Machine Learning for Determining the Transition Point in Hybrid Metaheuristics

Antonio Bolufé-Röhler
Mathematical and Computational Sciences
University of Prince Edward Island
Charlottetown, Canada
ORCID 0000-0002-3181-1864

Ye Yuan
Mathematical and Computational Sciences
University of Prince Edward Island
Charlottetown, Canada
yeyuan2@upei.ca

2021 IEEE Congress on Evolutionary Computation (CEC) | 978-1-7281-8393-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/CEC45853.2021.9504905

***Abstract*—High-level relay hybrids are among the most effective metaheuristics in multiple domains. However, the relay aspect of hybridization raises the problem of when to perform the transition from one algorithm to the next. This problem becomes more relevant in exploration-only exploitation-only hybrids, where each algorithm specializes in a specific task and performs rather poorly in the other. This paper presents a novel way of approaching the transition problem as a classification problem. Different classifiers are trained and tested on the MPS-CMAES hybrid; computational results are presented for the CEC'13 benchmark. The performance of the machine learning based hybrid confirms the effectiveness of the approach by achieving a significant improvement over the original hybrid.**

***Index Terms*—metaheuristics, machine learning, hybrid heuristics, Minimum Population Search, CMA-ES**

I. Introduction

The optimization of multimodal problems involves two distinct tasks: identifying promising attraction basins and finding the local optima in these basins. Many metaheuristics attempt to "transition" a single algorithm from the first task, namely exploration, to the second task, exploitation. Previous research has shown that it would be better for the two distinct tasks to instead be performed by two distinct algorithms. Such hybrids assign these tasks to specialized algorithms; an exploratory algorithm usually performs the first task while the second task is assigned to a local search method [1].

Many such hybrids follow a high-level relay hybridization strategy; i.e. the exploration algorithm is executed first, and local search starts from the solution(s) provided by the exploration algorithm [2]. Determining exactly when to stop the exploration algorithm and transition to local search becomes a critical decision in the design of such hybrids. Different factors influence when this moment is reached. For instance, when optimizing a unimodal function, all solutions belong to the same attraction basin; thus, it is convenient to perform an early transition to exploitation. Conversely, for multimodal functions it is necessary to find the best attraction basins; and exploration should be extended until these basins are detected.

In this paper we will use the definitions of exploration and exploitation presented in [3]. According to these definitions a search point is defined to be performing *exploration* if it is in a different attraction basin than its reference solution. It is defined to be performing *exploitation* if it is in the same attraction basin as its reference solution. Reference solutions are those against which new solutions are compared to when performing selection.

The most frequent approach among high-level relay hybrids consists of fixing a transition point, which becomes a parameter that needs to be tuned. However, a fixed transition point is far from ideal since the best moment to start the transition depends on the objective function being optimize; concretely, it will depend on the topological characteristics of the function. Because of the intrinsic randomness of heuristic search, it may also vary from one execution to another even for the same objective function [4].

Determining at runtime the topology of a function is a difficult problem. It requires analyzing the results produced during the exploratory phase in order to make a decision. This paper addresses the problem in a novel way by modeling it as a binary classification problem, and using machine learning classifiers to predict the optimum transition point. Information collected during the execution of the exploratory algorithm is provided to the classifier at regular intervals. Based on this information, the classifier decides whether to start the transition to exploitation (local search) or not.

This paper begins with a background on hybridization. The transition problem is then defined as a classification problem in Section III. Section IV presents results for the classifiers, while Section V presents the optimization results for the hybrids. A discussion concludes the paper in Section VI.

II. Background

The combination of metaheuristics with each other, and with components of other fields, is currently one of the most successful trends in optimization [5]. The success of hybrid metaheuristics has led to a large number of publications documenting the benefits of such hybrids [6]; and in some fields like Large-Scale Global Optimization, hybrid metaheuristics are state of the art [1], [7].

A novel research line in hybrid metaheuristics is the design of exploration-only exploitation-only hybrids [2], [4]. In such hybrids the processes of exploration and exploitation are

explicitly separated. The focus is in the design of "pure exploration" algorithms that can improve exploration by avoiding the negative effects of concurrent exploration and exploitation in the early stages of the search [8].

Minimum Population Search (MPS) is one such algorithm that was designed with the aim of achieving pure exploration, and it has been successfully used in hybrids for a variety of optimization problems [1], [9], [10]. However, the exploration-only exploitation-only approach reinforces the importance of finding the best transition point. An exploration-only algorithm would perform very poorly on unimodal functions (or a unimodal/plateau region of a multimodal function). Conversely, an exploitation-only algorithm would converge prematurely and fail to find the best attraction basins in a multimodal function.

Previous work on determining the transition point of exploration-only exploitation-only hybrids has mostly relied on fixed point transition [11], complex adaptive rules based on restarts [1], and only recently a machine learning approach has been attempted [4]. In [4] the transition problem is defined for the first time as a classification problem, but it is done in the context of combinatorial optimization; specifically for the problem of finding S-boxes with good cryptographic properties. The current paper extends the result for global optimization, and more importantly, tests the machine learning hybrid in a diverse set of benchmark functions; assessing the classifiers on a variety of multimodal and unimodal functions.

A. Minimum Population Search

Minimum Population Search is an algorithm explicitly designed for multimodal functions. The key ideas were initially developed for two-dimensional problems using only two population members in [12], later generalized for standard dimensions in [13], and scaled towards large scale problems in [14]. To preserve the diversity of the (small) population and avoid premature convergence, MPS includes the *Thresheld Convergence* (TC) technique as part of its original design.

Thresheld Convergence is a diversification technique that attempts to separate exploration and exploitation. TC uses a "threshold" function to establish a minimum search step, and managing this step makes it possible to control the transition from exploration to exploitation. Thus, convergence is "held" back until the last stages of the search process (thus the name *Thresheld*). TC has been successfully applied to many population-based metaheuristics (e.g. [15], [16]).

Solutions in MPS are sampled by taking a step away from a parent solution towards the centroid of the population and then taking another step orthogonal to the centroid step. The length of each step guarantees that the new solution is at least a threshold distance away from its parent. The threshold is initially set to a fraction of the search space diagonal and is updated by following the decay rule (1). In this equation, d is the main diagonal of the search space, $totalFEs$ is the total number of function evaluations, and FEs the evaluations performed so far. The α parameter determines the initial threshold and γ controls the decay rate.

$$minStep = \alpha * d * \left(\frac{totalFEs - FEs}{totalFEs}\right)^{\gamma} \quad (1)$$

B. The MPS-CMAES Hybrid

The chosen hybrid for this paper uses Minimum Population Search as the exploration algorithm and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for the exploitation phase. MPS has proven to be an effective exploratory method [2], [12] successfully used in multiple hybrids. One such hybrid is MPS-CMAES [1], which a comprehensive third-party study [9] has rated as the second-best of 15 tested methods for Large-Scale Global Optimization. Similarly, CMA-ES is an algorithm with a proven record in the optimization of a variety of functions, capable of conducting a very efficient local search with the right set of parameters [17].

The hybrid starts with MPS and then transitions to CMA-ES once the budget of function evaluations for the exploratory phase is depleted. CMA-ES begins from the best found solution using the remaining function evaluations. Algorithm 1 shows the primary hybrid that uses a fixed point transition represented by a given amount of function evaluations devoted for exploration ($explorationFEs$). CMA-ES is executed using *TolX* and *TolFun* as termination criteria with the values suggested in [17] and code from [18]. If CMA-ES converges before consuming the whole budget of evaluations then it is restarted from the next best solution reported by MPS.

Algorithm 1 MPS-CMA-ES ($maxFEs, explorationFEs$)

```
best_sol = MPS(explorationFEs)
FEs = explorationFEs
while FEs ≤ maxFEs do
    for best in best_sol do
        CMA-ES(best)
    end for
end while
return best found solution
```

In the hybrid we use a standard implementation of Minimum Population Search with $\alpha = 0.3$, $\gamma = 3$ and a population size equal to the dimensions of the objective function, as recommended in [1]. CMA-ES runs with the default parameter values as recommended in [17]; except for σ, which is selected to be much smaller, as to achieve a local search behavior. As recommended in the MPS-CMAES hybrid [1], σ is set to a value 100 times smaller than the default parameter, i.e. $\sigma = \frac{u\_bound - l\_bound}{300}$ (where u_bound and l_bound are the upper and lower bound respectively).

III. The Problem of Transition as a Classification Problem

The problem of finding the optimal transition point can be modeled as a classification problem, which can be solved using machine learning techniques. The idea is to call a classification algorithm at regular intervals while running MPS. This classifier will receive input information about the optimization process, and based on that information, it will predict (classify) if the current moment is or is not ideal for initiating the transition.

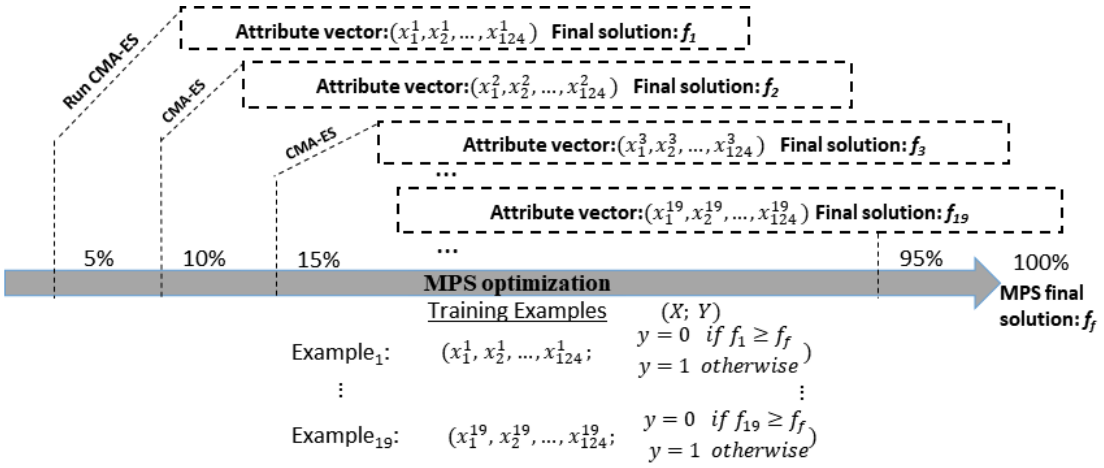

Fig. 1. Procedure for creating the training examples. MPS is executed and paused at regular intervals, information about the optimization process is gathered and CMA-ES is started. If the final result obtained by CMA-ES is better than the final result achieved by MPS without hybridization, then the label is set to 'yes' ($y = 1$) indicating that the transition is recommended at this point.

A. Creating the Training Examples

The appropriate training data needs to be created in order to train the classifier. The output label (target feature) is binary, 'yes' or 'no', depending on whether the transition should be started. The input will consist of 124 features characterizing the optimization process of MPS. The training data consists of pairs formed by the input vector $(x_1, x_2, ..., x_{124})$ and the corresponding output label y.

To collect the data MPS is paused 19 times during its execution at 5% intervals. Previous research has used 10% intervals for collecting the data; in this paper we opted to have more frequent intervals to promote a more accurate prediction of the transition point. These intervals could become arbitrarily small, but there is a trade-off between the frequency and the computational cost when creating the training examples and running the optimization algorithms.

At each of these intervals, information about the state of the algorithm is gathered. The state is represented by the following 124 scalar attributes:

- Feature 1: The function evaluations used so far.
- Feature 2: The function evaluations available.
- Feature 3: The minimum step of Thresheld Convergence in MPS.
- Feature 4: The maximum step of Thresheld Convergence in MPS (the double of the minimum step as in [12]).
- Features 5-34: The fitness of the best solution in MPS population in each of the previous 30 generations.
- Features 35-64: The fitness of the median in MPS population in each of the previous 30 generations.
- Features 65-94: The fitness of the worst solution in MPS population in each of the previous 30 generations.
- Features 95-124: The number of new solutions added to the MPS population in the previous 30 generations.

At each of these intervals, after the input vector data is gathered, CMA-ES is started from the best solution using the remaining function evaluations. A training pair is created with the gathered input vector and labeled as $y = 1$ if the result of CMA-ES was better than the final result obtained after resuming MPS (with no hybridization), or labeled $y = 0$ otherwise. By following this procedure, each execution of MPS yields 19 different training examples. Figure 1 shows a visual description of this process.

The training data was gathered using the 28 functions from the IEEE CEC'13 benchmark [19]. The functions are divided into three sets: unimodal functions (1 to 5), basic multi-modal functions (6 to 20) and composite multi-modal functions (21 to 28). For each function MPS was executed 100 times, including the 19 CMA-ES runs started at each interval. Thus, the dataset contains $19 * 100 * 28 = 53200$ training examples.

B. The Machine Learning Based Hybrid

The trained classifier is used in the MPS-CMAES hybrid to determine when to transit from MPS to CMA-ES. Information about the MPS optimization process is continuously gathered and at regular intervals (of 5% of function evaluations) the classifier is called. The classifier is passed the gathered information and predicts whether to perform the transition. Once the transition ocurrs, no more information is gathered and CMA-ES completes the execution of the hybrid in a similar way as it was done in Algorithm 1. The codes, data and results are available in [20].

IV. Computational Results on Classification

We tested the following classifiers: K-Nearest Neighbour (KNN), Gaussian naive Bayes (GNB), Gaussian Support Vector Machine (SVM) and Decision Tree (DT). These classifiers were chosen because of their fast training and prediction speeds, and small memory storage. The experiments were performed using MATLAB's classifier learning app, the full set of generated training examples and a holdout-validation set with 25% of the data set.

Table I shows the accuracy of the different models on the validation set. It can be noticed that DT and SVM perform

TABLE I
PERFORMANCE OF DIFFERENT CLASSIFIERS

Classifier	Accuracy
Decision Tree	**85.0**%
Gaussian SVM	**67.8**%
Gaussian naive Bayes	64.4%
K-Nearest Neighbour	62.9%

better that GNB and KNN; thus, we decided to focus our analysis on these two classifier. The DT classifier is a Fine Tree Classifier with the maximum number of splits at 100. The SVM classifier corresponds to a Fine Support Vector Machine classifier with a Gaussian kernel [18].

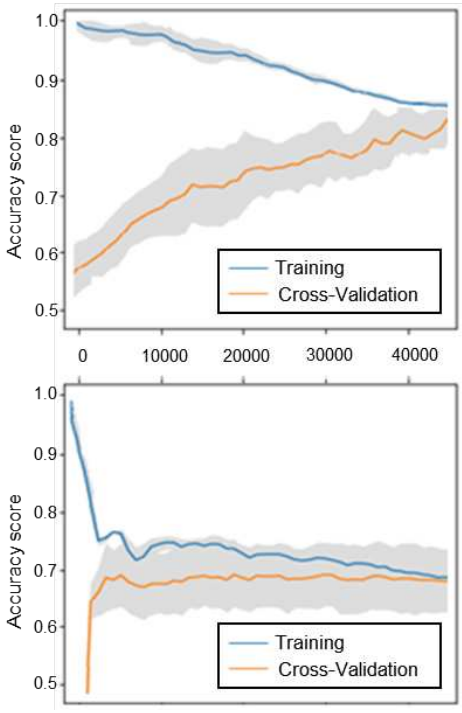

Fig. 2. Learning curves for Support Vector Machine and Decision Tree Classifier.

Figure 2 shows the learning curves for these two classifiers. The learning curves for the SVM classifier show that both the validation and the training score converge quickly to a value close to the final 67.8% accuracy. The small gap between the curves, the rather low score for a binary classification problem, and a quick convergence that doesn't seem to improve with further data, all suggest that this classifier suffers from a bias error.

The learning curves for the DT classifier show a slower convergence towards a higher score value. Taking into consideration that the validation curve keeps improving until the end, and that both curves don't fully converge, it seems reasonable to suggest that this model is affected by variance error (although not a particularly high one). It also suggests that the model could benefit from a larger data set.

A. Testing the Classifiers on the Hybrid

The Decision Tree and the Support Vector Machine classifiers are the best among the tested classifiers; however, there is still a statistically significant difference in their performance. This was confirmed by conducting a mid-p-value McNemar test, which rejected the null hypothesis of predicted class labels having equal accuracy at the 5% significance level. This raises the question of whether such a difference in classification performance will translate into the optimization results, once the classifiers are plugged into the machine learning hybrid. To test this hypothesis we compared the optimization results of both hybrid, which we will refer to as H-Tree and H-SVM respectively.

Table II presents the average results of 51 independent runs on each of the 28 benchmark functions in 30 dimensions, using a limit of 300,000 function evaluations; this experimental setting corresponds to the evaluation criteria defined for the IEEE CEC'13 benchmark in [19]. The table reports the mean error achieved by each algorithm, values in bold indicate the best performance for a given function. It also shows the relative difference in performance $100(a-b)/\max(a,b)$ achieved by H-Tree versus H-SVM. These values indicate by what amount (percent) H-Tree (b) outperforms H-SVM (a); positive values indicate that H-Tree performs better. A t-test between the two samples is also reported to allow a comparison on the basis of statistically significant differences at the 5% level.

Results on Table II show that the difference in classification performance is indeed reflected upon the optimization results of the hybrids. The H-Tree hybrid performs as good or better than the H-SVM hybrid in 26 out of the 28 functions. In the two functions where H-SVM performs slightly better the difference is not statistically significant. Another way of pointing out the clear superiority of H-Tree over H-SVM is that it achieves a statistically significant improvement in 17 out of the 28 functions, while in the other 11 functions both algorithms perform similarly.

Overall, H-Tree outperforms H-SVM by 20.40%, with this relative difference in performance raising up to 27.63% in the set of basic multimodal functions. The larger difference in performance on the set of multimodal functions is to be expected; it reinforces the importance of finding the correct transition point between exploration and exploitation in these type of functions. It is also to be expected that accurately determining this moment in unimodal functions (or functions with large attraction basins, as is the case in some of the composition functions) is less relevant because an efficient algorithm, such as CMA-ES, may still converge to the optimum even if the transition doesn't occur at the right time.

B. Comparing Transition Points

It is reasonable to expect that the ideal transition point will occur rather early for unimodal functions and at a more advanced moment when optimizing multimodal functions. Arguably there is no need to perform exploration in unimodal functions as all the points belong to the same attraction basin,

TABLE II
COMPARISON OF SVM AND DT CLASSIFIERS IN HYBRIDS

	H-SVM	**H-Tree**		
No.	Mean	Mean	%-diff	t-test
1	**0.00E+00**	**0.00E+00**	0.00%	–
2	**0.00E+00**	**0.00E+00**	0.00%	–
3	3.02E+01	**1.31E+01**	56.68%	0.00
4	**0.00E+00**	**0.00E+00**	0.00%	–
5	**0.00E+00**	**0.00E+00**	0.00%	–
6	8.80E-01	**0.00E+00**	100.00%	0.00
7	1.60E+01	**1.15E+01**	27.91%	0.00
8	2.14E+01	**2.09E+01**	2.41%	0.06
9	2.37E+01	**1.99E+01**	16.20%	0.00
10	1.48E-03	**7.40E-04**	50.0%	0.00
11	4.37E+01	**2.39E+01**	45.30%	0.00
12	3.65E+01	**2.32E+01**	36.48%	0.00
13	7.37E+01	**5.08E+01**	31.05%	0.00
14	3.27E+03	**2.80E+03**	14.51%	0.02
15	3.01E+03	**2.21E+03**	26.44%	0.00
16	5.96E-02	**4.81E-02**	19.37%	0.00
17	**6.17E+01**	6.60E+01	-6.42%	0.06
18	6.12E+01	**4.31E+01**	29.62%	0.00
19	2.32E+00	**2.26E+00**	2.74%	0.05
20	1.20E+01	**9.71E+00**	18.86%	0.00
21	3.21E+02	**1.67E+02**	48.11%	0.00
22	3.18E+03	**2.41E+03**	24.18%	0.00
23	2.98E+03	**2.56E+03**	14.17%	0.00
24	2.35E+02	**2.29E+02**	2.53%	0.05
25	2.82E+02	**2.71E+02**	3.78%	0.07
26	**2.14E+02**	2.25E+02	-4.81%	0.07
27	8.04E+02	**7.07E+02**	11.99%	0.00
28	**3.00E+02**	**3.00E+02**	0.00%	0.09
Overall			20.40%	
Unimodal			11.34%	
Multimodal			27.63%	
Composition			12.49%	

TABLE III
TRANSITION POINTS COMPARISON

No.	**H-SVM**	**H-Tree**	**Frequent**	**Optimum**
1	25.00%	5.00%	5.00%	5.00%
2	5.00%	5.00%	5.00%	5.00 %
3	5.00%	5.00%	5.00%	15.00%
4	5.00%	5.00%	5.00%	5.00 %
5	24.84%	5.00%	5.00%	5.00 %
6	25.00%	5.00%	5.00%	5.00%
7	25.00%	39.68%	50.00%	80.00%
8	25.00%	97.67%	90.00%	100.00%
9	25.00%	37.84%	30.00%	80.00%
10	25.00%	5.00%	10.00%	25.00%
11	25.00%	43.68%	55.00%	65.00%
12	25.00%	45.00%	55.00%	65.00%
13	25.00%	41.01%	50.00%	70.00%
14	25.00%	60.84%	65.00%	65.00%
15	25.00%	62.18%	65.00%	70.00%
16	25.00%	5.00%	10.00%	5.00%
17	25.00%	82.51%	100.00%	100.00%
18	25.00%	100.00%	100.00%	100.00%
19	25.00%	5.00%	15.00%	10.00%
20	25.00%	98.17%	90.00%	100.00%
21	24.51%	5.00%	5.00%	5.00%
22	25.00%	66.34%	70.00%	70.00%
23	25.00%	64.84%	70.00%	70.00%
24	25.00%	45.00%	45.00%	75.00%
25	25.00%	45.00%	50.00%	75.00%
26	25.00%	15.34%	65.00%	75.00%
27	25.00%	49.51%	50.00%	60.00%
28	25.00%	5.00%	10.00%	25.00%
Unimodal	16.98%	5.00%		
Multimodal	25.01%	48.57%		
Composition	24.95%	37.01%		

thus transitioning as early as possible to an exploitative (local search) method should yield the best results. Conversely, in multimodal functions, exploration must guarantee finding a good attraction basin before initiating the transition to local search. It is also reasonable to expect that the ideal transition point in multimodal functions will vary depending on topological properties; such as the number of attraction basins, their size and shape and correlations among variables.

Table III shows the transition points of H-SVM and H-Tree for each function. The transition point shows the average for each function expressed in terms of the percent of function evaluations at which the transition was started. The last two columns show the most frequent point at which a transition was recommended and the optimum transition point (the transition that yields the best result). If more than one transition point is the most frequent then the earliest one is reported.

The transition points predicted by the SVM classifier confirm that this model is suffering from a high bias and is incapable of learning a good hypothesis function. It accurately discriminates three of the unimodal functions for which it correctly suggests the earliest possible transition point. For all the other functions (most of which are multimodal) it predicts a transition point at 25% of function evaluations; but the model fails to predict distinctive transition points for each of the multimodal functions.

The Decision Tree classifier, on the other hand, does a much better job. First of all, it accurately identifies the unimodal functions and predicts the earliest transition point (5%) for them. For the multimodal functions, it predicts a full range of different transition points that ranges from an early transition in functions like F6 and F21, to no transition at all in F18. It is relevant to notice that for many functions the predictions tend to be closer to the earliest most frequent point than to the actual optimum transition. This is to be expected given the way the dataset was built; future work should avoid making every improving transition a positive example and focus on those closer to the optimum transition point.

It is interesting to notice the early transition on F6, the Rosenbrock's function, which is characterized by "having a very narrow valley from local optimum to global optimum" [19]. CMA-ES is known to be good at optimizing functions with such topological features (e.g. the elliptic function [21]), thus it makes sense that an early transition is preferable even though this is a multimodal function. Function

21 is a composition function that has Rosenbrock's as its primary function, thus it is reasonable to expect it has similar features to F6.

Figure 3 shows the distribution of ones and zeros in the obtained dataset. There are five distinctive behaviours characterized in the figure by representative functions. For instance, all the multimodal functions (F1-F6) plus F10, F21 and F28 have all their training examples with $y = 1$. This means that, at any transition point, switching to CMA-ES will yield better results. Functions like F16 and F19 have a decreasing distribution of 1s, meaning it is highly recommendable to transition at the beginning, but less recommendable as optimization advances. A third group represented by F9 shows the opposite distribution: early transitions are not recommended but later ones are.

Another group of 7 functions, represented in the figure by F13, has a distribution that peaks somewhere between 50% and 80%, and the optimum transition point is also found in this interval. These are multimodal functions that benefit from a long exploration phase, but yield the best result if the final iterations are devoted to local search. Finally, there are 4 functions represented by F8, for which the transition to CMA-ES is inconvenient at any point; these functions are highly multimodal, non-separable and asymmetrical.

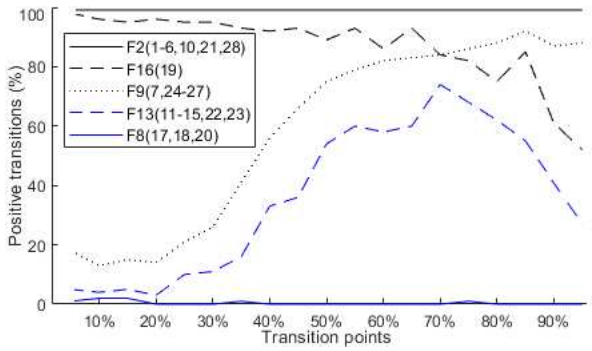

Fig. 3. Distribution of positive transitions in the dataset.

V. Computational Results on Optimization

In this section we present the optimization results of the H-Tree hybrid compared to other optimization algorithms. These results are not meant to compare the performance of H-Tree against some random metaheuristic algorithm chosen from literature, but intended to emphasize the effectiveness of the machine learning approach to hybridization. Results do include comparisons against MPS and CMA-ES, which on their own have proven to be effective metaheuristics.

Table IV shows the results of the Decision Tree hybrid compared against it composing algorithms: MPS and CMA-ES; and against a fixed point transition hybrid with the transition set at a 50% of the function evaluations. The table shows the mean fitness error over 51 independent runs; the best result among all four algorithms is bolded. The table also presents the relative difference in performance of H-Tree vs the other algorithms. A positive value indicates the amount by which H-Tree outperforms the corresponding algorithm, a negative value indicates H-Tree performs worse. A t-test is also reported. The final column shows the results that could be achieved by transitioning at the optimum transition point.

The Decision Tree hybrid performs as well or better than the other algorithms in 16 out of the 28 benchmark functions. It is outperformed by MPS in 7 functions, but only in 4 is the difference statistically significant. H-Tree outperforms MPS with a statistically significant difference in 10 functions, and over the entire benchmark it outperforms MPS by 22.99%. As expected, CMA-ES is the best algorithm in the set of unimodal functions, outperforming H-Tree on this set by 19.50%. However, H-Tree performs much better on the rest of benchmark, particularly in the set of multimodal functions where it outperforms CMA-ES by over 50%. The hybrid achieves 28.84% improvement in performance on the entire benchmark. These result clearly show evidence of the benefits of hybridization and explicitly separating the tasks of exploration and exploitation.

The comparison against the fixed transition hybrid aims to illustrate the benefits of the machine learning approach to hybridization. It is no surprise that both hybrids perform similarly, in 14 of the functions there is no statistically significant difference. However, the fact that in the other 14 functions (and most of the multimodal ones) the difference in performance is statistically significant proves that the use of the classifier is relevant. In the set of multimodal fuctions H-Tree outperforms the fixed point hybrid by 13.57%, and by 11.27% over the entire benchmark.

A noteworthy failure in the prediction of the transition point occurs in the Rotated Schaffer's No.7 function (F7 in the benchmark). In this function, Decision Tree predicts an early transition when the optimal transition actually happens at the 80% of function evaluations. The reason seems to be the difference between the optimum transition (80%) and the most frequent transition (50%); it leads to the largest under performance of H-Tree against the fixed hybrid.

Another inaccuracy in the classification occurs in functions F11 and F12, which are the original (separable) Rastrigin function and the Rotated Rastrigin function, respectively. In both these functions the Decision Tree classifier predicts the transition slightly before the 50%; 43.68% and 45.00% respectively (Table III). Even though this is a small difference compared to a 50% transition, this slight variation in the prediction moment clearly reflects on the final results. This most likely happens because MPS is very effective in optimizing the Rastrigin function, as it was specifically designed for highly multimodal yet global convex functions like Rastrigin [12]. These results also reinforce the importance of performing the transition at the right moment.

Although the H-Tree hybrid clearly improves the performance of its composing algorithms and the fixed hybrid, the results for the best transition point show that there is room for further improvement. To seize this opportunity it might be necessary to change the dataset to better reflect the points of optimum transition instead of earliest improving transition.

TABLE IV
H-TREE COMPARED AGAINST COMPOSING ALGORITHMS AND FIXED HYBRID

	H-Tree	**MPS**			**CMA-ES**			**Hybrid transition at 50%**			**Best transition**
No.	Mean	Mean	%-dif	t-test	Mean	%-dif	t-test	Mean	%-dif	t-test	Mean
1	**0.00E+00**	**0.00E+00**	0.00%	–	**0.00E+00**	0.00%	–	**0.00E+00**	0.00%	–	0.00E+00
2	**0.00E+00**	6.71E+05	100.00%	0.00	**0.00E+00**	0.00%	–	**0.00E+00**	0.00%	–	0.00E+00
3	1.31E+01	1.13E+07	100.00%	0.00	**2.84E+00**	-78.24%	0.00	2.32E+01	43.68%	0.00	2.33E-08
4	**0.00E+00**	4.24E+01	100.00%	0.00	**0.00E+00**	0.00%	–	**0.00E+00**	0.00%	–	0.00E+00
5	**0.00E+00**	5.53E-03	100.00%	0.00	**0.00E+00**	0.00%	–	**0.00E+00**	0.00%	–	0.00E+00
6	**0.00E+00**	2.52E+01	100.00%	0.00	4.34E+00	100.00%	0.00	2.64E+00	100.00%	0.00	0.00E+00
7	1.15E+01	6.61E+00	-42.55%	0.00	3.10E+01	62.92%	0.00	**6.60E+00**	-42.65%	0.00	2.69E+00
8	**2.09E+01**	2.10E+01	0.12%	0.13	2.15E+01	2.60%	0.06	2.15E+01	2.58%	0.10	2.09E+01
9	1.99E+01	1.96E+01	-1.56 %	0.07	**1.63E+01**	-17.91%	0.01	1.81E+01	-9.13%	0.04	1.61E+01
10	**7.40E-04**	3.41E-02	97.83 %	0.00	1.49E-02	95.02%	0.00	3.78E-03	80.43%	0.00	0.00E+00
11	2.39E+01	**2.08E+01**	-13.04%	0.01	5.18E+01	53.84%	0.00	2.26E+01	-5.41%	0.05	1.73E+01
12	2.32E+01	**1.91E+01**	-17.86%	0.00	5.14E+01	54.81%	0.00	1.92E+01	-17.29%	0.01	1.41E+01
13	5.08E+01	5.33E+01	4.67 %	0.07	1.01E+02	49.52%	0.00	**4.83E+01**	-4.98%	0.06	2.64E+01
14	**2.80E+03**	2.88E+03	2.95%	0.07	3.39E+03	17.48%	0.02	3.05E+03	8.15%	0.04	2.26E+03
15	2.21E+03	**2.08E+03**	-5.95%	0.06	3.23E+03	31.61%	0.00	2.54E+03	12.92%	0.02	1.48E+03
16	**4.81E-02**	1.47E-01	67.25%	0.00	8.84E-00	99.46%	0.00	6.48E-02	25.76%	0.00	2.39E-02
17	6.60E+01	**4.29E+01**	-35.02%	0.00	1.42E+02	53.40%	0.00	6.18E+01	-6.25%	0.07	4.21E+01
18	**4.31E+01**	4.40E+01	2.07%	0.11	2.82E+02	84.72%	0.00	6.30E+01	31.63%	0.00	4.36E+01
19	**2.26E+00**	3.40E+00	33.58%	0.00	3.31E+00	31.85%	0.00	2.52E+00	10.59%	0.04	1.67E+00
20	**9.70E+00**	9.72E+00	0.07%	0.11	1.50E+01	35.24%	0.00	1.17E+01	17.26%	0.00	9.67E+00
21	**1.67E+02**	3.13E+02	46.84%	0.00	3.13E+02	46.84%	0.00	3.15E+02	47.01%	0.00	1.77E+02
22	**2.41E+03**	2.44E+03	1.20%	0.08	3.98E+03	39.50%	0.00	2.90E+03	16.99%	0.01	1.87E+03
23	**2.56E+03**	2.63E+03	2.43%	0.07	4.11E+03	37.73%	0.00	2.59E+03	1.22%	0.07	1.84E+03
24	2.29E+02	2.36E+02	2.93%	0.07	**2.25E+02**	-1.54%	0.08	2.33E+02	1.70%	0.06	2.27E+02
25	2.71E+02	2.73E+02	0.42%	0.11	**2.67E+02**	-1.72%	0.07	2.69E+02	-0.96%	0.07	2.66E+02
26	2.25E+02	**2.05E+02**	-8.65%	0.06	2.85E+02	21.10%	0.00	2.31E+02	2.59%	0.06	2.17E+02
27	7.07E+02	7.53E+02	6.07%	0.04	7.11E+02	0.52%	0.12	**7.05E+02**	-0.10%	0.09	6.41E+02
28	**3.00E+02**	**3.00E+02**	0.00%	–	3.68E+02	18.44%	0.02	**3.00E+02**	0.00%	–	3.00E+02
Overall			**22.99%**			**28.84%**			**11.27%**		22.67%
Unimodal			80.00%			-19.50%			8.74%		20.00%
Multimodal			12.84%			50.12%			13.57%		31.63%
Composition			6.41%			19.14%			8.56%		7.53%

A. Machine Learning Hybrid vs. Fixed Point Transition

Table V presents the relative improvement of H-Tree vs hybrids with fixed transition at different points. Overall, the best fixed transition point is at 75%, which is reasonable since usually exploration is more difficult than exploitation and it benefits from more function evaluations. But the machine learning approach still improves this hybrid by 10%.

It is relevant to notice that H-tree outperforms all the fixed transition point hybrids on every subset, except for the unimodal set on a 5% transition. From Table III we know that Decision Tree predicts the earliest transition point (i.e. 5%) for every unimodal function (thus the difference between H-Tree and a fixed hybrid at 5% is small and not significant). However, the optimum transition for F3 (Rotated Bent Cigar Function) is at 15%, and a transition at 25% also yields better results than a transition at 5%. The Rotated Bent Cigar Function is the only unimodal function not optimally solved by CMA-ES and the hybrids. This shows that for the most challenging unimodal functions CMA-ES may benefit from some initial exploration.

From a machine learning perspective, an interesting insight is that the Classifier learns to predict the earliest transition point (5%) instead of the optimum transition point (25%). However, this is not surprising since the training examples are created to capture a positive example ($y = 1$) for any transition in which CMA-ES leads to a better result. Thus, in unimodal functions all the 19 training examples created from a run are positive, and there is no information that allows the classifier to learn that a better result can be achieved at a 25% transition. Future work should consider alternative ways for creating the training examples that could capture this information.

TABLE V
H-TREE VS FIXED TRANSITION HYBRIDS

	5%	**25%**	**50%**	**75%**	**95%**
Overall	13.9%	16.6%	11.2%	10.0%	29.4%
Unimodal	3.0%	-9.3%	8.7%	19.9%	80.0%
Multimodal	18.8%	27.2%	13.6%	10.9%	25.8%
Composition	11.6%	12.9%	8.6%	2.1%	4.8%

VI. Discussion

The results presented in this paper confirm that the optimum transition point in relay hybrids is tied to the characteristics of the objective function. Furthermore, even for the same function this point may vary depending on the specific run; training examples from different runs on the same function did not always suggest the same transition point. For instance, in the Ratrigin Function (F11), positive training examples representing an effective transition appear in some runs as early as 30% of the function evaluations. However, in other runs the final result of MPS was better than any hybridization point, i.e. no transition point was recommended.

This starkly contrasts with what happens in unimodal functions, where all the 19 transition points led to a better performance; thus all the training examples are positive examples. For some functions like F14 (Schwefel's) transition becomes effective on a specific interval, e.g. from 35% to 65% of function evaluations, but if transition is not performed at this interval then it should not be performed afterwards. For other functions like F18 (Rotated Lunacek Bi-Rastrigin) transition is not recommended at all.

This illustrates the complexity of the classification problem at hand, and raises the question of how well the trained classifier will be able to generalize when optimizing functions other than those present in the benchmark. Fortunately, the CEC'13 benchmark includes a large and diverse set of functions, thus the classifier is expected to be robust in this sense. To confirm this, an experiment was performed where the Decision Tree was trained with the examples from every function except one selected function. Then the machine learning hybrid was tested on that function and its results compared against those of the classifier trained with the full data set. This experiment was repeated for every function in the benchmark and no statistically significant difference was observed when optimizing any of the selected functions.

This experiment suggests that the model is capable of recognizing general topological characteristics that are reflected upon the 124 input features, more than just learning the topology of individual functions. However, the training examples are not only specific to the benchmark functions, but also to the experiment design, i.e. the number of dimensions and the stopping criteria. Thus, future work should analyze how these factors may influence the accuracy of the prediction and the optimization performance.

Future work should also consider applying more complex models, like deep neural networks, and collecting more features. Furthermore, future work could focus on predicting more than just the optimum transition point. Results presented in [16] show that the effectiveness of exploration varies depending on the topology of the function. Evidence was also presented that exploration can be controlled by adjusting the γ parameter on algorithms using Thresheld Convergence. Simple if-then-else rules were used for adjusting γ during the optimization, which lead to a significant improvement in performance. Instead of using fixed rules, the problem of adjusting γ could be modeled as a classification problem: whether to increase or reduce it at regular intervals. Combining the the transition point and γ predictions could lead to the design of better exploration-only exploitation-only hybrids.

References

[1] A. Bolufé-Röhler, S. Fiol-González, and S. Chen, "A minimum population search hybrid for large scale global optimization," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 1958–1965.

[2] A. Bolufé-Röhler and S. Chen, "A multi-population exploration-only exploitation-only hybrid on cec-2020 single objective bound constrained problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

[3] D. Tamayo-Vera, S. Chen, A. Bolufé-Röhler, J. Montgomery, and T. Hendtlass, "Improved exploration and exploitation in particle swarm optimization," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2018, pp. 421–433.

[4] A. Bolufé-Röhler and D. Tamayo-Vera, "Machine learning based metaheuristic hybrids for s-box optimization," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–14, 2020.

[5] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, vol. 74.

[6] C. Blum, A. Roli, and M. Sampels, *Hybrid metaheuristics: an emerging approach to optimization*. Springer, 2008, vol. 114.

[7] D. Molina, A. LaTorre, and F. Herrera, "Shade with iterative local search for large-scale global optimization," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[8] Y. Gonzalez-Fernandez and S. Chen, "Leaders and followers—a new metaheuristic to avoid the bias of accumulated information," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 776–783.

[9] F. D. E. Glorieux, B. Svensson and B. Lennartson, "Constructive cooperative coevolution for large-scale global optimisation."

[10] A. Bolufé-Röhler, A. Coto-Santiesteban, M. Rosa-Soto, and S. Chen, "Minimum population search, an application to molecular docking," *GECONTEC: Revista Internacional de Gestión del Conocimiento y la Tecnología*, vol. 2, no. 3, 2014.

[11] C. Blum and G. R. Raidl, *Hybrid Metaheuristics: Powerful Tools for Optimization*. Springer, 2016.

[12] X. G. W. Chu and S. Sorooshian, "A new evolutionary search strategy for global optimization of high-dimensional problems," in *Information Sciences, vol. 181*, 2011, p. pp. 4909–4927.

[13] A. Bolufé-Röhler and S. Chen, "Minimum population search – a scalable metaheuristic for multi-modal optimization," in *Revista Investigación Operacional, vol 36(1)*, 2015, p. pp. 85–95.

[14] A. Bolufé-Röhler, S. Chen, and D. Tamayo-Vera, "An analysis of minimum population search on large scale global optimization," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 1205–1212.

[15] A. Piad-Morffis, S. Estévez-Velarde, A. Bolufé-Röhler, J. Montgomery, and S. Chen, "Evolution strategies with thresheld convergence," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 2015, pp. 2097–2104.

[16] D. Tamayo-Vera, A. Bolufé-Röhler, and S. Chen, "Estimation multivariate normal algorithm with thresheld convergence," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*, 2016, pp. 3425–3432.

[17] N. Hansen, "The cma evolution strategy: A tutorial," 2016.

[18] "Matlab learning app classifiers," June 2020. [Online]. Available: https://www.mathworks.com/help/stats/choose-a-classifier.html

[19] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," *Computational Intelligence Laboratory, Zhengzhou University, Technical Report*, vol. 201212, no. 34, pp. 281–295, 2013.

[20] A. Bolufe-Rohler, "Machine learning transition point for mps-cmaes hybrid," https://github.com/Bolufe-Rohler/ML-transition-point-MPS-CMAES-hybrid, 2021.

[21] A. Auger and N. Hansen, "A restart cma evolution strategy with increasing population size," in *2005 IEEE congress on evolutionary computation*, vol. 2. IEEE, 2005, pp. 1769–1776.