA Mapping. Methods in Molecular Biology (Methods and Protocols), vol 1182*Humana Press, New York, NY, USA.

[15] A. Krek, D. Grün, M. N. Poy et al., "Combinatorial microRNA target predictions," *Nature Genetics*, vol. 37, no. 5, pp. 495–500, 2005.

[16] A. J. Enright, B. John, U. Gaul, T. Tuschl, C. Sander, and D. S. Marks, "MicroRNA targets in *drosophila*," *Genome Biology*, vol. 5, no. 1, article R1, 2003.

[17] B. John, A. J. Enright, A. Aravin, T. Tuschl, C. Sander, and D. S. Marks, "Human microrna targets," *PLoS Biology*, vol. 2, no. 11, article e363, 2004.

[18] B. P. Lewis, I.-h. Shih, M. W. Jones-Rhoades, D. P. Bartel, and C. B. Burge, "Prediction of mammalian microRNA targets," *Cell*, vol. 115, no. 7, pp. 787–798, 2003.

[19] R. C. Friedman, K. K.-H. Farh, C. B. Burge, and D. P. Bartel, "Most mammalian mRNAs are conserved targets of micro-RNAs," *Genome Research*, vol. 19, no. 1, pp. 92–105, 2009.

[20] M. Rehmsmeier, P. Steffen, M. Hochsmann, and R. Giegerich, "Fast and effective prediction of microRNA/target duplexes," *RNA*, vol. 10, no. 10, pp. 1507–1517, 2004.

[21] S. Nam, B. Kim, S. Shin, and S. Lee, "miRGator: an integrated system for functional annotation of microRNAs," *Nucleic Acids Research*, vol. 36, Supplement 1, pp. D159–D164, 2008.

[22] E. R. Gamazon, H.-K. Im, S. Duan et al., "Exprtarget: an integrative approach to predicting human microrna targets," *PLoS One*, vol. 5, no. 10, article e13534, 2010.

[23] R. Edgar, M. Domrachev, and A. E. Lash, "Gene expression omnibus: Ncbi gene expression and hybridization array data repository," *Nucleic Acids Research*, vol. 30, no. 1, pp. 207–210, 2002.

[24] K. D. Kaya, G. Karakülah, C. M. Yakıcıer, A. C. Acar, and Ö. Konu, "mESadb: microRNA expression and sequence analysis database," *Nucleic Acids Research*, vol. 39, Supplement 1, pp. D170–D180, 2011.

[25] D. Bielewicz, J. Dolata, A. Zielezinski et al., "mirEX: a platform for comparative exploration of plant pri-miRNA expression data," *Nucleic Acids Research*, vol. 40, no. D1, pp. D191–D197, 2012.

[26] G. Sales, A. Coppe, A. Bisognin, M. Biasiolo, S. Bortoluzzi, and C. Romualdi, "MAGIA, a web-based tool for miRNA and genes integrated analysis," *Nucleic Acids Research*, vol. 38, Supplement 2, pp. W352–W359, 2010.

[27] D. Corrada, F. Viti, I. Merelli, C. Battaglia, and L. Milanesi, "myMIR: a genome-wide microRNA targets identification and annotation tool," *Briefings in Bioinformatics*, vol. 12, no. 6, pp. 588–600, 2011.

[28] S. Beretta, M. Castelli, Y. Marítnez et al., "A machine learning approach for the integration of mirna-target predictions," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 528–534, Heraklion, Greece, Febuary 2016.

[29] S. Bandyopadhyay and R. Mitra, "TargetMiner: microRNA target prediction with systematic identification of tissue-specific negative examples," *Bioinformatics*, vol. 25, no. 20, pp. 2625–2631, 2009.

[30] S.-D. Hsu, Y.-T. Tseng, S. Shrestha et al., "miRTarBase update 2014: an information resource for experimentally validated miRNA-target interactions," *Nucleic Acids Research*, vol. 42, pp. D78–D85, 2014.

[31] V. Ingalalli, S. Silva, M. Castelli et al.K. Krawiec, M. I. Heywood, M. Castelli et al., "A multi-dimensional genetic programming approach for multi-class classification problems," in *17th European Conference on Genetic Programming, volume 8599 of LNCS*, M. Nicolau, Ed., pp. 48–60, Springer, Granada, Spain, 2014.

[32] L. Munoz, S. Silva, L. Trujillo et al.M. I. Heywood, J. McDermott, M. Castelli et al., "M3GP: multiclass classification with GP," in *18th European Conference on Genetic Programming, volume 9025 of LNCS*, pp. 78–91, Springer, Copenhagen, 2015.

[33] M. Sipper, F. W, K. Ahuja, and J. H. Moore, "Investigating the parameter space of evolutionary algorithms," *BioData Mining*, vol. 11, no. 1, p. 2, 2018.

[34] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT press, 1992.

[35] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity*, vol. 4, no. 4, pp. 31–52, 1999.

[36] P. Angeline and K. Kinnear, *Massively Parallel Genetic Programming*, MIT Press, 1996.

[37] M. García-Valdez, L. Trujillo, J.-J. Merelo, F. F. de Vega, and G. Olague, "The EvoSpace model for pool-based evolutionary algorithms," *Journal of Grid Computing*, vol. 13, no. 3, pp. 329–349, 2015.

[38] A. Ruepp, A. Kowarsch, D. Schmidl et al., "PhenomiR: a knowledgebase for microRNA expression in diseases and biological processes," *Genome Biology*, vol. 11, no. 1, article R6, 2010.