Out-of-the-box Parameter Control for Evolutionary and Swarm-based Algorithms with Distributed Reinforcement Learning

Ph.D. Thesis Defense

Ph.D. Candidate: Marcelo Gomes Pereira de Lacerda, M.Eng.
Supervisor: Prof. Teresa Bernarda Ludermir, Ph.D.
Co-Supervisor: Prof. Fernando Buarque de Lima Neto, Ph.D

March 2021

Introduction

Introduction

- **Evolutionary** (EA) and **swarm-based** (SI) algorithms are very successful metaheuristics used to solve NP-hard optimization problems.
- Their performances strongly depend on their parameters.
- **No Free Lunch Theorem**: No parameter setup is optimal for every optimization problem.
- Manual parameter adjusment can be very **hard**, **tedious**, and **costly**.
- Automatic parameter adjustment:
  - Tuning methods;
  - **Control methods**.

Introduction

- Parameter control methods:
  - Control methods tailored to specific problems;
  - **Out-of-the-box control methods.**
- The vast majority of the studies: methods tailored to specific problems.
- Very few studies propose truly out-of-the-box methods.
- Most of them use **Reinforcement Learning** to find the **control policy**.

Introduction

- RL-based approaches have presented the most promising results.
- However, the problem of out-of-the-box parameter control is far from being solved.
- Issues identified in the literature:
  - Training parameter control policies for EA and SI with RL can be very **computationally demanding**.
  - RL algorithms usually require the adjustment of many **hyperparameters**. Also, the search for an optimal policy can be very **unstable**.
  - Very limited benchmarks have been used to assess level of **generality** of the proposals.

Introduction

General objective: Propose an **out-of-the-box policy training method** for parameter control of mono-objective EA and SI algorithms with **distributed Reinforcement Learning**.

Introduction

- **Specific objectives:**
  - Propose a **scalable parameter control policy training** method that produces policies that can be used in an **unseen** problem.
  - Propose a mechanism that diminishes the **difficulties** on the hyperparameter adjustment and reduces the **instability** of the learning process.
  - Assess the **generality** of the proposed method in an experimental benchmark with **truly** varied scenarios.

Background

Background

- Reinforcement Learning
  - Interaction of agent with an environment.
  - The agent takes actions and receives rewards.
  - The agent improves its performance by trial-and-error.

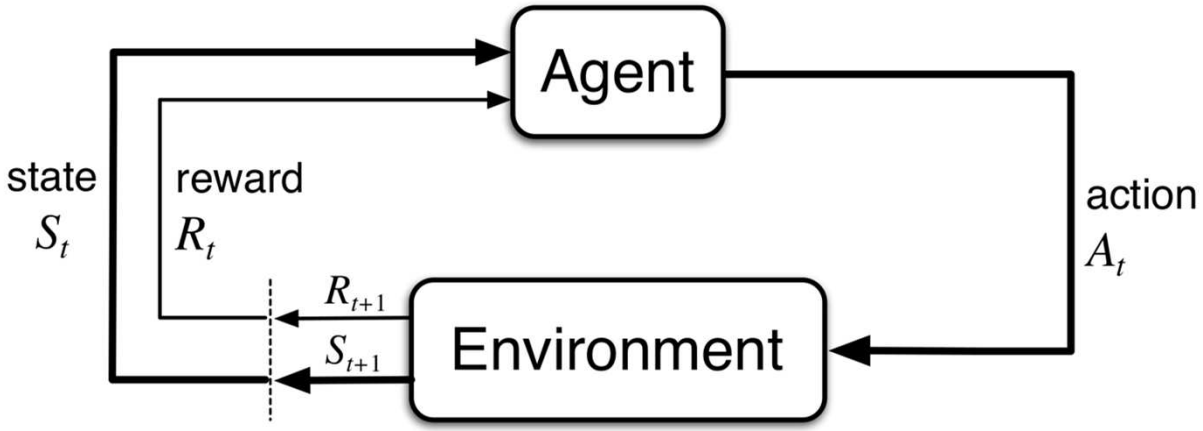


Figure taken from (SUTTON; BARTO, 2018).

Background

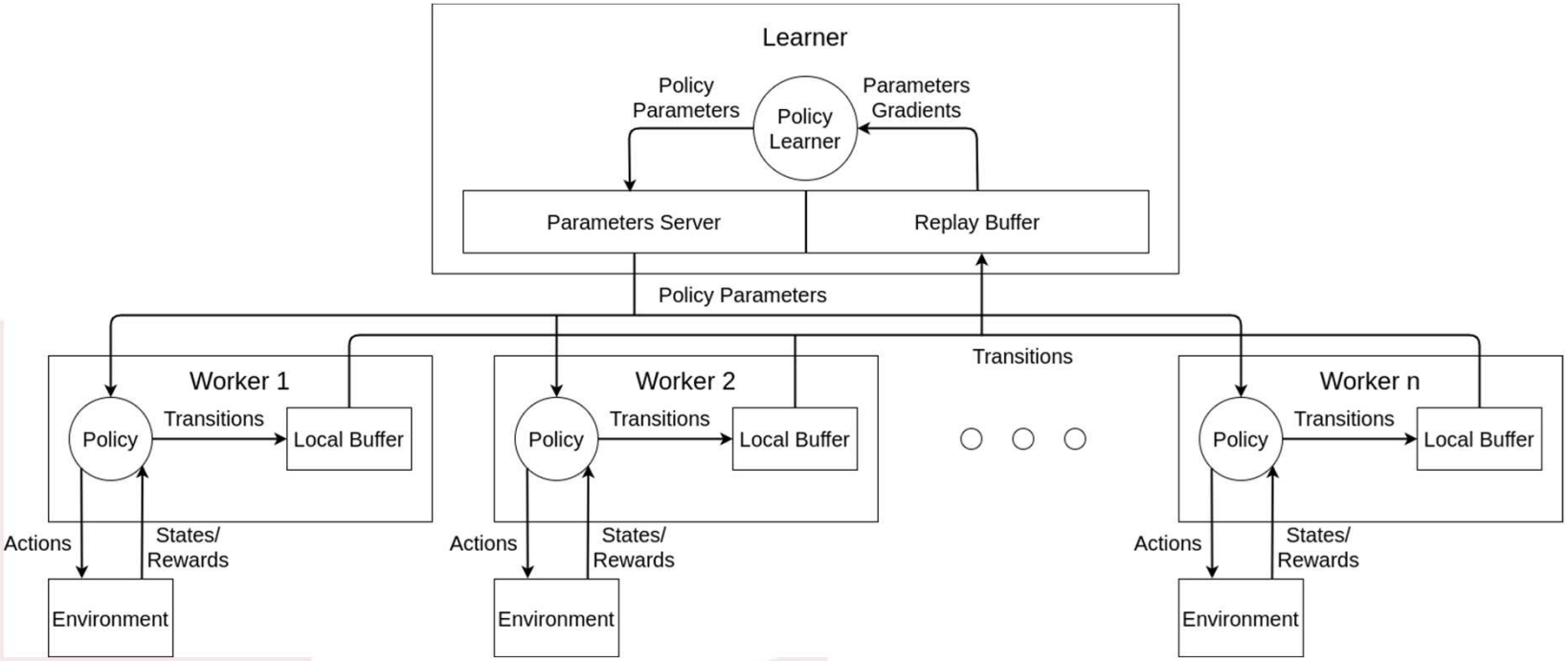
- Reinforcement Learning
  - The agent **learns** the **optimal policy** throughout the training process.
  - **Optimal Policy**: Best **action** for each state of the **environment**.
  - **Tabular** representation: map each state for each best action.
  - Defining all possible states and the expected return for each action can be **unfeasible**.
  - **Artificial Neural Networks** (ANNs) can be used to approximate policies.

Background

- Reinforcement Learning
  - General rationale:
    1. An **initial policy** is created;
    2. The agent **collects experiences** interacting with the environment with the current policy;
       - Experience: previous state, action taken, new state, and the reward received.
    3. The collected experiences are used to **train** the agent's ANNs. Then, the algorithm's flow goes back to step 2.

Background

- Reinforcement Learning
  - Training such ANNs can be very **costly**.
    - Many RL algorithms are know to be very **sample-inefficient**.
  - **Distributed prioritized** experience replay: Ape-X.



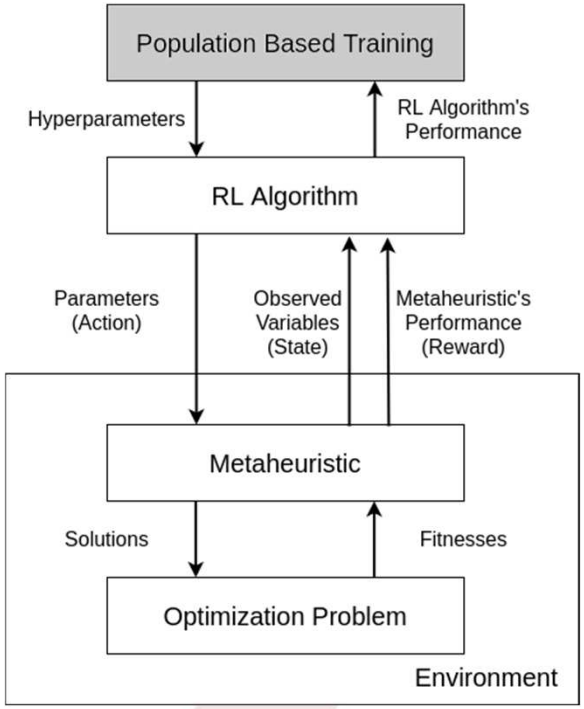
Background

- Population-based Training
  - RL algorithms usually have many **hyperparameters** to be adjusted.
  - They are **too sensitive** to them.
  - They usually present an **unstable learning process.**.
  - Population-based Training (PBT):
    - **Multiple training processes** run in parallel.
    - They exchange **parameters** and **hyperparameters** periodically.
      - Evolution of parameters and hyperparameters.
      - PBT is a hyperparameter control method for ML.
    - In RL, PBT brings more **diversity** to the policy search.
      - Reduces the instability of the policy search;
      - Increases the probability of finding a very good policy.

Training Parameter Controllers with Distributed Reinforcement Learning

Training Parameter Controllers with Distributed Reinforcement Learning

- **Out-of-the-box** training methodology for parameter control policy for EA and SI algorithms with **distributed Reinforcement Learning** and **PBT**.
- PBT evolves the parameters and hyperparameters of an RL algorithm, which learns an optimal policy for parameter control for EA and SI: **two levels of control** above the metaheuristic.



Training Parameter Controllers with Distributed Reinforcement Learning

- **State** variables: 254 floating-point values.
- **Action** space: Continuous space with dimensions between zero and one.

$$p'_i = p_i(p_{i,max} - p_{i,min}) + p_{i,min}$$

- **Reward** function (proposed by Schuhardt *et al.* in 2019):
  - Returns a **positive** value if the **best fitness** among the individuals **increases** between iterations.
  - Returns a **negative** value if the **best fitness decreases** between iterations.

$$r(s_t, a_t) = \alpha_r log_{10} \frac{F_{max}(s_t)}{F_{max}(s_{t-1})}$$

Training Parameter Controllers with Distributed Reinforcement Learning

- Learning the control policy
  - Policies are trained to be used with a **metaheuristic** to solve an **unseen** (unknown) problem.
  - *n* **known** functions: 1 **training** function and *n-1* **validation** functions.
  - After each **training epoch**:
    - **Validation** of the current policy;
    - The current policy is saved in a **pool**.
  - After the training process, the pool is full of policies trained with 1 training function.
  - How can we populate the **pool** with policies trained with **different** functions?
    - Use every known function once as **training** function.
  - In the end, a **single** policy must be **selected**.

Training Parameter Controllers with Distributed Reinforcement Learning

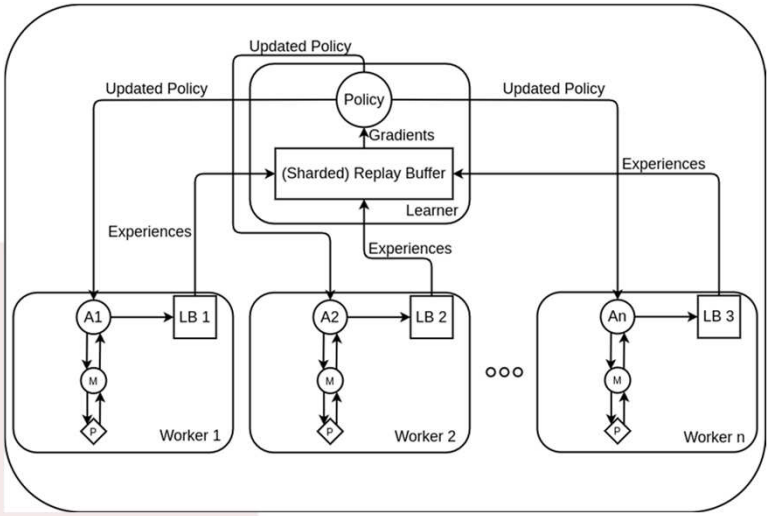
- Choosing **one policy** among the **pool of policies**:
  - Single score for each policy.
  - The policy with the **highest** score is chosen.
  - Score of a policy *p* trained with function *g*:
    - For **each validation function *f***, do:
      - We divide the number of policies that performed **worse** than *p* in *f* by the number of policies in the pool, except *p* and the policies trained with *f* itself.
    - **Average** all values.
  - The score of a policy is the average **percentile** of its **performance** among all validation functions, comparing it with all **valid** policies.
  - **How well does this policy perform in previously seen functions different from the problem it was trained for.**

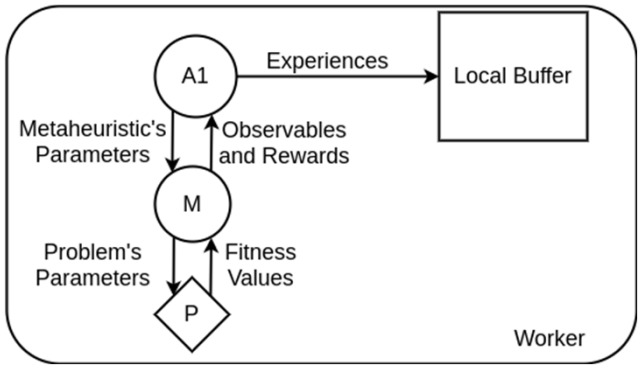
Training Parameter Controllers with Distributed Reinforcement Learning

- Choosing **one policy** among the **pool of policies**:
  - Example:
    - We **know** functions 1, 2, 3, and 4.
    - Trained *p* on function 1.
    - **Validation** on function 2: *p* is the worst policy among the **12%** best policies.
    - **Validation** on function 3: *p* is the worst policy among the **8%** best policies.
    - **Validation** on function 4: *p* is the worst policy among the **10%** best policies.
    - Average percentile: 1 - (**0.12**+**0.08**+**0.1**) / 3 = 0.9
    - Score of *p*: **0.9**

Training Parameter Controllers with Distributed Reinforcement Learning

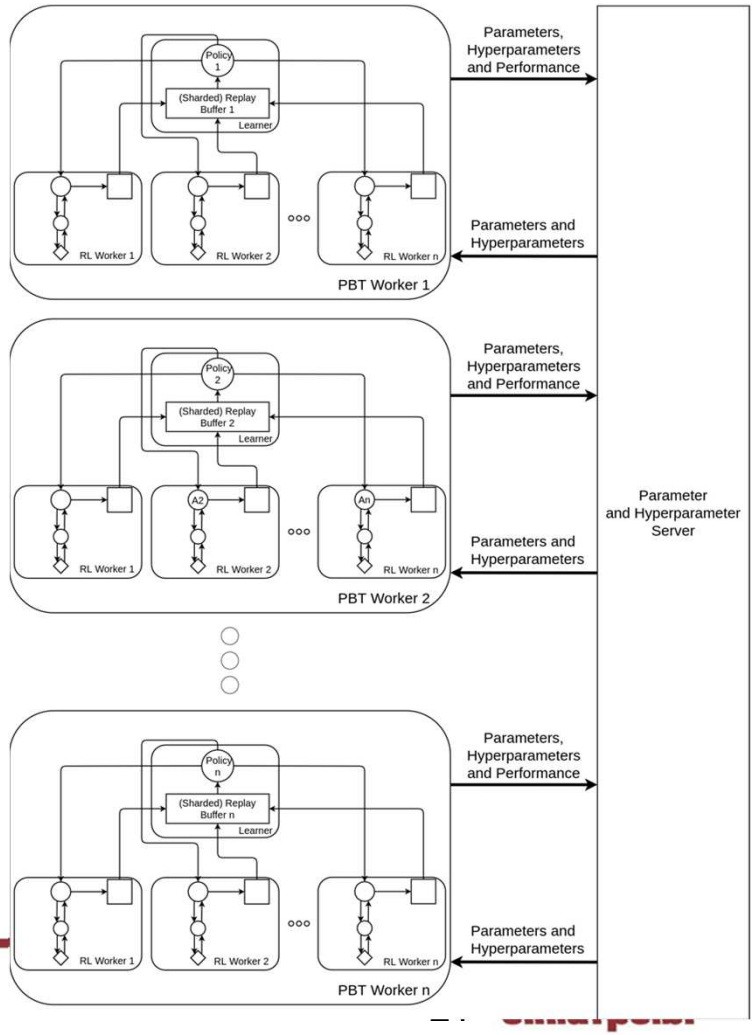
- **Distributed** architecture (issue #1): RL layer.
  - **Out-of-the-box distributed architecture** for the training process of parameter control policies for EA and SI.
  - Based on Ape-X.





Training Parameter Controllers with Distributed Reinforcement Learning

- **Distributed** architecture (issue #2): PBT layer.
  - **Distribute** the training process;
  - **Reduce the burden** of hyperparameter adjustment for the RL algorithm;
  - **Increase the probability** of finding and keeping a **good** policy;
  - **Reduce the instability** of the learning process.



Experimental results and discussion

Experimental results and discussion

- Experimental methodology
  - RL algorithm used: **TD3**
    - State-of-the-art.
    - Continuous action spaces;
  - PBT's hyperparameters setting:
    - Perturbation interval: 4;
    - Quantile fraction: 0.125;
    - Resample probability: 0.5.
    - PBT workers: 16.
    - RL workers: 2.
  - They were defined after a **hyperparameter analysis** and a few other preliminary experiments.
  - The **five** most important TD3's hyperparameters were controlled by PBT. Example: learning rates, L2 regularization factor, etc.
  - The TD3's **fixed** hyperparameters are available in Section 5.1.2 in the thesis.
  - A budget of 24 hours was given to all training processes.

Experimental results and discussion

- Experimental methodology
  - Choosing the Metaheuristics and the Optimization Problems
    - HCLPSO, FSS, and DE: 29 bound constrained benchmark problems with 10 dimensions from CEC17 (**continuous** objective functions);
    - Binary GA: 23 instances of the Knapsack problem 20 and 50 cities (**binary** objective functions);
    - ACO: 23 instances of the Traveling Salesman Problem with 10, 30, and 50 cities (**combinatorial** objective functions).
  - Total: **133** varied scenarios (issue #3).

Experimental results and discussion

- Experimental methodology
  - Compare our policies with:
    - **Random** policy;
      - The parameters are set randomly each iteration.
    - **Static** policies tuned by a distributed implementation of **I/F-Race**;
      - I/F-Race: state-of-the-art tuning algorithm;
      - The evaluations of the surviving configurations for each iteration are performed in parallel;
      - All CPUs are used (virtual and physical).
    - **Human-designed** policies.
      - Parameters defined according to relevant papers about the algorithms.
  - Selected **policy** and the **best** policy.
  - **Budget** for one training/testing episode: 300 iterations.

Experimental results and discussion

- Experimental methodology
  - **All parameters** of each algorithm were controlled, except the **population size**, which was set to 100 for all algorithms.
  - The complexities added to the process when individuals are **created** or **removed** are out of the scope of this work.
  - However, advances have been made in:
    - "Population Size Control for Efficiency and Efficacy Optimization in Population Based Metaheuristics" - Status: **published**.
    - "Towards a Parameterless Out-of-the-box Population Size Control for Mono-Objective Metaheuristics" - Status: **submitted**.

Experimental results and discussion

- Results and discussions:
  - The experiments were divided into three parts:
    - **Hyperparameter** analysis;
    - Analysis of **different episode budgets** in the training and the testing phases;
    - **Generality** assessment.
  - The **performance** of a given policy in a given unseen function is measured by the **fitness of the best solution ever found** after 300 iterations of the metaheuristics.
  - Every **comparison** between two policies in a given **benchmark function** was made by comparing their performances on 30 independent runs.
    - **Wilcoxon Rank-sum** test (samples with the same sample sizes) or **Mann-Whitney** test (samples with different sample sizes).

Experimental results and discussion

- Results and discussions:
  - Performance of the **proposed method** in a given function **A**: performance of a policy selected among all policies that were **not** trained with **A**.
  - Policy A **significantly overcame** policy B: **p-value** computed by comparing the performances of A and B is **lower or equal than 0.05**.

Experimental results and discussion

- Results and discussions:
  - Hyperparameter analysis:
    - Only **HCLPSO** is used with the 29 CEC17 functions.
    - **Best** policy.
    - Each of the five hyperparameters of PBT were **varied** while the others were kept **constant** with the default values.
      - Perturbation interval: 4;
      - Quantile fraction: 0.125;
      - Resample probability: 0.5.
      - PBT workers: 16.
      - RL workers: 2.

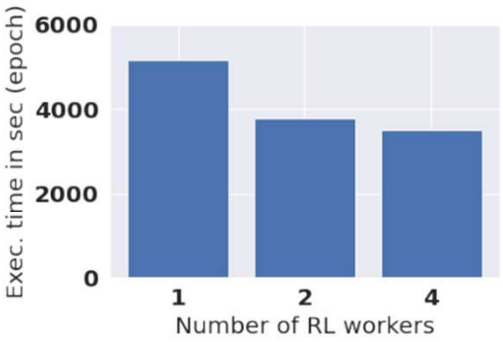
Experimental results and discussion

- Results and discussions:
  - Hyperparameter analysis: **Number of PBT workers.**
    - Tested with **4, 8**, and **16** PBT workers, for each of the CEC17 problems as **testing** functions.
    - The setups were compared with each other for all **29** problems in the benchmark.
    - For each comparison, we selected the *p*% best policies.
    - Measure of **success** of a setup **A**: number of comparisons where the setup A performed **significantly better** than other setup.

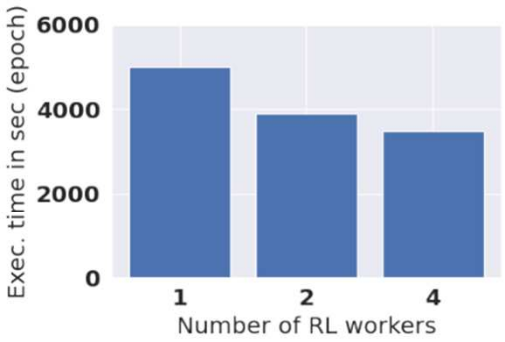
	Number PBT workers		
Quantile (p)	4	8	16
1.0	**34**	33	15
0.9	**37**	32	12
0.8	**39**	27	14
0.7	**41**	26	14
0.6	**42**	24	16
0.5	**45**	21	19
0.4	**45**	19	22
0.3	**43**	16	24
0.2	**39**	15	24
0.1	**38**	18	23
0.01	22	10	**25**
Best	3	5	**11**

Experimental results and discussion

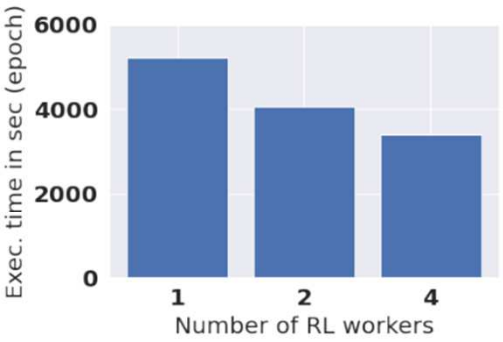
- Results and discussions:
  - Hyperparameter analysis
    - **Number of RL workers:** affects only the **efficiency**.
    - Average execution time in seconds of a training **epoch** across all training processes:



(a) Function 1



(b) Function 15



(c) Function 29

- Average execution time of the **validation** phase: 3000 seconds

Experimental results and discussion

- Results and discussions:
  - Hyperparameter analysis
    - **Perturbation interval**
    - Comparing different percentages of top policies:

	Perturbation interval		
Quantile (p)	**2**	**4**	**8**
1.0	**49**	34	1
0.9	**45**	36	2
0.8	37	**39**	5
0.7	36	**41**	4
0.6	29	**43**	10
0.5	25	**44**	12
0.4	28	**44**	12
0.3	19	**41**	19
0.2	17	**35**	20
0.1	16	**26**	24
0.01	15	16	**20**
Best	2	1	**3**

Experimental results and discussion

- Results and discussions:
  - Hyperparameter analysis
    - **Quantile fraction**: the lower, the more diverse the policy search.
    - Comparing different percentages of top policies:

	Quantile fraction		
Quantile (p	0.125	0.25	0.375
1.0	**46**	21	16
0.9	**46**	18	20
0.8	**47**	18	17
0.7	**49**	20	15
0.6	**50**	20	15
0.5	**51**	20	15
0.4	**52**	20	12
0.3	**48**	22	11
0.2	**48**	26	7
0.1	**48**	26	6
0.01	**27**	19	7
Best	**10**	3	1

Experimental results and discussion

- Results and discussions:
  - Hyperparameter analysis
    - **Resample probability**: the higher, the more diverse the policy search.
    - Comparing different percentages of top policies:

	Resample probability		
Quantile (p)	**0.25**	**0.5**	**0.75**
1.0	4	**42**	37
0.9	7	**42**	32
0.8	8	**43**	32
0.7	6	**43**	32
0.6	5	**45**	32
0.5	5	**44**	30
0.4	6	**42**	33
0.3	8	**36**	34
0.2	9	32	**40**
0.1	9	24	**42**
0.01	15	25	**28**
Best	**4**	**4**	1

Experimental results and discussion

- Results and discussions:
  - **Analysis of different budgets in the training and testing phases**
    - **Best** policy.
    - Can we **save** training time using **less** iterations per episode in the training phase?
    - HCLPSO + CEC17 benchmark
    - **100** iterations vs **300** iterations in the **training** phase.
      - Always **300** iterations in the **testing** phase.

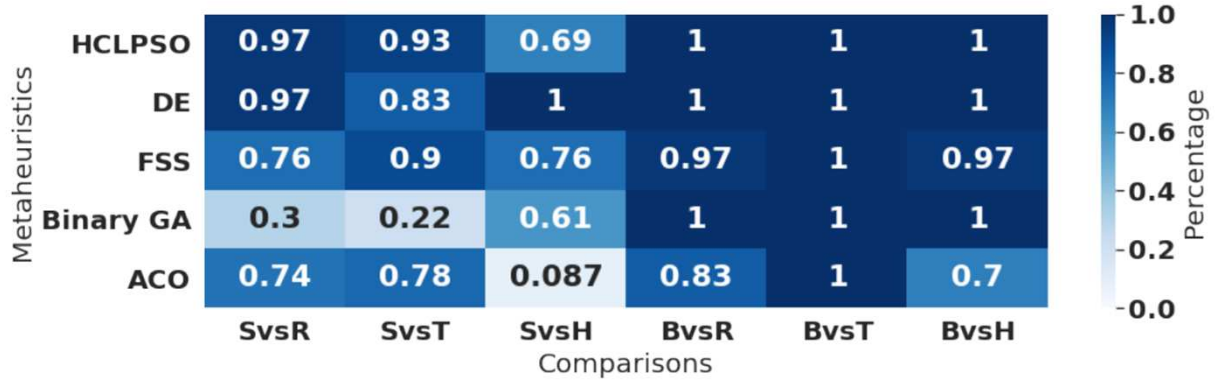
	Iterations	
Quantile (p)	100	300
1.0	4	**25**
0.9	5	**24**
0.8	3	**24**
0.7	3	**24**
0.6	4	**24**
0.5	4	**24**
0.4	5	**24**
0.3	6	**22**
0.2	9	**19**
0.1	11	**14**
0.01	5	**8**
Best	**1**	**1**

Experimental results and discussion

- Results and discussions:
  - **Generality assessment (issue #3)**
    - **133** different scenarios.
    - Compare **selected** and **best** policies with **random**, **tuned** static, and **human-designed** policies.

Experimental results and discussion

- Results and discussions:
  - **Generality assessment (issue #3)**



Better and p-value <= 0.05 or similar (p-value > 0.05)

Better and p-value <= 0.05

Experimental results and discussion

- Results and discussions:
  - **Generality assessment (issue #3)**
    - The **selected policies** are, on average, placed among the...
      - ...4.21% best policies for **HCLPSO**,
      - ...11.12% best policies for **DE**,
      - ...16.01% best policies for **binary GA**,
      - ...18.89% best policies for **FSS**,
      - ...and 34.46% best policies for **ACO**.
        - » ACO instances were **randomly generated** and, thus, are **highly uncorrelated**.

Conclusions

Conclusions

- This study proposed an **out-of-the-box** policy training method for **parameter control** of mono-objective EA and SI algorithms with **distributed Reinforcement Learning**, addressing the following issues identified in the literature.
  1. Propose a **scalable parameter control policy training** method that produces policies that can be used in an **unseen** problem.
  2. Propose a mechanism that diminishes the **difficulties** on the hyperparameter adjustment and reduces the **instability** of the learning process.
  3. Assess the **generality** of the proposed method in an experimental benchmark with **truly** varied scenarios.

Conclusions

- Contributions:
  1. Publication of the first **systematic literature review** on **out-of-the-box** parameter control for EA and SI.
  2. The proposed training process is clearly able to benefit from **parallel computing platforms** (**issue #1**).
  3. The **hyperparameter analysis** provided in this study revealed a few insights regarding the adjustment of the **PBT layer's hyperparameters**.
  4. The performance of the best policies are **less sensitive** to the technique's hyperparameters (**issue #2**).
  5. The proposed method presented a satisfactory **level of generality**, working well for the vast majority of the **varied testing scenarios**.

Conclusions

- Contributions:
  6. In the majority of the test cases, best policies found by our **AI** outperformed the **human-designed** and **tuned** policies.
  7. We have **advanced** the only study that had applied RL-algorithms with **continuous action space** to **out-of-the-box** parameter control for EA and SI so far: *SCHUCHARDT, J.; GOLKOV, V.; CREMERS, D. Learning to Evolve. arXiv: 1905.03389, 2019.*
- **Important**: The proposed method is still **very costly**. However, it is important to observe that once the policy is **trained** and selected to be used in a given unseen problem, the controller is used in the **production mode** (only feedforward in the ANNs-based method, such as TD3).

Conclusions

- Avenues of future research:
  - Choose a single training function according to the unseen problem at hand.
    - Dramatically reduces the burden of the testing phase, what allows sizeable speed-ups.
  - Include the population size in the set of controlled parameters.
  - Use RL algorithms that satisfactorily deals with sparse reward functions.
  - The proposed method should be applied to operator selection as well and could be extended to multiobjective optimization.

Conclusions

- Future of the field
  - **Hybridization** of RL and EA/SI is a quite promising field.
  - It has received an **increasing attention** from the community.
    - 1st Evolutionary Reinforcement Learning Workshop (GECCO'21).
  - Meta-learning for population-based algorithms.
    - Optimization algorithms for neural networks (**learn to learn**).
    - EA/SI **entirely** designed by AI.

Papers related to this study

1. Population Size Control for Efficiency and Efficacy Optimization in Population Based Metaheuristics
    - Authors: **Marcelo Gomes Pereira de Lacerda**, Hugo Amorim Neto, Teresa Bernarda Ludermir, Herbert Kuchen, and Fernando Buarque de Lima Neto.
    - Status: **Published** in 2018 IEEE Congress on Evolutionary Computation (CEC) (Qualis: **A1**).
2. On the Learning Properties of Dueling DDQN in Parameter Control for Evolutionary and Swarm-based Algorithms
    - Authors: **Marcelo Gomes Pereira de Lacerda**, Fernando Buarque de Lima Neto, Hugo Amorim Neto, Herbert Kuchen, Teresa Bernarda Ludermir.
    - Status: Published in 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI) (Qualis: **B4**).
3. A systematic literature review on general parameter control for evolutionary and swarm-based algorithms
    - Authors: **Marcelo Gomes Pereira de Lacerda**, Luis Filipe de Araújo Pessoa, Fernando Buarque de Lima Neto, Teresa Bernarda Ludermir, Herbert Kuchen.
    - Status: **Published** in Swarm and Evolutionary Computation (Qualis: **A1**).
4. Towards a Parameterless Out-of-the-box Population Size Control for Mono-Objective Metaheuristics
    - Authors: **Marcelo Gomes Pereira de Lacerda**, Hugo de Andrade Amorim Neto, Teresa Bernarda Ludermir, Herbert Kuchen, Fernando Buarque de Lima Neto.
    - Status: Submitted to a scientific journal.

Papers related to this study

5. A Distributed Training Process for an Out-of-the-box Parameter Controller for Metaheuristics
    - Authors: **Marcelo Gomes Pereira de Lacerda**, Fernando Buarque de Lima Neto, Teresa Bernarda Ludermir, Herbert Kuchen.
    - Status: Submitted to a scientific journal.
6. Distributed Reinforcement Learning for Out-of-the-box Parameter Control in Evolutionary and Swarm-based Algorithms
    - Authors: **Marcelo Gomes Pereira de Lacerda**, Fernando Buarque de Lima Neto, Teresa Bernarda Ludermir, Herbert Kuchen.
    - Status: Submitted to a scientific journal.

Out-of-the-box Parameter Control for Evolutionary and Swarm-based Algorithms with Distributed Reinforcement Learning

Ph.D. Thesis Defense

Ph.D. Candidate: Marcelo Gomes Pereira de Lacerda, M.Eng.
Supervisor: Prof. Teresa Bernarda Ludermir, Ph.D.
Co-Supervisor: Prof. Fernando Buarque de Lima Neto, Ph.D

March 2021