Learning Hybridization Strategies in Evolutionary Algorithms

Antonio LaTorre, José-María Peña and Santiago Muelas
Computer Architecture Department, Facultad de Informática
Universidad Politécnica de Madrid, Spain
{atorre, jmpena, smuelas}@fi.upm.es

Alex A. Freitas
Computing Laboratory, University of Kent
United Kingdom
A.A.Freitas@kent.ac.uk

December 23, 2009

Abstract

Evolutionary Algorithms are powerful optimization techniques which have been applied to many different problems, from complex mathematical functions to real-world applications. Some studies report performance improvements through the combination of different evolutionary approaches within the same hybrid algorithm. However, the mechanisms used to control this combination of evolutionary approaches are not as satisfactory as would be desirable. In most cases, there is no feedback from the algorithm nor any regulatory component that modifies the participation of each evolutionary approach in the overall search process. In some cases, the algorithm makes use of some information for an on-line adaptation of the participation of each algorithm. In this paper, the use of Reinforcement Learning (RL) is proposed as a mechanism to control how the different evolutionary approaches contribute to the overall search process. In particular, three learning policies based on one of the state-of-the-art RL algorithms, Q-Learning, have been considered and used to control the participation of each algorithm by learning the best-response mixed strategy. To test this approach, a benchmark made up of six large-scale (500 dimensions) continuous optimization functions has been considered. The experimentation carried out has proved that RL control mechanisms successfully learn optimal patterns for the combination of evolutionary algorithms in most of the proposed functions, being able to improve the performance of both individual and non RL hybrid algorithms.

Multiple Offspring Sampling, Reinforcement Learning, Q-Learning, Hybrid Evolutionary Algorithms.

1 Introduction

Reinforcement learning (RL) is a powerful AI technique to construct best-response strategies for an agent interacting within an environment. As a machine learning topic, RL deals with the identification of the sequence of actions an agent has to carry out in order to maximize a reward function. RL has successfully been applied to different domains from classic games to control systems.

Furthermore, this paper considers the MOS (Multiple Offspring Sampling) [20] hybrid evolutionary framework. This framework is able to combine different population-based optimization heuristics, such as genetic algorithms (GA), estimation of distribution algorithms (EDA), evolutionary strategies (ES) or differential evolution (DE). MOS uses the mechanisms offered by these heuristics to create the offspring for the next generation. In each generation, MOS selects the sampling quota, by means of a Participation Function (PF), applied to each of the participant techniques.

MOS has been used, with remarkable results, to solve different complex optimization benchmarks [18, 21]. These results have been obtained using a short-term evaluation of the quality of the techniques, mainly from the individuals generated by each technique during the last generation.

This paper proposes a different approach to deal with MOS offspring sampling quota assignment. This approach considers the adjustment of this quota as a decision problem that can be solved using a long-term strategy. The rationale behind this idea is inspired by the objective not to maximize the quality (fitness) of the best individual in a given generation but the quality (fitness) of the best individual in the last generation. As a result of this approach, the reward obtained by this long-term situation should be propagated back to the decision taken at a given point during the evolutionary process.

Moreover, it should be taken into account that the problem of parameterization of a heuristic algorithm is a complex scenario for RL because actions, decided by the strategies, perform stochastically. This stochastic behavior comes from both randomness in single individual generation and survival of the fittest individuals depending on the selection schema (and the fitness of the rest of the individuals in the population). Consequently, this study requires the application of RL techniques that are able to deal with stochastic reward functions, like those used for stochastic games, such as WoLF [4], PHC [4] or TERSQ [24].

The goal of this paper is two-fold: (i) improving the results obtained by MOS deciding the sequence of actions that performs the best when looking for the long-term objective, and (ii) validating whether RL techniques are able to learn the best-response mixed strategy for a complex scenario where the reward function is stochastically determined.

Finally, this paper is structured as follows: Section 2 briefly reviews previous work on Hybrid Evolutionary Algorithms and Reinforcement Learning techniques. In Section 4 the proposed hybrid evolutionary algorithm with RL-based participation adjustment is presented. Section 5 describes the experimental scenario: benchmark functions considered, configuration of the algorithms

and procedure for the experimentation. In Section 6 the experimental results are detailed and discussed, whereas Section 7 presents the main conclusions of this study.

2 Related Work

This section will review some of the most relevant work on Hybrid Evolutionary Algorithms (Section 2.1) and Reinforcement Learning (Section 2.2)

2.1 Hybrid Evolutionary Algorithms

In spite of the wide range of application fields and good results that Evolutionary Algorithms have obtained in complex optimization problems, their results are not always as good as one would expect. Normally, most researchers only test a few different algorithms when trying to solve a particular optimization problem. Even if the most suitable algorithm for that problem has been selected, it is hard to find its best configuration (parameters and set of operators). Additionally, different algorithms work better on some problems than on others. This is in accordance with the No Free Lunch Theorem [36], which states that for any algorithm with an outstanding performance on a given problem there is always another problem where a different algorithm performs better. Although NFL is based on certain extreme theoretical considerations [7, 23], real-world problems also show differences in the comparative performance of several algorithms. Some studies into Hybrid Evolutionary Algorithms [12, 21] show that with the combination of different search strategies it is possible to obtain better results compared to their individual performance.

According to Sinha and Goldberg [29], there are three main reasons for hybridization in Evolutionary Algorithms:

1. An improvement in the performance of the Evolutionary Algorithm (for example, the speed of convergence).

2. An improvement in the quality of the solutions obtained by the Evolutionary Algorithm.

3. To incorporate the Evolutionary Algorithm as a part of a larger system.

A comprehensive review of Hybrid Evolutionary Algorithms can be found in [10]. In this section we will focus on the first two reasons for hybridization in Evolutionary Algorithms: improvement in performance of the EAs and quality of the obtained solutions.

Much work has been done on the hybridization of different recombination operators in Genetic Algorithms. Some good examples in this line are the fuzzy logic controller proposed in [11] to control the participation of different crossover operators or the experiments by Hong [13] where the participation of several crossover operators is adjusted based on the progress introduced into the

population by using each of them. Other algorithms propose island GAs where each island evolves a population by means of recombination operators with different characteristics, trying to achieve a good trade-off between exploration and exploitation mechanisms by controlling migratory processes [30]. Finally, some studies propose the use of two different populations: one for the problem itself and another one for the set of operators that will be used [17].

However, hybridization is not restricted to taking place within the same evolutionary paradigm. Some studies propose Hybrid Evolutionary Algorithms where two or more different algorithms collaborate through the search process. This is the case for the GA-EDA algorithm [25] where a GA and an EDA are combined and applied to the resolution of both discrete and continuous problems. Shi et al. [27] proposed a hybrid EA-PSO algorithm where both subsystems are carried out in parallel, and a few individuals are exchanged every generation. Tseng and Liang [33] proposed a hybrid approach that combines Ant Colony Optimization (ACO), a Genetic Algorithm and a Local Search for the Quadratic Assignment Problem (QAP). In their experiments, alternative phases of ACO and GAs are executed. The pheromone values necessary for the ACO are also updated while the algorithm is in the GA phase to assure the correct behavior of the ACO. The Local Search improves solutions produced by both algorithms.

Finally, a third way of hybridization in Evolutionary Algorithms deals with the encoding of solutions. Studies carried out by [15] on the difficulty of different optimization problems have measured one of the aspects of problem complexity by correlating the difference between fitness function values and the Euclidean distance in the solution space. Experimental results show that a translation of the fitness landscape can make a problem easier or harder to solve. However, few works have taken this issue into consideration. In [26] the authors propose a Genetic Algorithm to solve different optimization functions where the individuals can be encoded with a cartesian or a pseudo-polar coding and be combined by using either of them. In [21], five Genetic Algorithms are combined to solve several instances of the TSP (Travelling Salesman Problem). Four of these GAs used an integer path representation, while the other one used a real ranking encoding.

2.2 Reinforcement Learning

In [16], the authors define Reinforcement Learning (RL) as *"The problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment"*. In the standard Reinforcement Learning model, the agent is connected to its environment in a bidirectional way. On the one hand, the agent receives an input, via perception, and some information about the state of the environment. On the other hand, the agent interacts with its environment by carrying out an action than can potentially change the state of the environment. A reinforcement signal is associated with each action and the behavior of the agent should chose actions maximizing the long-term sum of the reinforcement signal. This behavior can be learnt if a systematic trial-and-error

search is guided by the appropriate algorithm.

Formally, the Reinforcement Learning model is made up of:

- A discrete set of states, S.
- A discrete set of actions, A.
- A set of reinforcement signals (typically $0, 1$ or real numbers).

An important issue in Reinforcement Learning is how the agent will take the future into account. There are basically three models that try to optimize the reward in different moments. The *finite-horizon* model tries to optimize the reward in the following h steps. The *infinite-horizon* model considers the attenuated long-term rewards as if they were an interest rate. Finally, the *average-reward* model takes into account the long-term average reward.

Problems with delayed reinforcement, such as Reinforcement Learning, are well modeled as *Markov Decision Processes (MDPs)*. An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, T, R)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the action set of the agent and T is the transition function. As MDPs have non-deterministic state transitions, T is defined as $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. R is the reward function of the agent, and it is defined as a probability distribution: $\mathcal{S} \times \mathcal{A} \rightarrow PD(\mathbb{R})$.

There are two alternatives for obtaining an optimal policy for an MDP. First, a controller can be learnt without learning a model (Model-free approach). Second, a model can be learnt and, then, a controller can be derived from it (Model-based approach). In this contribution more attention will be paid to the Model-free approach and, especially, to a family of algorithms called *Q-learning*.

In Q-learning a matrix of Q-values is maintained. These values are the expected discounted reinforcement of taking action a in state s. This matrix is initialized to zero and its values are updated by means of the typical Q-learning rule:

$$Q(s, a) = Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

where $\{s, a, r, s'\}$ is an *experience tuple* summarizing a single transition in the environment, α is the learning rate and γ is the discount factor. In this tuple, s represents the state of the agent before the transition, a the action carried out, r the instantaneous reward it receives and s' the resulting state. This algorithm was proposed by Watkins in [35] and proved to converge to the optimal policy $Q*$ with probability 1 if each action is executed in each state an infinite number of times in infinite runs and if α is decayed appropriately [14].

Some variants of these algorithms have been successfully applied to MDPs. PHC and WoLF [5] are extensions of the Q-learning algorithm particularly designed to deal with stochastic scenarios. Both approaches maintain a learning rate in the form of a selection probability for each action-state pair. The main difference is that, in PHC, the learning rate is constant while WoLF changes this value whether it is winning or losing, following the idea of learning fast (higher rates) while losing and learning at a slower rate when winning.

In [3], the authors extend the WoLF algorithm to incorporate the concept of Infinitesimal Gradient Ascent (IGA) presented by [28] to define the "winning" situations required to update the learning rate in WoLF. GIGA-WoLF [2] is an extension of the latter considering the concept of Generalised IGA [37]. BL-WoLF [6] is an enhanced version of WoLF that provides a bounded-loss where the cost of learning is measured by the losses suffered by the learning agent (rather than the number of rounds). Another variant is Hyper-Q [32], in which values of mixed strategies rather than base actions are learnt, and in which other agents' strategies are estimated from observed actions via Bayesian inference. WPL (Weighted Policy Learner) [1] is a new RL algorithm which does not assume any knowledge of the underlying game structure.

A new Q-Learning variant called TERSQ was presented in [24]. The main idea underlying this algorithm is the use of a global stochastic quota, σ, in order to select the action to be carried out. The action with the best Q-value will be selected with a probability σ, whereas the remaining actions are stochastically selected with a probability of $1 - \sigma$ according to their Q-value ranking.

It is important to mention that few works have used RL as a regulatory mechanism for metaheuristics. In [22], the authors propose a hyper-heuristic where the basic heuristics are selected by a procedure inspired by Reinforcement Learning. In [8] the authors used RL techniques for the online control of some of the parameters of a Steady State Genetic Algorithm, which slightly improves the performance of the standard algorithm. However, none of these papers have explored the ability of RL to control the behavior of hybrid Evolutionary Algorithms, which is the goal of this work.

3 A Short Review of the MOS Algorithm

In our previous work [18], MOS is introduced as a hybrid adaptive algorithm capable of simultaneously handling several evolutionary approaches and of dynamically adjusting the participation of each of them in the overall search process. The Algorithm 1 presents a pseudocode of MOS describing the general functioning of this hybrid approach.

In MOS, the main algorithm handles the mechanisms to produce new individuals of the different evolutionary approaches (recombination operators in GAs, probabilistic models in EDAs, etc.). Each of these mechanisms capable of producing a new offspring is called a *technique*. Specifically, a technique can be defined as (a) a particular evolutionary model, (b) with an appropriate encoding, (c) using specific operators (if needed), and (d) configured with its necessary parameters.

Each of these techniques is able to produce a subset of the offspring from the current population, that is shared by all the techniques.

In this context, the tuple $(n, \mathcal{T}, \mathcal{P}, \mathcal{O})$ is called an MOS system, where n is the number of techniques in the technique set $\mathcal{T} = \{T_i\}$. $\mathcal{P} = \{P_i\}$ is the m-size set of common population per generation and $\mathcal{O} = \{O_i^{(j)}\}$ is the $n \times m$ set of offspring population per technique and generation.

Algorithm 1: Multiple Offspring Sampling Algorithm

1 **begin**
2 Uniformly distribute participation among the n used techniques $\rightarrow \forall j\ \Pi_0^{(j)} = \frac{|O_0|}{n}$
3 Create initial global population of candidate solutions P_0. Each technique produces a subset of individuals according to its participation ($\Pi_0^{(j)}$)
4 Evaluate initial population P_0
5 **while** *termination criterion not reached* **do**
6 **for** *every available technique* T_j **do**
7 **while** *ratio* $\Pi_i^{(j)}$ *not exceeded* **do**
8 Create new individuals from current population P_i
9 Evaluate new individuals
10 Add new individuals to an auxiliary population $O_i^{(j)}$
11 **end**
12 Update Quality of $T_j \rightarrow Q_i^{(j)} = Q(O_i^{(j)})$
13 **end**
14 Combine populations $O_i^{(j)}\ \forall j$ and P_i according to a pre-established criterion to generate P_{i+1}
15 Update participation ratios from Quality values computed in step 12 $\rightarrow \forall j\ \Pi_{i+1}^{(j)} = PF(Q_i^{(j)})$
16 **end**
17 **end**

The number of individuals that each technique can generate each generation ($\Pi_i^{(j)}$) is called its participation ratio. This ratio is uniformly distributed at the beginning of the search process, and it is periodically updated according to a given policy. In the canonical version of MOS, this adjustment is carried out by what is known as a *Participation Function*. These functions can carry out simple static assignments or, more interestingly, dynamic adjustments according to a *Quality Measure* that evaluates how good the offspring of each technique is from the point of view of that measure ($Q_i^{(j)}$).

At this point, different measures can be proposed, depending on the concept of quality that we consider. One option would be to consider the Fitness Average of the subset of the best individuals of the offspring generated by each technique as our measure of quality. Other alternatives could be used (Negative Slope Coefficient [34], Age, Diversity, etc.). If the Fitness Average measure is selected, the quality value computed from the offspring population produced by a technique j in generation i ($O_i^{(j)}$) is obtained as defined in Equation 1.

$$Q(O_i^{(j)}) = \sum_{o \in O_i^{(j)}} \frac{fit(o)}{|O_i^{(j)}|} \tag{1}$$

Every generation, the quality of each of the available techniques is recomputed, as depicted in step 12 of the MOS algorithm. As from now, the notation in Equation 2 will be used:

$$Q_i^{(j)} = Q(O_i^{(j)}) \quad and \quad Q_i^{(best)} = Q(O_i^{(best)}) \tag{2}$$
$$best = \underset{j}{argmax}(Q_i^{(j)})$$

Additionally, several Participation Functions (PFs) can be proposed. For this work, a dynamic Participation Function has been considered (Equation 3). This PF computes, in each generation, a trade-off factor for each technique, $\Delta_i^{(j)}$, that represents the decrease of participation for the $j-th$ technique in the $i-th$ generation, for every technique except the best performing one. This technique will increase its participation by the sum of all those $\Delta_i^{(j)}$.

$$\boldsymbol{PF}_{dyn}(Q_i^{(j)}) = \begin{cases} \Pi_i^{(j)} + \eta & \text{if } j = best, \\ \Pi_i^{(j)} - \Delta_i^{(j)} & \text{otherwise} \end{cases} \tag{3}$$
$$\eta = \sum_{k \neq best} \Delta_i^{(k)}$$

In our previous work [19], two different strategies for computing the aforementioned $\Delta_i^{(j)}$ factors have been proposed. In the first one, this factor is computed from the relative difference between the quality of the *best* and the $j-th$ offspring populations (Equation 4), where n represents the number of available techniques. This strategy keeps the overall population size fixed throughout the process. A minimum participation ratio can be established, but no maximum participation ratio is considered (apart from the implicitly maximum ratio $1-((n-1)\cdot ratio_{min})$ when using n techniques and the minimum participation ratio has been fixed to $ratio_{min}$).

$$\Delta_i^{(j)} = \xi \cdot \frac{Q_i^{(best)} - Q_i^{(j)}}{Q_i^{(best)}} \cdot \Pi_i^{(j)} \quad \forall j \in [1,n] \ / \ j \neq best \tag{4}$$

In previous equation, ξ is a reduction factor for the ratio that is transferred from one technique to the other (usually 0.05).

The second strategy (defined in Equation 5) also considers the quality value to carry out the participation adjustment. However, in this case the overall population size can change from one generation to the next. This means that an initial population size $|O_0|$ must be defined, and that the current population size can be decreased or increased through the evolutionary process, but it can never exceed the initial value. Additionally, a minimum participation ratio per technique is also allowed, as in the previous case, but a maximum participation ratio should be also defined, usually $\Pi_0^{(j)} = \frac{|O_0|}{n}$, n being the number of available techniques.

$$\Delta_i^{(j)} = \left(1 - \frac{Q_i^{(j)}}{Q_i^{(best)}}\right) \cdot \Pi_i^{(j)} \quad \forall j \in [1,n] \ / \ j \neq best \tag{5}$$

Algorithm 2: Multiple Offspring Sampling with RL Algorithm

```
1  begin
2      Initialize Matrix of Q-values (Q(s, a) = 0)
3      Create an initial overall population of candidate solutions P_0.
4      Evaluate initial population P_0
5      while termination criterion not reached do
6          if maximum number of generations per state reached then
7              Move from state s to s'
8          end
9          while |O_i| < |P_i| do
10             Apply Learning Policy to create a new individual and add it to the
               offspring population O_i.
11         end
12         Combine populations O_i and P_i according to a pre-established criterion to
           generate P_{i+1}
13     end
14 end
```

For the experimentation reported in this paper, only the first strategy has been considered as it obtained a better overall performance.

4 RL to Control MOS Strategies

In the previous sections we have reviewed the Multiple Offspring Sampling algorithm and the possibility of using RL to control its behavior. In this section, MOS will be extended to make use of the specific RL mechanisms reviewed in Section 2.2.

Each action in the MOSRL algorithm (MOS with RL extensions) is the creation of new offspring by means of one of the available reproductive techniques. The set of available states has been established by discretizing the participation of each reproductive techniques into eleven possible values ($\{0.0, 0.1, \ldots, 1.0\}$) with the only constraint being that the sum of the participation of each technique must be equal to 1. In addition, to favor the exploration, a maximum number of generations are allowed for each state. After every N generations, a state transition is automatically carried out to a new state where the discrete participations stay unchanged and the generation information is updated.

$$State = (\Phi \times G) \tag{6}$$

$$P = (\varphi_j^{(1)}, \ldots, \varphi_j^{(n)}) \in \{0.0, 0.1, \ldots, 0.9, 1.0\}^n \ / \ \sum_{i=1}^{n} \varphi_j^{(i)} = 1.0 \tag{7}$$

$$G \in \{[k \cdot N, (k+1) \cdot N) \ : \ k = 0 \ldots M\} \tag{8}$$

For example, if a hybrid algorithm with four techniques is being run and the current state is represented by the tuple $\{parts = \{0.1, 0.3, 0.4, 0.2\}, gen \in [100, 200)\}$ and the generation 200 arrives, then a new state transition is carried

out to a new state represented by the tuple $\{parts = \{0.1, 0.3, 0.4, 0.2\}, gen \in [200, 300)\}$.

Let $(n, \mathcal{T}, \mathcal{P}, \mathcal{O})$ be an MOS system, as seen in the previous section, and let $(\mathcal{S}, \mathcal{A}, T, R)$ be the tuple that describes an MDP. We say that this MDP is an MOS control strategy where:

$$\mathcal{S} = \{s_i : \exists j \in [0, m] s_i = sp(j, P_j)\} \tag{9}$$

$$\mathcal{A} = \{a_i : i \in [1, n]\} \tag{10}$$

$$T = \{(s_i, a_j, a_k, \pi_{i,j,k})\} \tag{11}$$

$$R = \{(s_i, a_j, r_{i,j})\} \tag{12}$$

Where sp is a state projection function that maps all possible population configurations into a set of MDP states (Equations 13, 14 and 15), $ndval$ being a function that approximates a real participation value to its nearest discrete value in $\{0.0, 0.1, \ldots, 1.0\}$.

$$sp(j, P_j) = (\Phi, G) \tag{13}$$

$$\Phi = \left(\varphi_j^{(1)}, \ldots, \varphi_j^{(n)}\right) \ / \ \varphi_j^{(i)} = ndval\left(\frac{|P_j^{(n)}|}{\sum_{i=1}^{n} |P_j^{(i)}|}\right) \tag{14}$$

$$G = [k \cdot N, (k+1) \cdot N) \ / \ k = \left\lfloor \frac{j}{N} \right\rfloor] \tag{15}$$

Each a represents the action "*create a new individual using the offspring mechanisms of technique i*". State transitions $(s_i, a_j, a_k, \pi_{i,j,k})$ express that there is a probability $\pi_{i,j,k}$ of changing from state s_i to state s_k when creating a new individual with technique j, action a_j. This probability depends on (i) the quality of the new individual, (ii) the selection pressure also based on the quality of other individuals, and (iii) the state projection function (multiple populations are mapped to the same states).

The elements in the reward function R, $(s_i, a_j, r_{i,j})$, mean "*the immediate reward obtained from creating a new individual with technique j in state s_i*". This reward value $r_{i,j}$ can be expressed as the improvement in the quality of the existing population derived from the contribution of the newly created individual.

For this work, three learning policies have been considered (PHC, WoLF and TERSQ) which are based on the state-of-the-art Q-learning algorithm. Algorithm 2 provides a detailed description of the common parts of the proposed algorithm, while a more thorough description of the three policies used will be offered in Sections 4.1, 4.2 and 4.3.

4.1 PHC Learning Policy

Policy Hill Climbing (PHC) was proposed in [4] as an extension to the Q-learning algorithm. In addition to Q-values, the algorithm also maintains the current

Algorithm 3: Policy Hill Climbing Algorithm

begin
  Let α and δ be learning rates, Initialize

$$Q(s,a) \leftarrow 0, \quad \pi(s,a) \leftarrow \frac{1}{|A|}$$

  while *current generation not finished* **do**
    From state s select action a with probability $\pi(s,a)$
    Q-values are updated observing reward r and next state s',

$$Q(s,a) = Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

    Update $\pi(s,a)$ and constrain it to a legal probability distribution

$$\pi(s,a) \leftarrow \pi(s,a) + \begin{cases} \delta & if a = \underset{a'}{argmax}(Q(s,a')) \\ \frac{-\delta}{|A|-1} & \text{otherwise} \end{cases}$$

  end
end

mixed policy ($\pi(s,a)$). This policy controls the probability of selecting a given action during the learning phase. It is updated by increasing the probability of selecting the best performing action according to a learning rate $\delta \in (0,1]$.

A detailed description of this policy is provided by Algorithm 3.

4.2 WoLF Learning Policy

WoLF policy was proposed in [4] as an extension of the PHC policy reviewed in the previous section. The basic idea is to modify the learning rate used dynamically to encourage convergence without sacrificing rationality. Intuitively, the algorithm tries to learn quickly when it is losing and more slowly when it is winning. To determine whether the algorithm is winning or losing, the current policy's payoff is compared with that of the average policy over time. For this purpose, the algorithm requires two learning parameters: δ_l, that will be used when the algorithm is losing, and δ_w, that will be used when the algorithm is winning, with $\delta_l > \delta_w$.

Algorithm 4 provides a detailed description of this policy.

4.3 TERSQ Learning Policy

In [24] the TERSQ algorithm was introduced. The main idea of this algorithm is to use an overall stochastic quota, σ, in order to select the action to be carried out. A binomial decision process is carried out in such a way that actions with best Q-values are selected with a probability of σ while the remaining actions are stochastically selected with a probability of $1-\sigma$ according to their Q-value ranking.

Algorithm 4: WoLF Algorithm

begin

Let α, $\delta_l > \delta_w$ be learning rates, Initialize

$$Q(s,a) \leftarrow 0, \quad \pi(s,a) \leftarrow \frac{1}{|A|}, \quad C(s) \leftarrow 0$$

while *current generation not finished* **do**

From state s select action a with probability $\pi(s,a)$

Q-values are updated observing reward r and next state s',

$$Q(s,a) = Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

Update estimate of average policy, $\bar{\pi}$,

$$C(s) \leftarrow C(s) + 1$$

$$\forall\ a' \in A \quad \bar{\pi}(s,a') \leftarrow \bar{\pi}(s,a') + \frac{1}{C(s)}(\pi(s,a') - \bar{\pi}(s,a'))$$

Update $\pi(s,a)$ and constrain it to a legal probability distribution

$$\pi(s,a) \leftarrow \pi(s,a) + \begin{cases} \delta & if a = \underset{a'}{argmax}(Q(s,a')) \\ \frac{-\delta}{|A|-1} & \text{otherwise} \end{cases}$$

where,

$$\delta = \begin{cases} \delta_w & if \sum_a \pi(s,a)Q(s,a) > \sum_a \bar{\pi}(s,a)Q(s,a) \\ \delta_l & \text{otherwise} \end{cases}$$

end

end

The σ value is selected for each round based on three different criteria. From these criteria, three phases can be established: (1) the *Tentative Phase* in which the algorithm tries all the possible σ values (from a finite set of values, named Γ) to get an initial estimation of the performance of every possible σ value; (2) the σ *Adjustment Phase* where σ values are proportionally chosen according to their average performance $\tau(\sigma)$ (which is updated at the end of each round), and; (3) the *Optimal* σ *Phase* where the σ value with highest average performance is selected for the rest of the learning process. The usual Q-learning technique is applied throughout all the process.

A detailed description of this policy can be found in Algorithm 5.

5 Experimental Scenario

For the experiments carried out in this work, the benchmark proposed for the CEC'08 Special Session and Competition on Large Scale Global Optimization [31] has been considered. This benchmark is made up of six scalable continuous functions with some of the characteristics that make these type of functions hard to be solved: multi-modality, non-separability, shifted global optimum

Algorithm 5: TERSQ Learning Policy

begin
  Let
    $\mathcal{A}$ be the set of possible actions for the state s, and $a \in \mathcal{A}$ one action for this state,
    α, γ the learning parameters,
    $\sigma \in \Gamma = \{0.0, 0.1, \ldots, 1.0\}$ the overall quota used to select A_{max},
    $\tau(\sigma)$ the average performance of σ
    σ be selected from Γ following the specific criteria of the current phase.
  Initialize $Q(s, a) \leftarrow 0$
  while *current generation not finished* **do**
    for *each action a on each state s* **do**
      Compute $\pi(s, a)$, a basic probability obtained by a ranking process where actions are sorted according to their Q-values in an increasing order:

$$\{A'_i\} = sort(\{a\}) \tag{16}$$

$$\forall_{i=1}^{n} \pi(s, \{A'_i\}) = i \times \pi_0 \quad / \quad \sum \pi(s, \{A'_i\}) = 1 \tag{17}$$

    end
    These probabilities are adjusted by the σ quota as follows,

$$\pi(s, a) = \pi(s, a) \times (1 - \sigma), \quad a \neq A_{max} \tag{18}$$

    and for the A_{max} (action with the best actual Q-value)

$$\pi(s, A_{max}) = (\pi(s, A_{max}) \times (1 - \sigma)) + \sigma \tag{19}$$

    Select action a with probability $\pi(s, a)$.
    Q-values are updated observing reward r and next state s',

$$Q(s, a) = Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

  end
  Update the $\tau(\sigma)$ according to the evaluation of the round.
end

etc. Specifically, the six considered functions are: Ackley, Griewank, Rastrigin, Rosenbrock, Schwefel and Sphere. These types of benchmarks, and this one in particular, are being used more and more in Conferences and Special Issues on Continuous Optimization as they ease the comparison of different search algorithms on a complete and predefined set of functions.

The first part of this section details the configuration used for the different tested approaches (single GAs, standard MOS algorithm and MOS algorithm combined with RL techniques), while the second part of this section presents the experimental procedure carried out.

5.1 Algorithms

Table 1 presents the configuration used by each of the evolutionary approaches considered in this experimentation (single GAs, Standard MOS and MOS with RL techniques). The number of executions has been set so that the RL algorithms have enough information to learn optimal hybridization strategies (2.5M

evaluations per execution). The convergence criterion has been fixed to a maximum number of evaluations as proposed in [31]. No minimum participation ratio has been imposed to the MOS algorithm because, as we will see in Section 6, two of the proposed techniques have a performance much worse than the other two, what makes undesirable to waste computational efforts on maintaining a marginal participation of these algorithms. The complexity of the considered functions has been established to 500 dimensions, a considerably large size which makes these functions quite hard to solve.

Table 1: Algorithm Configuration

	GAs	Standard MOS		MOSRL
Executions		111		
Convergence Criterion		2500000 FEs[1]		
Problem Size		500 dimensions		
Population Size		50		
Elitism		Full Elitism		
Participation Function	-	Dynamic[2]	Constant	-
Minimum Participation	-	0%	-	-

[1] Fitness Evaluations, as specified in [31]
[2] Dynamic PF, with constant population size

Table 2 presents the set of techniques used by the hybrid configurations. This set of techniques has been constructed by combining two crossover and two mutation operators classic in the literature for continuous optimization. Many researchers have considered the combination of the BLX-α Crossover and the Gaussian Mutator but, as we will see in Section 6, the combination of the BLX-α Crossover with the Uniform Mutator reports better results.

Table 2: Set of Techniques for the Hybrid Evolutionary Algorithm

	BCUM	UCUM	BCGM	UCGM
Crossover	BLX-α[1]	Uniform	BLX-α[1]	Uniform
Mutator	Uniform		Gaussian	
Crossover Rate	90%			
Mutation Rate	1%			
Selection	Roulette-Wheel			

[1] BLX-α with $\alpha = 0.5$

Table 3 shows the values for the parameters needed by the different RL policies. The learning rate (α) and the discount factor (γ) common to the three Q-Learning based policies have been selected following the guidelines provided in [9]. For the learning rates specific to PHC and WoLF (δ, δ_l and δ_w), the

recommended values from [4] have been used. The σ values needed by the TERSQ policy are adjusted by the policy itself, as it was explained in algorithm 5, and do not need to be fixed.

Table 3: Parameters of the RL policies

	PHC	WoLF	TERSQ
α		0.7	
γ		0.6	
δ	0.01	-	
δ_l	-	0.02	-
δ_w	-	0.005	-

5.2 Procedure

For each of the proposed problems, the following experimental procedure is carried out:

- Each evolutionary technique is executed individually.
- The four proposed evolutionary techniques are combined within MOS with both constant and dynamic participation functions and the configuration presented in Table 1.
- The four proposed evolutionary techniques are combined within the RL version of MOS with the configuration presented in Table 1.
- For the TERSQ algorithm, 11 rounds are carried out in the Tentative Phase (one for each of the possible σ values), 50 rounds in the σ Adjustment Phase and 50 rounds in the σ Optimal phase. These values have been identified as optimal in terms of performance and execution time in a previous experimentation. For the other two algorithms, PHC and WoLF, the same total number of 111 rounds are executed.
- For the non-RL algorithms (single GAs and standard MOS configurations) 111 independent runs have been carried out (the same number of executions as the RL algorithms).
- The average error in comparison with the global optimum is reported for each problem and configuration.
- A ranking analysis based on the fitness of each algorithm is carried out.
- The results obtained by each algorithm are pair-wise compared using a Wilcoxon non-parametric t-test against the others.

- A new ranking analysis is carried out, this time based on the results of the Wilcoxon t-test. The goal of this global analysis is to rank the performance of the proposed algorithms. In this analysis, if one algorithm is significantly better than other (p-value < 0.05), the winning algorithm is granted +1 "wins" and the losing algorithm is penalised with -1 "wins". As all the algorithms are compared against each other, they are ranked (depending on how many other configuration are better/worse).

- The evolution of the participation of the different techniques in hybrid algorithms (both standard MOS and MOSRL) is analysed to check how different search strategies can boost the performance of individual techniques and how this participation evolves in the case of the RL algorithms.

6 Results and Discussion

Table 4 presents the average error obtained by each of the configurations on the six proposed problems. For each column, the smallest error is shown in bold. From these data, it can be observed that one of the RL policies, PHC, obtains the best results in 4 out of the 6 considered functions. In the other two functions, one of the single GAs, the BCUM configuration (BLX-α Crossover + Uniform Mutation), reports the lowest average error. The differences in performance between PHC and the best single GA ranges from 4% to 40% for the four functions where it obtains lower average error and from 1% to 8% in the other two functions. The other RL policies (WoLF and TERSQ) report worse average errors as a result of their more conservative behavior, as will be seen when the participation plots are analysed.

Table 4: Average error on the six proposed functions when the four reproductive techniques are considered.

	Ackley	Griewank	Rastrigin	Rosenbrock	Schwefel	Sphere
BCGM	8.80e+00	2.26e+02	2.19e+03	1.46e+09	4.82e+01	2.65e+04
BCUM	3.17e+00	**8.84e+00**	1.24e+03	7.65e+06	3.61e+01	**9.14e+02**
UCGM	8.08e+00	1.78e+02	5.67e+02	5.84e+08	3.82e+01	2.02e+04
UCUM	3.67e+00	1.58e+01	2.29e+02	4.18e+06	3.21e+01	1.68e+03
MOS Const	5,31e+00	3.57e+01	6,52e+02	4,04e+07	4,54e+01	4,08e+03
MOS Dyn	4.41e+00	1.94e+01	4.89e+02	7.92e+06	3.78e+01	2.22e+03
PHC	**3.04e+00**	9.57e+00	**2.22e+02**	**2.51e+06**	**3.18e+01**	9.56e+02
WoLF	3.98e+00	1.85e+01	3.13e+02	1.27e+07	3.67e+01	2.11e+03
TERSQ						
Tentative	4.80e+00	3.17e+01	5.28e+02	1.78e+07	3.76e+01	3.52e+03
Adjustment	4.63e+00	2.89e+01	5.21e+02	1.84e+07	3.80e+01	3.33e+03
Optimal	4.31e+00	2.82e+01	5.01e+02	1.74e+07	3.77e+01	3.21e+03

It is interesting to note that at least two of the single GAs report average errors with the same order of magnitude for the four functions where the PHC policy obtains the best results. In the other two functions, one of the single GAs obtains significantly better results than the others, with an average error at least one order of magnitude lower than the rest of the GAs. This can

explain that, even if the hybrid approaches are able to detect this difference of performance, they still waste some valuable fitness evaluations with the less performant techniques, especially at the early stages of the search process.

Figs. 1, 2, 3 and 4 present a comparative view of the participation progress for two of the six proposed functions. These problems have been selected as representative of the remaining functions. Regarding the problems for which plots have not been provided, they exhibit similar behaviors to those observed in the Sphere function in the case of Ackley, Griewank and Rosenbrock functions and to those observed in the Rastrigin function in the case of the Schwefel problem.

Fig. 1 presents the participation progression of the four hybrid approaches in the Sphere function. For each of the RL policies, two snapshots of the learning procedure are provided: after 61 executions and after 111 executions. The first snapshot coincides with the end of the adjustment phase of the TERSQ policy, while the second one is taken at the end of the last execution. There are similarities and differences among the different learning policies. The first one is how differently PHC performs participation transitions compared to the other three hybrid approaches. This policy, which obtained the best results in this problem among the hybrid approaches, systematically swaps between BCUM and UCUM techniques with participation ratios that are close to 1 for one of the techniques in most of the cases. On the other hand, the other two RL policies are more conservative in this respect, especially the TERSQ policy that, in the last executions, almost performs an uniform distribution of the participation. From the results reported in Table 4 it seems that, the more conservative the algorithm, the worse average error it obtains.

Fig. 2 depicts the participation progression for the four hybrid algorithms on the Rastrigin function. In this problem, the behavior of the PHC policy is different to the one it had in the previous function. In this case, the systematic exchange of algorithms lasts only until generation $10,000 - 15,000$. From this point, the UCUM technique executes most of the time, with occasional collaboration of the UCGM technique. WoLF presents a similar behavior but, as we have seen in the Sphere function, with a more conservative strategy. The fluctuations are, in this case, more remarkable than in the previous problem but not as much as with the PHC policy. The other two hybrid approaches present a more conservative behavior, as in the previous problem. It is important to note that these two algorithms assign more participation to the UCGM than to the UCUM unlike PHC and WoLF policies. This seems to be directly correlated with the average error reported by each algorithm, as the performance of PHC and WoLF on this problem is comparable, while the TERSQ and MOS Dynamic approaches obtain similar average errors.

In Table 5 the results of both ranking analyses are presented. The first column, *Average Ranking*, orders the algorithms based on their average error in the six proposed functions. The second column, *Wins*, reports the number of wins for each algorithm in the pair-wise statistical comparison carried out. Both rankings provide more or less the same information, except for a small difference in the ranking of the WoLF and BCUM algorithms (WoLF obtains a

worse ranking according to its average error but a better one according to its statistical results). These results confirm that there is statistical evidence to establish that the PHC policy to learn the hybridization strategy obtains the best results for this benchmark. However, as the WoLF and TERSQ strategies, as well as the MOS Dynamic approach, obtained worse results, in general, than the UCUM and BCUM individual techniques, a second experimental phase was carried out, considering only these two techniques for hybridization.

Table 5: Ranking and statistical test results for both the single and the hybrid algorithms for the first experimental configuration.

	Average Ranking	Wins
PHC	1.33	7
UCUM	2.50	5
BCUM	3.00	1
WoLF	4.00	2
MOS Dyn	5.00	1
TERSQ	5.50	0
MOS Const	7.17	-4
UCGM	7.50	-4
BCGM	9.00	-8

Table 6 shows the results obtained when only BCUM and UCUM are taken into account by the hybrid approaches. The first thing that can be observed from these results is that, with this configuration of techniques, there is always an RL algorithm that obtains better results than the individual algorithms. The second remark is that, in this second experiment, the differences between the PHC policy and the other two hybrid algorithms have been considerably reduced. Furthermore, now the TERSQ algorithm obtains lowest average error in as many functions as PHC does. WoLF policy continues to obtain worse results than the other two RL policies. However, the gap among them is smaller now.

Table 6: Average error on the six proposed functions when only BCUM and UCUM techniques are considered.

	Ackley	Griewank	Rastrigin	Rosenbrock	Schwefel	Sphere
MOS Const	3.15e+00	9.55e+00	3.87e+02	3.95e+06	3.66e+01	1.01e+03
MOS Dyn	3.14e+00	9.47e+00	2.42e+02	3.75e+06	3.50e+01	1.01e+03
PHC	3.01e+00	9.10e+00	**2.07e+02**	**2.38e+06**	**3.12e+01**	9.43e+02
WoLF	2.95e+00	**8.47e+00**	2.63e+02	2.42e+06	3.44e+01	8.85e+02
TERSQ						
Tentative	2.93e+00	8.37e+00	2.82e+02	2.35e+06	3.43e+01	8.76e+02
Adjustment	2.94e+00	8.55e+00	2.73e+02	2.40e+06	3.43e+01	8.76e+02
Optimal	**2.93e+00**	8.48e+00	2.39e+02	**2.38e+06**	3.41e+01	**8.79e+02**

In Table 7 the results of both the average error ranking and the statistical ranking analysis are presented. This analysis confirms the impression derived from the results collected in Table 6. With this configuration it is the TERSQ policy and not the PHC policy which obtains the best ranking, considering both the average error and the statistical tests. Moreover, the four dynamic

Table 7: Ranking and statistical test results for both the single and the hybrid algorithms for the second experimental configuration.

	Average Ranking	Wins
TERSQ	1.92	6
PHC	2.42	5
WoLF	2.83	3
MOS Dyn	4.58	1
UCUM	5.17	1
BCUM	5.50	-1
MOS Const	5.75	-2
UCGM	7.83	-5
BCGM	9.00	-8

hybrid algorithms (the three RL policies and the hybrid algorithm with dynamic adjustment of participation) obtain better rankings than any of the individual techniques. Only the hybrid algorithm with constant participation ratios obtains worse results than UCUM and BCUM single techniques.

Fig. 3 depicts the evolution of the participation of each of the techniques using the four dynamic approaches. In this case, the lowest average error was obtained by the TERSQ policy, followed by the WoLF policy. These two RL algorithms have in common that their adjustment of the participation of the two available techniques is not very significant, practically static from the generation $3,000 - 4,000$. If we compare the results of these two policies on this function with that of the hybrid approach witha a constant participation ratio we can see that, even though the participation is mostly static from generation $3,000 - 4,000$ on, the results obtained by these two algorithms are up to a 15% better. This means that the hybrid strategy learnt for these generations, where the UCUM initially receives a slightly higher participation ratio, is responsible for improving the final results compared to a completely constant participation assignment.

In Fig. 4 the evolution of the participation is presented for the Rastrigin function. In this problem, the behavior of every algorithm is completely different to that exhibited in the previous function. In this problem, all the algorithms, to a greater or lesser extent, make a much more abrupt participation adjustment. Again, the algorithms with a more conservative behavior obtain worse results.

Comparing the participation plots of the Sphere function (Figs. 1 and 3) it can be observed that, apart from the absence of the BCGM and UCGM techniques in the second plot, the behavior of the four hybrid algorithms is quite similar with four and two techniques. This does not happen in the case of the Rastrigin function (Figs. 2 and 4). In this function, the configurations with four techniques boosted the participation of the techniques UCUM and UCGM. For the second experiment, the UCGM technique was not used (we only considered the two techniques with best average individual performance). Despite this, the results obtained by the hybrid approaches with only two techniques are statistically better ($p - value < 0.05$) than those obtained with four techniques, which suggest again the idea that the real contribution of the other techniques to

that leading the search process takes place at the very beginning of the execution or, eventually, with brief phases where these techniques take over for a small period of time to create individuals that increase the diversity of the population. For this purpose, either UCGM or BCUM seem to add enough diversity to the population to let the leading technique converge to better final solutions.

7 Conclusions

In this paper, a new hybrid algorithm with dynamic adjustment of participation by means of Reinforcement Learning techniques has been presented. The experimental results show statistical evidence that the regulation of the participation with RL techniques can boost the performance of the new RL hybrid algorithm compared to both traditional hybrid algorithms and individual genetic algorithms.

The three proposed policies have been able to learn, to different degrees, the most efficient strategy for the combination of the available reproductive techniques. However, some differences are exhibited by these three policies. PHC seems to be more drastic in its decisions, while WoLF and TERSQ show a more conservative behavior in this sense. If we observe how the probability of selecting an action within the PHC policy is updated (step 6 of Algorithm 3), we can see that after a few executions the probability values for the best action and the rest of the actions quickly converge to one and zero, respectively. This can explain the drastic behavior of the PHC policy and thus a further research into the selection of the learning ratios should be considered. Taking these considerations into account, PHC seems to be more appropriate in contexts where more techniques are available and a quick detection of the best technique(s) is crucial for the performance of the algorithm. On the other hand, the other two policies seem to be more suitable when the performance of the available techniques is similar and small differences should be taken into account.

As future work, it would be interesting to extend this research to a larger number of functions of different dimensionality, as well as carrying out tests with larger and more diverse sets of techniques (including other evolutionary approaches such as Estimation of Distribution Algorithms, Differential Evolution, etc.).

Finally, even though MOSRL performance has been improved by using RL policies, it could be argued that a performance comparison between traditional EAs and MOSRL is not fair, as the latter takes advantage of the results obtained in previous executions in order to learn the best strategy to obtain the final result. However, the interest of this research is not only to state whether a hybrid evolutionary algorithm guided by RL techniques can obtain better results than classic algorithms but also if they are able to learn the best hybrid strategy for a given problem. This could be of useful application to real-world problems that have to be solved hundreds of times with slightly different input data, such as planning or scheduling problems.

8 Acknowledgments

The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the *Centro de Supercomputación y Visualización de Madrid (CeSViMa)* and the Spanish Supercomputing Network.

This work was supported in part by the Ministry of Education and Science under grants no. AP-2004-0949 and TIN2007-67148.

References

[1] S. Abdallah and V. Lesser. A Multiagent Reinforcement Learning Algorithm with Non-linear Dynamics. *Journal of Artificial Intelligence Research*, 33:521–549, 2008.

[2] M. Bowling. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems*, volume 17, pages 209–216. MIT Press, 2005.

[3] M. Bowling and M. Veloso. Convergence of gradient dynamics with a variable learning rate. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 27–34. Morgan Kaufmann, 2001.

[4] M. Bowling and M. Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1021–1026, August 2001.

[5] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[6] V. Conitzer and T. Sandholm. Bl-wolf: A framework for loss-bounded learnability in zero-sum games. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 91–98, 2003.

[7] S. Droste, T. Jansen, and I. Wegener. Optimization with randomized search heuristics - the (a)nfl theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, 287(1):131–144, 2002.

[8] A. Eiben, M. Horvath, W. Kowalczyk, and M. Schut. Reinforcement learning for online control of evolutionary algorithms. In *Proceedings of the 4th International Workshop on Engineering Self-Organising Systems, ESOA 2006*, volume 4335 of *Lecture Notes in Computer Science*, pages 151–160. Springer-Verlag GmbH, 2007.

[9] E. Even-Dar and Y. Mansour. Learning rates for q-learning. *Journal of Machine Learning Research*, 5:1–25, 2004.

[10] C. Grosan and A. Abraham. Hybrid evolutionary algorithms: Methodologies, architectures, and reviews. In C. Grosan, A. Abraham, and

H. Ishibuchi, editors, *Hybrid Evolutionary Algorithms*, volume 75 of *Studies in Computational Intelligence*, pages 1–17. Springer-Verlag GmbH, 2007.

[11] F. Herrera and M. Lozano. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In F. Herrera and J. Verdegay, editors, *Genetic Algorithms and Soft Computing*, pages 95–125. Physica-Verlag, 1996.

[12] T. Hong, H. Wang, and W. Chen. Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of Heuristics*, 6(4):439–455, September 2000. Kluwer Academic Publishers, Hingham, MA, USA.

[13] T. Hong, H. Wang, W. Lin, and W. Lee. Evolution of appropriate crossover and mutation operators in a genetic process. *Applied Intelligence*, 16(1):7–17, January 2002. Springer Netherlands.

[14] T. Jaakkola, M. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

[15] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, 1995. Morgan Kaufmann.

[16] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[17] S. Koh, S. Leow, and K. Loke. An adaptive genetic algorithm for permutation based optimization problems. In *Proceedings of the AIAI2005- Second IFIP Conference on Artificial Intelligence Applications and Innovations*, Beijing, September 2005.

[18] A. LaTorre, J. Peña, S. González, V. Robles, and F. Famili. Breast cancer biomarker selection using multiple offspring sampling. In *Proceedings of the ECML/PKDD 2007 Workshop on Data Mining in Functional Genomics and Proteomics: Current Trends and Future Directions*, Warsaw, Poland, September 2007. Springer Verlag.

[19] A. LaTorre, J. Peña, S. Muelas, and M. Zaforas. Hybrid evolutionary algorithms for large scale continuous problems. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO 2009*, 2009.

[20] A. LaTorre, J. Peña, V. Robles, and P. de Miguel. Supercomputer scheduling with innovative evolutionary techniques. In F. Xhafa and A. Abraham, editors, *Meta-heuristics for Scheduling: Distributed Computing Environments*, volume 146 of *Studies in Computational Intelligence*. Springer Verlag, Germany, 2008.

[21] A. LaTorre, J. Peña, V. Robles, and S. Muelas. Using multiple offspring sampling to guide genetic algorithms to solve permutation problems. In M. Keijzer, editor, *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO 2008*, pages 1119–1120, New York, NY, USA, July 2008. ACM Press.

[22] A. Nareyek. Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer Decision-Making*, pages 523–544. Kluwer Academic Publishers, 2004.

[23] M. Oltean. Searching for a practical evidence of the no free lunch theorems. In *First International Workshop on Biologically Inspired Approaches to Advanced Information Technology, BioADIT 2004*, volume 3141 of *Lecture Notes in Computer Science*, pages 472–483. Springer-Verlag GmbH, 2004.

[24] L. Peña, A. LaTorre, J. Peña, and S. Ossowski. Tentative exploration on reinforcement learning algorithms for stochastic rewards. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligent Systems*, Lecture Notes in Computer Science. Springer-Verlag GmbH, June 2009.

[25] V. Robles, J. Peña, P. Larrañaga, M. Pérez, and V. Herves. Ga-eda: A new hybrid cooperative search evolutionary algorithm. In J. Lozano, P. Larrañaga, I. Inza, and E. Bengotxea, editors, *Towards a New Evolutionary Computation. Advances in the Estimation of Distribution Algorithms. Series: Studies in Fuzziness and Soft Computing*, volume 192 of *Lecture Notes in Computer Science*, pages 187–219. Springer-Verlag GmbH, 2004.

[26] T. Schnier and X. Yao. Using multiple representations in evolutionary algorithms. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 479–486, La Jolla, CA, USA, July 2000. IEEE Press.

[27] X. Shi, Y. Liang, H. Lee, C. Lu, and L. Wang. An improved ga and a novel pso-ga-based hybrid algorithm. *Information Processing Letters*, 93(5):255–261, March 2005.

[28] S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 541–548. Morgan, 2000.

[29] A. Sinha and D. Goldberg. A survey of hybrid genetic and evolutionary algorithms. Technical Report 2003004, Illinois Genetic Algorithms Laboratory (IlliGAL), January 2003.

[30] E. Takashima, Y. Murata, N. Shibata, and M. Ito. Self adaptive island ga. In *Proceedings of the 2003 Congress on Evolutionary Computation, CEC '03*, volume 2, pages 1072–1079. Springer-Verlag GmbH, December 2003.

[31] K. Tang, X. Yao, P. Suganthan, C. MacNish, Y. Chen, C. Chen, and Z. Yang. Benchmark functions for the cec 2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.

[32] G. Tesauro. Extending q-learning to general adaptive multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 16, pages 871–878. MIT Press, 2004.

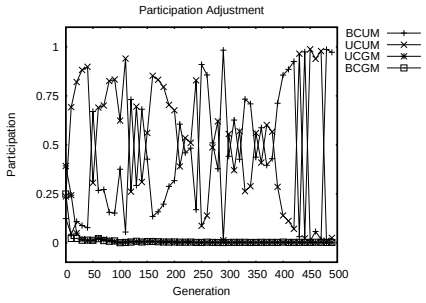
[33] L. Tseng and S. Liang. A hybrid metaheuristic for the quadratic assignment problem. *Computational Optimization and Applications*, 34(1):85–113, May 2006.

[34] L. Vanneschi, M. Tomassini, P. Collard, and S. Vérel. Negative slope coefficient : A measure to characterize genetic programming fitness landscapes. In P. C. et al., editor, *Proceedings of the 9th European conference on Genetic Programming, EuroGP 2006*, volume 3905 of *Lectures Notes in Computer Science*, pages 179–189. Springer-Verlag GmbH, April 2006.

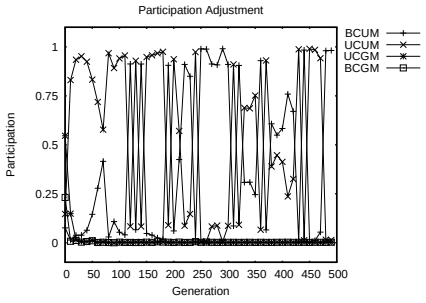
[35] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[36] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

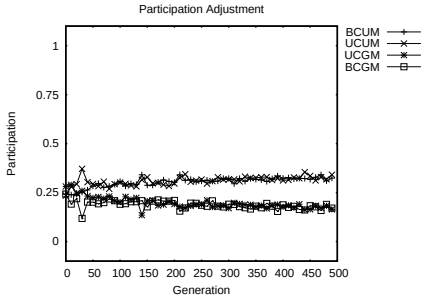
[37] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936, 2003.



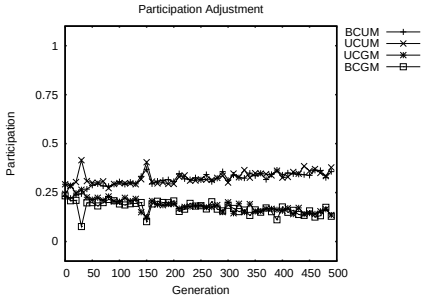
(a) PHC after 61 executions



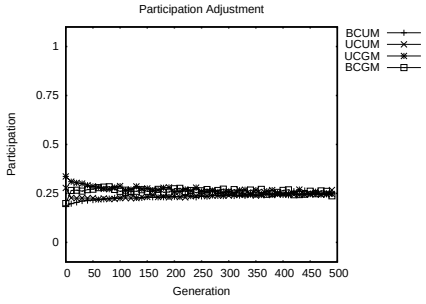
(b) PHC after 111 executions



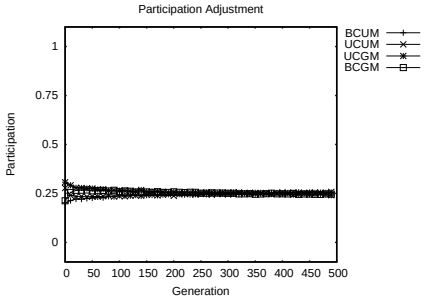
(c) WoLF after 61 executions



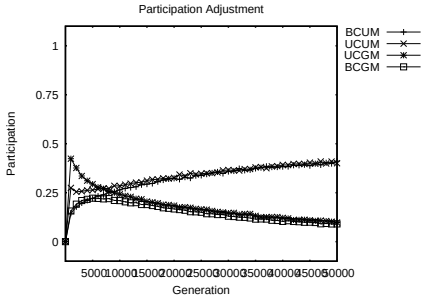
(d) WoLF after 111 executions



(e) TERSQ after 61 executions

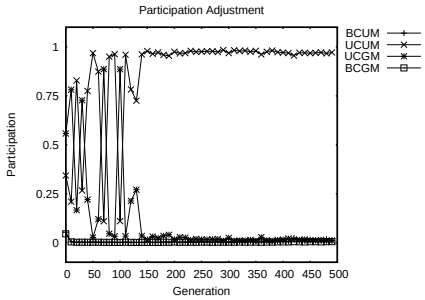


(f) TERSQ after 111 executions

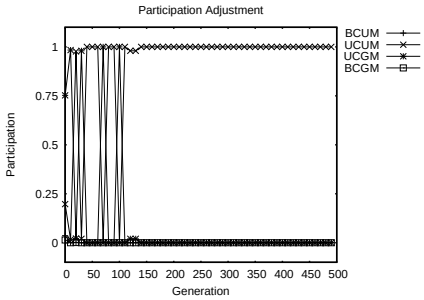


(g) MOS Dynamic after 111 executions

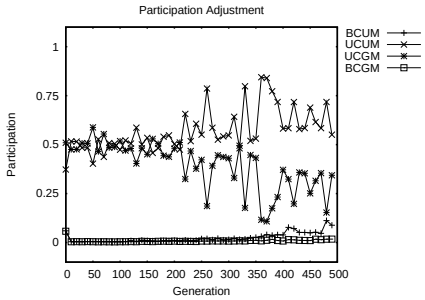
Fig. 1: Participation Progress of hybrid algorithms in the Sphere function



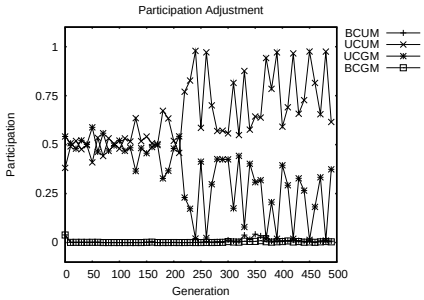
(a) PHC after 61 executions



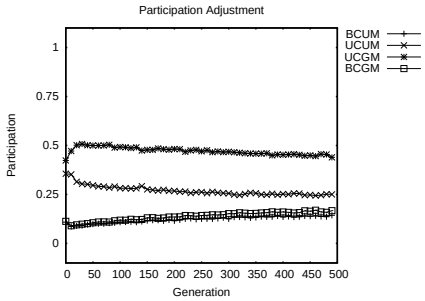
(b) PHC after 111 executions



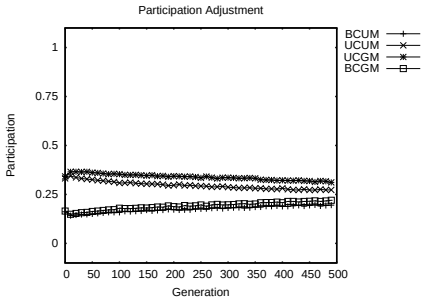
(c) WoLF after 61 executions



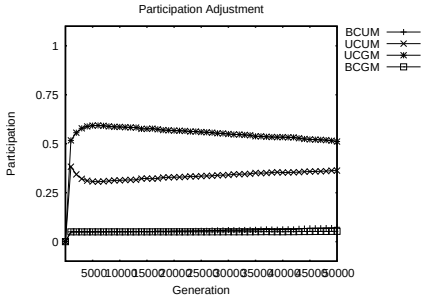
(d) WoLF after 111 executions



(e) TERSQ after 61 executions

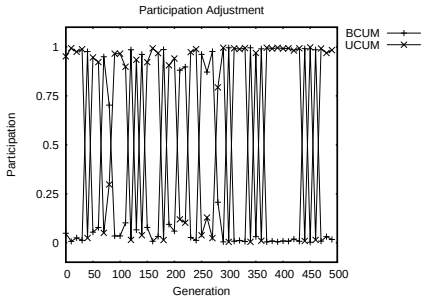


(f) TERSQ after 111 executions

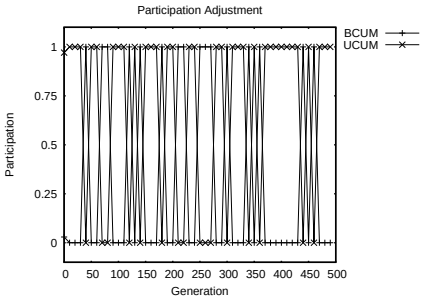


(g) MOS Dynamic after 111 executions

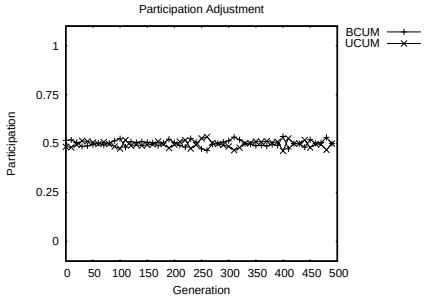
Fig. 2: Participation Progress of hybrid algorithms in the Rastrigin function



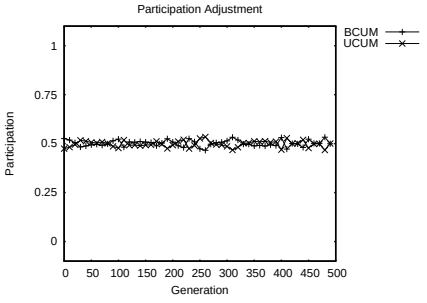
(a) PHC after 61 executions



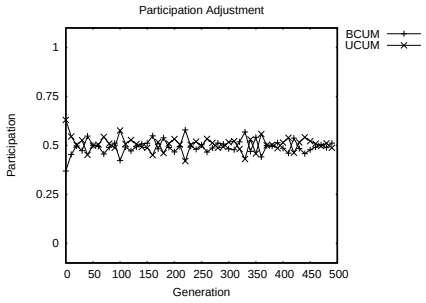
(b) PHC after 111 executions



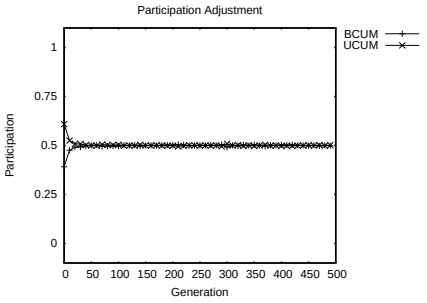
(c) WoLF after 61 executions



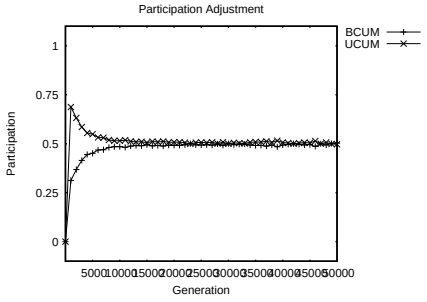
(d) WoLF after 111 executions



(e) TERSQ after 61 executions

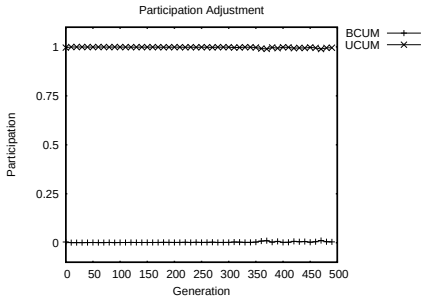


(f) TERSQ after 111 executions

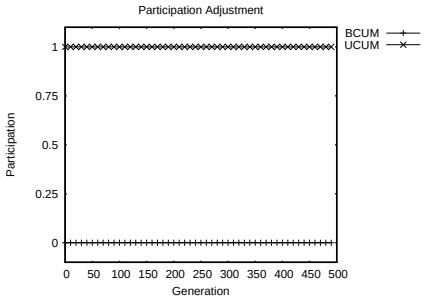


(g) MOS Dynamic after 111 executions

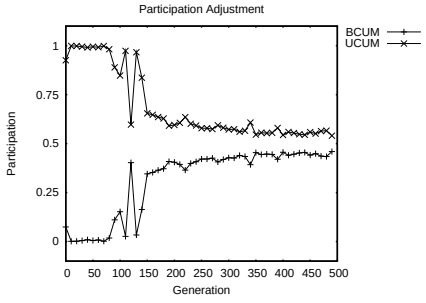
Fig. 3: Participation Progress of hybrid algorithms in the Sphere function



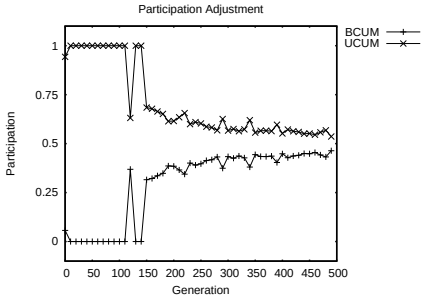
(a) PHC after 61 executions



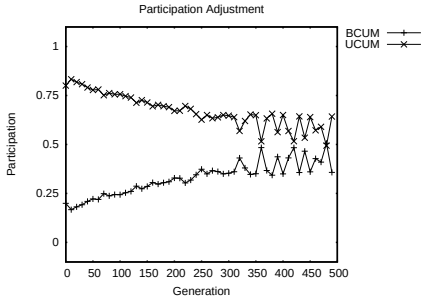
(b) PHC after 111 executions



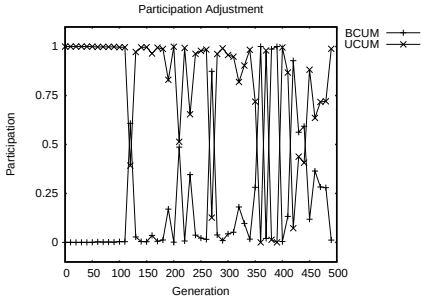
(c) WoLF after 61 executions



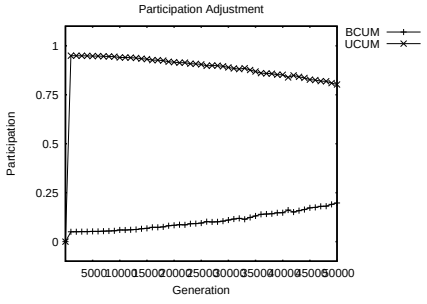
(d) WoLF after 111 executions



(e) TERSQ after 61 executions



(f) TERSQ after 111 executions



(g) MOS Dynamic after 111 executions

Fig. 4: Participation Progress of hybrid algorithms in the Rastrigin function