Bayesian Optimization for Developmental Robotics with Meta-Learning by Parameters Bounds Reduction

1st Maxime Petit
LIRIS, CNRS UMR 5205
Ecole Centrale de Lyon
Ecully, France
maxime.petit@ec-lyon.fr

2nd Emmanuel Dellandrea
LIRIS, CNRS UMR 5205
Ecole Centrale de Lyon
Ecully, France
emmanuel.dellandrea@ec-lyon.fr

3rd Liming Chen
LIRIS, CNRS UMR 5205
Ecole Centrale de Lyon
Ecully, France
liming.chen@ec-lyon.fr

arXiv:2007.15375v1 [cs.RO] 30 Jul 2020

Abstract—In robotics, methods and softwares usually require optimizations of hyperparameters in order to be efficient for specific tasks, for instance industrial bin-picking from homogeneous heaps of different objects. We present a developmental framework based on long-term memory and reasoning modules (Bayesian Optimisation, visual similarity and parameters bounds reduction) allowing a robot to use meta-learning mechanism increasing the efficiency of such continuous and constrained parameters optimizations. The new optimization, viewed as a learning for the robot, can take advantage of past experiences (stored in the *episodic* and *procedural* memories) to shrink the search space by using reduced parameters bounds computed from the best optimizations realized by the robot with similar tasks of the new one (*e.g.* bin-picking from an homogenous heap of a similar object, based on visual similarity of objects stored in the *semantic* memory). As example, we have confronted the system to the constrained optimizations of 9 continuous hyperparameters for a professional software (Kamido) in industrial robotic arm bin-picking tasks, a step that is needed each time to handle correctly new object. We used a simulator to create bin-picking tasks for 8 different objects (7 in simulation and one with real setup, without and with meta-learning with experiences coming from other similar objects) achieving goods results despite a very small optimization budget, with a better performance reached when meta-learning is used (84.3% vs 78.9% of success overall, with a small budget of 30 iterations for each optimization) for every object tested (p-value=0.036).

Index Terms—developmental robotics, long-term memory, meta learning, hyperparmeters automatic optimization, case-based reasoning

I. INTRODUCTION

In the field of robotics, many frameworks and algorithms require optional parameters settings in order to achieve strong performance (*e.g.* Deep Neural Networks [1], Reinforcement Learning [2]). Even if a human expert can manually optimized them, the task is tedious and error-prone, in addition to being costly in term of time and money when applied to the private industrial sector, in particular in situations where the hyperparameters have to be defined frequently (*e.g.* for each object to be manipulated or for each manipulation task). Optimization processes can be used to overcome these challenges on constrained numerical hyper-parameters search, such as Bayesian

Fig. 1. Real robotics setup with an industrial Fanuc robot for a grasping task from homogeneous highly cluttered heap of elbowed rubber tubes.

Optimization [3]–[5]. This method is especially suited where running the software (treated as black-box function) to be optimized will be expensive in time and will produce noisy score (the case for real robotics grasping applications). These methods are classically used before the deployment of the system *in-situ*, or launched manually when needed: they are separated from the autonomous "life" of the robot's experience (*i.e.* they are used offline). Therefore the optimizations are always starting from scratch (*i.e. cold-start*) because they are not taking advantage of the knowledge from previous experiences of the system (*i.e. warm-start* [6]).

Our contribution consists of an enhanced version of the work from Petit *et al.* [7]: a developmental cognitive architecture providing a robot with a long-term memory and reasoning modules. It is allowing the robot to store optimization runs for bin-picking tasks using a professional grasping software, and utilize such experiences to increase the performance of new optimizations. In their initial works, when confronted to a new object for the bin-picking for which the grasping software
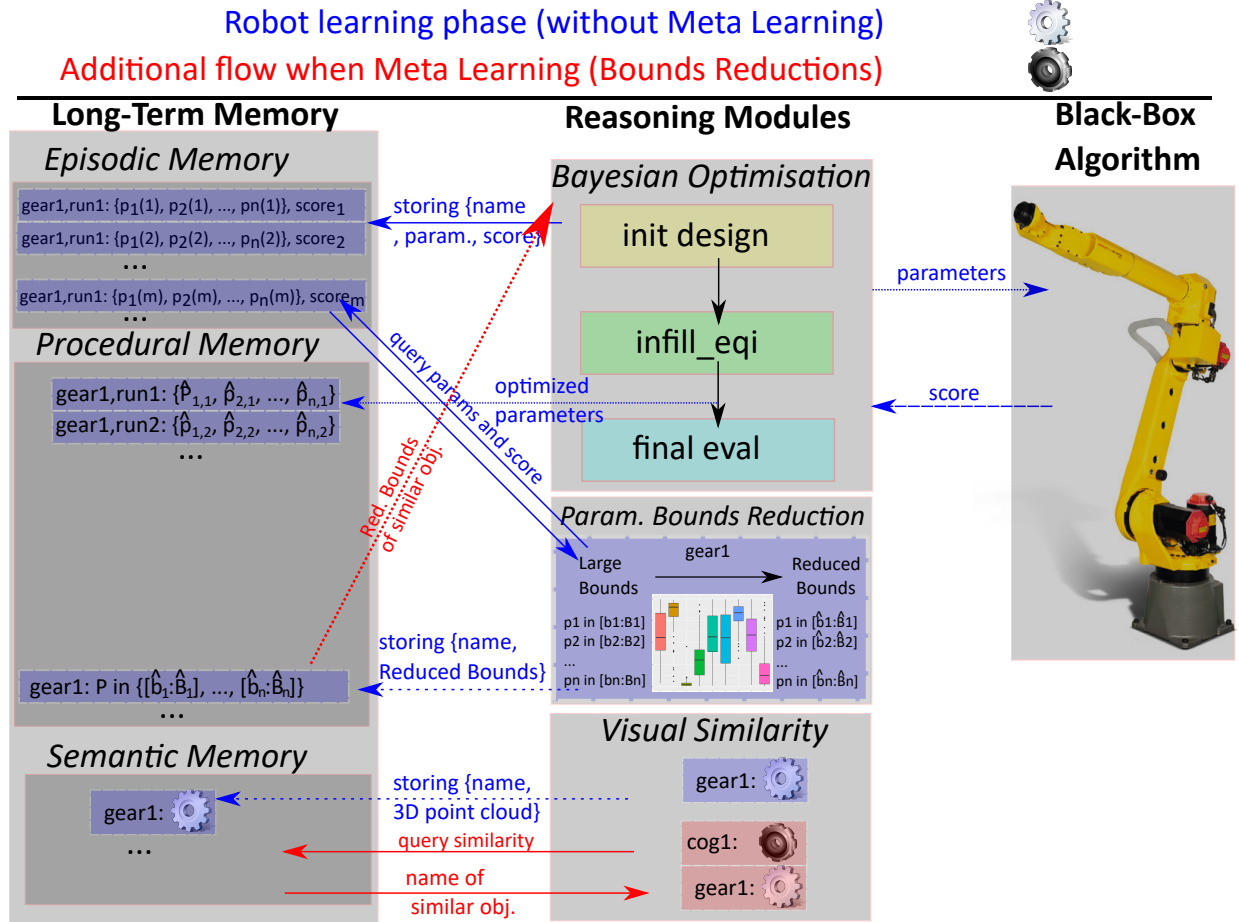
Fig. 2. Architecture of the extended cognitive developmental framework, based on Long-Term Memory (with episodic, procedural and semantic memories) and Reasoning Modules (Bayesian Optimisation, Visual Similarity and the new Parameters Bounds Reduction) allowing a robot to learn how to grasp objects. This learning consists of guiding an efficient continuous hyper-parameters constrained optimization of black-box algorithm controlling the robot. The blue arrows represent the data flows during a learning phase without meta-learning (*i.e.* without taking advantage of the Long-Term Memory, just storing the experiences). The red arrows shows the additional queries and exchanges of information during a learning phase with meta-learning, based on the visual similarity between objects the robot knows how to grasp and the new one.

parameters will have to be optimized, the robot is able to find a better solution faster with a transfer-learning strategy. This consists of extracting the best sets of parameters already optimized from a similar object and forcing the reasoning module to try it at the beginning of the optimization. Our contribution is the design of a meta-learning method for such optimization, in order to reduce the search space initially, thus avoiding unnecessary explorations in some areas. More specifically, we will use reduced parameters bounds that are extracted from the best previous optimization iterations of task or object that are similar to the new one, leading to a more efficient learning.

II. RELATED WORK

Bayesian Optimization (BO) is a common method in the robotic field for optimizing quickly and efficiently constrained numerical parameters [8]–[10]. In particular, Cully *et al* implemented an extended version allowing a robot to quickly adjust its parametric gait after been damaged [11] by taking advantages of previous simulated experiences with damaged

legs. The best walking strategies among them were stored in a 6-dimensional behavioural grid (discretized with 5 values per dimension representing the portion of time of each leg in contact with the floor). We take inspiration from this work, where the behavioural space will be represented by the similarity between objects the robot will have to learn to manipulate.

The meta-learning concept of this work, focusing on reducing the initial search space of constrained numerical parameters optimization is inspired by the work of Maesani *et al.* [12], [13] known as the Viability Evolution principle. It consists, during evolutionary algorithms, of eliminating beforehand newly evolved agents that are not satisfying a viability criteria, defined as bounds on constraints that are made more stringent over the generations. This is forcing the generated agents to evolve within a smaller and promising region at each step, increasing the efficiency of the overall algorithm. We follow here the same principle by reducing the hyperparameters bounds based on past similar experience

before the beginning of the optimization process, providing to it a smaller search space.

III. METHODOLOGY

The architecture of the cognitive robotics framework (see Fig. 2) is based upon the work of Petit et al. [7]. It consists of the construction and exploitation with different reasoning capacities of a Long-Term Memory storing information in 3 sub-memories as described by Tulving [14]: 1) the *episodic memory* storing data from personal experiences and events, then linked to specific place and time, 2) the *procedural memory* containing motor skills and action strategies learnt during the lifetime and 3) the *semantic memory* filled with facts and knowledge about the world. The developmental optimization with meta-learning will use this framework as followed: the Bayesian Optimization will provide all the data about its exploration and store them in the *episodic memory*, with the optimized set of parameters stored in the *procedural memory*. Parameters Bounds Reduction module will analyze the data for each task from the *episodic memory* in order to compute reduced parameters bounds still containing the best values for each parameters. A Visual Similarity module will be able to compare the similarity between different tasks (*e.g.* grasping an object O_1 and an object O_2) in order to have access to previous knowledge stored in the *procedural memory* and linked to a known similar task when confronted to a new one. This will allow the robot to use a smaller search optimization space when trying to learn how to achieve a task A by using the reduced parameters bounds computed from a similar and already explored and optimized task B.

A. Bayesian Optimisation module

We have chosen Bayesian Optimization as method for constrained optimization process of the robotic algorithm blackbox, implemented using the R package *mlrMBO* [15] with Gaussian Process as surrogate model. A BO run optimizes a number of parameters with iterations (*i.e.* trials) where the set of parameters is selected (and tested) differently depending on the current phase, out of 3, of the process:

- *"initial design"*: selecting points independently to draw a first estimation of the objective function.
- Bayesian search mechanism (*"infill eqi"*), balancing exploitation and exploration. It is done by extracting the next point from the acquisition function (constructed from the posterior distribution over the objective function) with a specific criteria. We have chosen to use the Expected Quantile Improvement (EQI) criteria from Pichney *et al.* [16] because the function to optimize is heterogeneously noisy. EQI is an extension of the Expected Improvement (EI) criteria where the improvement is measured in the model rather than on the noisy data, and so is actually designed to deal with such difficult functions.
- final evaluation (*"final eval"*), where the best predicted set of hyper-parameters (prediction of the surrogate, which reflects the mean and is less affected by the noise)

is used several times in order to provide an adequate performance estimation of the optimization.

B. Memory

Similarly to others implementations of a long-term memory system [17], [18], the experience and knowledge of the robot are stored in a PostgreSQL database. The *episodic* memory stores each experience of the robot, and consists for this work of the information available after each iteration i of the Bayesian Optimization's run r: the label of the task (*e.g.* the name of the object for which the robot has to optimize parameters in order to manipulate it), the set of m hyper-parameters tested $\{p_1(i), p_2(i), ..., p_m(i)\}$ and the corresponding score s_i obtained with such setup. The *semantic memory* is filled and accessed by the Visual Similarity module and contains the visual information about the objects that the robot used during its optimization runs, and are stored as point clouds. The *procedural memory* is composed by 2 types of data: 1) optimized sets of parameters of each run of each object are stored by the Bayesian Optimisation module, in order to be quickly loaded by the robot if needed, and 2) reduced parameters bounds for each object, corresponding of constrained boundaries for each parameters values obtained when looking at the parameters values distribution from the best iterations of a specific task/object. This information is pushed here by the Parameters Bounds Reduction module, that we will describe later.

C. Visual Similarity module

The Visual Similarity module is retrieving the most similar object from the *semantic* memory (*i.e.* CAD model of known object, meaning the robot has already optimized the corresponding parameters) where confronted to CAD models of a new objects to be optimized. It is based on an extension of the deep learning method for 3D classification and segmentation PointNet [19] which provides a numerical metrics for the similarity between 2 objects as the distance of the 1024 dimensions global features from the models. The most similar object corresponds to the minimal distance.

D. Meta Learning: Parameters Bounds Reductions

The Meta Learning aspect is realized with the use of reduced, more adequate, promising and efficient parameters bounds when launching the constrained optimization of a novel task (*i.e.* bin-picking a new object), using the reduced parameters bounds extracted from the experience of the robot with bin-picking a similar object to the new one. When looking at the distribution of the parameters values explored during the iterations that provided the best results, an efficient parameters bounds would provide roughly a uniform distribution of the parameters values among the best iteration, meaning that they are many parameters values within that provide good results. On the opposite, a very thin distribution means that a huge part of the search landscape for the parameters are sub-optimized and will cost optimization budget to be explored futilely. We want then to reduce the parameters bounds in order to force the
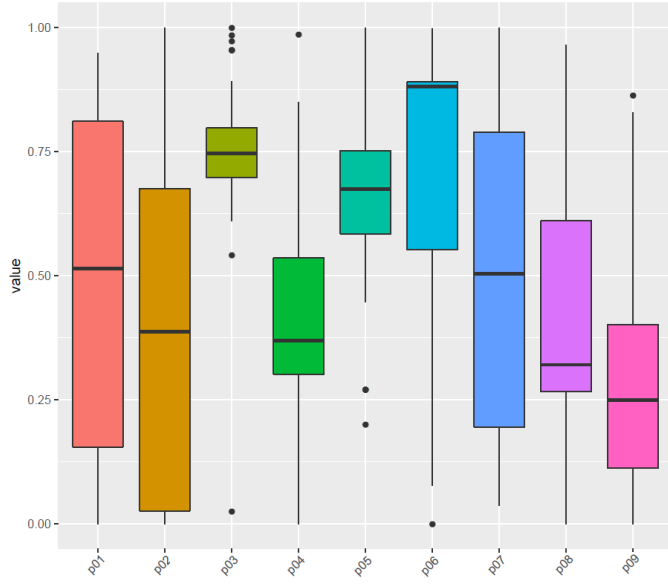
Fig. 3. Distribution of the scaled values of 9 parameters from the best 35% optimizations iterations of the object to be grasped called *m782*. Some parameters have a uniform [0:1] distribution (*e.g.* p1) but some do not and their median is either around 0.5 (*e.g.* p7), higher (*e.g.* p5) or smaller (*e.g.* p9). See Table I for the corresponding new reduced parameter bounds.

Algorithm 1 Algorithm for bounds Reduction

Input: All iterations of all runs for object O with scaled parameters values ($\in [0 : 1]$)

Output: New reduced bounds for object O

1: Select $I_n(O)$ the n% best iterations for O
2: **for** each parameters $p_j(O)$ **do**
3:    Compute p_{dm}, p-value of Dudewicz-van der Meulen test for uniformity for $p_j(O)$ values from $I_n(O)$
4:    **if** $(p_{dm} < \alpha_{dm})$ **then**
5:       Compute p_w, p-value of Wilcoxon test (H0: $\mu = 0.5$)

6:       **if** $(p_w < \alpha_w$ and median$(p_j(O)) > 0.5)$ **then**
7:          Increase lower bound for $p_j(O)$ to the 5% percentile of $p_i(O)$ values from $I_n(O)$
8:       **else if** $(p_w < \alpha_w$ median$(p_j(O)) < 0.5)$ **then**
9:          Reduce upper bound for $p_j(O)$ to the 95% percentile of $p_j(O)$ values from $I_n(O)$
10:      **else**
11:         Reduce upper & increase lower bounds for $p_i(O)$
12:      **end if**
13:   **end if**
14: **end for**
15: **return** Modified Parameters bounds

---

optimization process to focus on the more promising search space. We describe here how the module is able to reduced the parameters bounds from past optimization of an object O, summarized in Alg. 1 in order to increase the efficiency of future optimization runs for the same or similar object.

First, the module is checking the *episodic* memory of the robot to retrieve every results of past optimization iterations for

the object O, $I(O)$. Among them, we only keep the iterations that provided the best results, filtering to have the n% best remaining and obtain $I_n(O)$, a subset of $I(O)$. Then the module will analyze the distribution of every parameters p_j explored for the object O and scaled in [0:1], where an example of such distribution is shown in Fig. 3 under the form of boxplots. For each parameter, we check the uniformity of the distribution in [0:1] using the Dudewicz-van der Meulen test [20], an entropy-based test for uniformity over this specific distribution. If the p-value p_{dm} is below the alpha risk α_{dm}, we can reject the uniformity hypothesis for the current distribution: we can eliminate some range values for the parameter. However, it can goes several ways: we can lower the upper bounds, increasing the lower bounds, or doing both. This decision will be based on the result on a non-parametric (we cannot assume the normality of the distribution) one-sample Wilcoxon signed rank test against an expected median of $\mu = 0.5$ producing a p-value p_w and using another alpha risk α_w. If the $p_w < \alpha_w$ we can reject the hypothesis that the distribution is balanced and centered around 0.5. If that is the case and the distribution is not uniform, that means that both bounds can to be reduced (lowering the upper bounds and increasing the lower one). If not, that means the distribution is favoring one side (depending on the median value) and only the bounds from the opposite side will be more constrained: the lower bounds will be increased if the median is greater than 0.5, or the upper bounds will be smaller if the median is lower than 0.5. The bounds are modified to the xth percentile value of the parameters for the lower bounds and to the Xth percentile for the upper bounds, with $0 \leq x < X \leq 1$. Eventually, they are stored in the *procedural memory* and linked to their corresponding object, in order to be easily accessible in the future and used by future optimization process instead of the default and larger parameters bounds.

IV. Experiments

The experiment setup is similar to the describe in [7] allowing to compare some of their results with ours. We are indeed aiming at optimizing some parameters of a professional software called Kamido[1] (from Sileane) that we are treating as a black-box. The parameters are used by Kamido to analyze RGB-D images from a fixed camera on top of a bin and extract an appropriate grasping target for an industrial robotic arm with parallel-jaws gripper in a bin-picking task from an homogeneous heap (*i.e.* clutter composed by several instances of the same object).

We use real-time physics PyBullet simulations where objects are instantiated from Wavefront OBJ format on which we apply a volumetric hierarchical approximate convex decomposition [21]. The function to be optimized will be the percentage of success at bin-picking, where an iteration of the task consist of 15 attempts to grasp cluttered objects in the bin and to release the catch in a box. We also introduce a partial

---

[1]http://www.sileane.com/en/solution/gamme-kamido

New objects to be optimized

A C2 · C1 C2 · D D' · P1 P2 · hammer_j hammer_t · m784 m782 · smallCoke bathDetergent

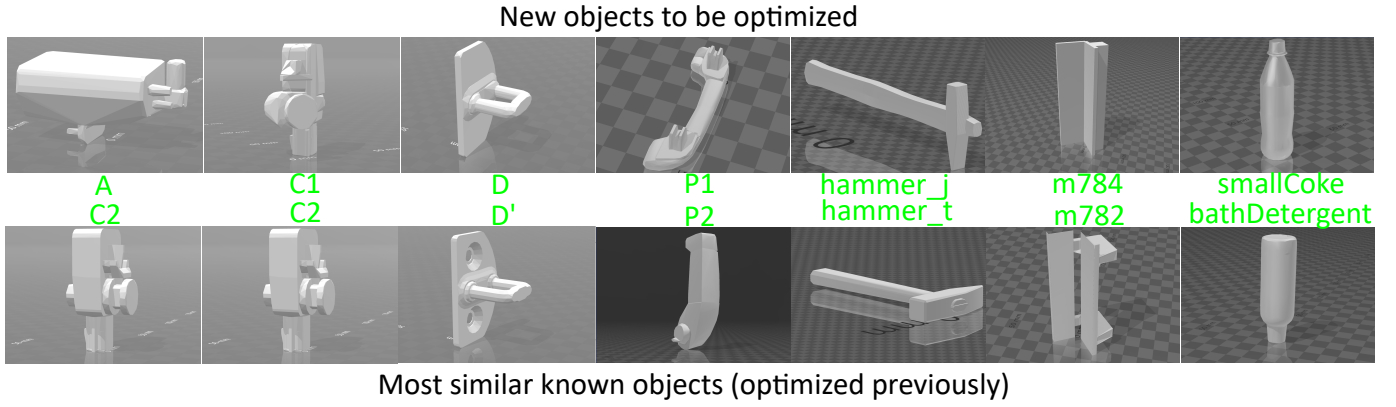## Most similar known objects (optimized previously)

Fig. 4. CAD models of the simulated objects

reward (0.5 instead of 1) when the robot is grasping an object but fails to drop it into the deposit box.

To be able to compare each learning run with the same learning condition, we authorize a finite budget for the BO process of 35 iterations, decomposed as follows: 10 for the *"initial design"*, 20 for the core BO process and 5 as repetitions of the optimized set of parameters in order to provide a more precise estimation of the performance. As opposed to the experiment done in [7], we decided to constrain more the learning setup, providing only 30 (10+20) iterations instead of 68 (18+50). Indeed, the learning curve seemed to flattened around this number of iterations in their work, so we wanted to compare the quality of the optimization at an earlier stage. For the bounds reduction algorithm, we use a selection of the best 35% iterations for each object thus allowing a good range of potential efficient set of parameters from a very noisy objective function, and alpha risk of 0.15 for both the Dudewicz-van der Meulen and Wilcoxon tests (*i.e.* $\alpha_{dm} = \alpha_w = 0.15$). The percentile used for the bounds reductions are x=0.05 and X=0.95 in order to discard any potential outliers that might otherwise forbid a strong reduction in boundaries.

The other aspect of the setup are unchanged. Indeed, during the initial design phase, the set of parameters are selected using a Maximin Latin Hypercube function [22] allowing a better exploration by maximizing the minimum distance between them. The kernel for the GP is the classic Matern 3/2 and the criteria for the bayesian search mechanism on the acquisition function is an EQI with a quantile level of $\beta = 0.65$. The infill criterion is optimized using a stochastic derivative-free numerical optimization algorithm known as the Covariance Matrix Adapting Evolutionary Strategy (CMA-ES) [23], [24] from the package *cmaes*.

For the experiments presented in this work, we used some objects from [7], namely the reference A, C1, C2, D and D' in order to compare the performance of the method with a smaller learning budget as explained earlier. We also introduce new objects, some from a CAD database of real industrial reference (P1 and P2), and some from other common databases, such as *hammer_t* and *hammer_j* from turbosquid, *m782* and *m784* from Princeton Shape Benchmark [25], and *bathDetergent* and

TABLE I
BOUNDS FOR EACH PARAMETER TO BE OPTIMIZED, WITH THE LARGER "DEFAULT" AND THE REDUCED BOUNDS OBTAINED FROM SEVERAL OBJECTS USING THE PARAMETERS BOUNDS REDUCTIONS MODULE.

Obj.	p1	p2	p3	p4	p5	p6	p7	p8	p9
Def.	-20:20	5:15	16:100	5:30	5:30	5:40	30:300	5:20	1:10
C2	-20:20	**8:15**	**46:92**	5:30	**13:30**	**24:37**	**100:220**	**5:15**	**3:9**
D'	**-18:10**	5:15	**49:99**	**8:23**	5:30	5:40	30:300	**8:20**	**2:8**
P_2	-20:20	**6:14**	**20:69**	5:30	5:30	**18:37**	**114:267**	**5:19**	1:10
ham_t	-20:20	5:15	**46:100**	**8:30**	5:30	**17:40**	30:300	5:20	1:10
m782	-20:20	5:15	**68:96**	**7:23**	**12:30**	**9:37**	30:300	**5:19**	**1:8**
bathDet.	**-15:9**	**9:15**	**69:100**	**10:30**	**18:30**	**27:40**	**30:276**	5:20	1:10

cokeSmallGrasp from KIT [26]. New objects are shown in Fig. 4, along the objects (C2, C2, D', P2, *hammer_t* and *m782*) that has been optimized previously by the robot and that is the most similar, using the Visual Similarity module.

The experiments will consist of the optimization process for 7 objects (A, C1, D, P1, *hammer_j*, *m784* and *cokeSmall*) taken from 4 different object databases) when the method has been applied 6 times independently (*i.e.* runs) with 2 conditions: one optimization without any prior knowledge use, and one using meta-learning. This last condition involves retrieving the most similar and already optimized object known by the robot when confronted to the optimization of a new unknown object. Then the robot extracts the reduced boundaries of the best set of parameters it already tried with the similar object (the best 35% set of parameters) using the appropriate reasoning module described earlier. It then constrains the parameters values with these new reduced bounds during the optimization process. The reduced parameters bounds of each object similar to the references are presented in Table I.

V. RESULTS

In this section, we present the results from the experiments, focusing first on the performance during the optimization process, at both *initial design* and *infill eqi criteria* phase, with the Fig. 5. We can see that using the meta-learning (*i.e.* using prior information about the performance of set of parameters from similar object to the new one) allows the optimization process to have a *warmstart* during the *initial design* phase with a mean performance of already more than 75% compared

to ~65% when the parameters bounds are not restricted. It means that the algorithm process is avoiding spending optimization budget to explore parameters values that are inside the default bounds, but outside the bounds of interests from similar object, thus exploring un-optimized parameters values. This leads to a search space with more promising areas densities that the Bayesian Optimization process is able to explore more efficiently during the *infill eqi criteria* phase.
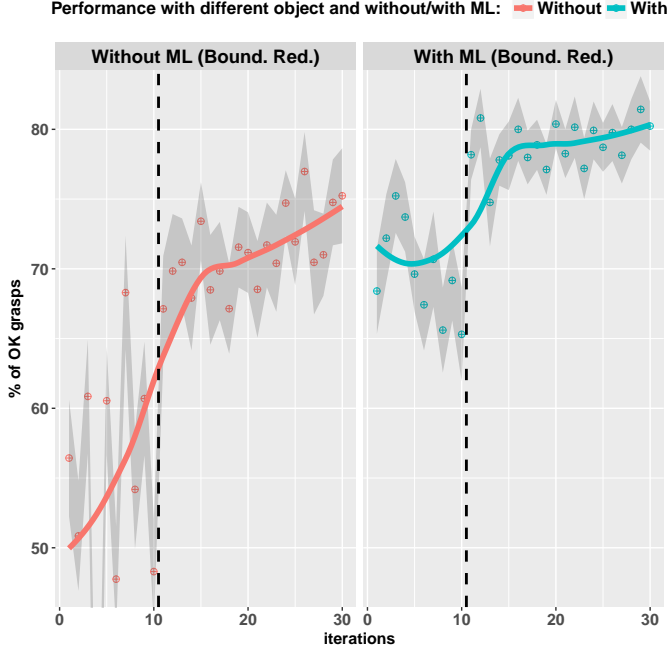

Fig. 5. Performance for each iteration (all objects) of the optimization runs, during the *initial design* (Iteration 1:10, Left of the vertical dotted line) and *infill eqi criteria* phase (Iteration 11-30, Right of the dotted line). Crossed circles are means among all runs at each iteration, while the grey area is the standard deviation. Curves corresponds to a smoothing among the points, using the non-parametric LOcally weighted redgrESSion (*i.e.* loess) method.

We then look at the final performances of every runs for every objects, split in two sets (without and with meta-learning) shown in Fig. 6. The mean performance overall increases from 78.9% (Q1: 73.1, median: 83.3, Q3:86.7) without the bounds reduction step to 84.3% (Q1: 78.1, median: 85, Q3:89.2) when the Bayesian Optimization is using meta-learning (Wilcoxon test). In addition, the worst performance after optimization among every runs and objects, even with a very short learning budget (30 iterations to optimize 9 continuous hyper-parameters), is at a decent 70.6% when using this meta-learning technique (vs 28.3% otherwise).

Detailed and numerical results of the experiments, split among all objects, are shown in Table II. First, we can compare the performance of the optimization method for object A, C1 and D at an earlier stage (after 30 learning iteration instead of 68) than the experiments from [7]. We indeed achieved similar performance for these objects under this harsher experiment design but with meta-learning, with respectively a mean success among all runs of 75.9%, 79.4%
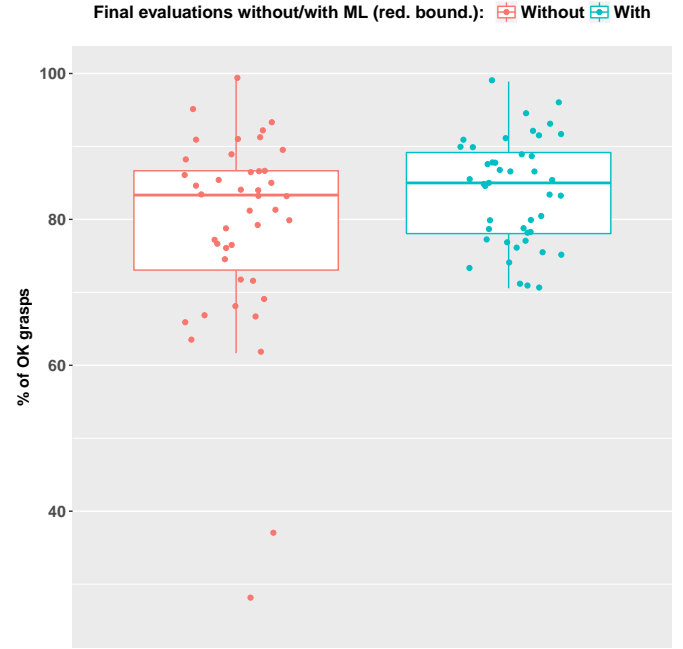

Fig. 6. Boxplot of the final performance after Bayesian Optimization on all objects for all runs, without and with meta-learning (Parameters Bounds Reduction applied to new objects from the bounds of a similar optimized object). Each dot is the mean final performance after an optimization run.

and 89.4% (30 iterations learning) vs 76.1%, 81.3% and 87.3% (68 iterations learning).

Looking at every object's performance, shown also in a paired graph from Fig. 7 , We can also clearly see the benefit of using the meta-learning method during the optimization process, with a better mean performance for every object among all the runs, leading to a significantly better score (paired sampled Wilcoxon test p-value=0.031). Table II also shows that worst performance is anyhow always better (at least > 70.6%) when using the meta-learning, providing a higher minimum expected performance (paired sampled Wilcoxon test p-value=0.031). Overall, it seems that the robot is benefiting more from the meta-learning when the task is more difficult (*i.e.* when percentage of success is overall lower) like with objects A and D, with a lower success score with BO only of respectively 68.4% and 65.1%) and the constrained search space allows the Bayesian Optimization to be more efficient and find promising parameters sooner, and for each run. However, the Bayesian Optimisation can still be efficient even without meta-learning as seen from the performance of the best runs, however the optimization are less reliable: most runs will not be as efficient as with meta-learning.

We have also implemented our architecture on a real robotic arm Fanuc, however the specific version of the robot (M20iA/12L vs M10iA12), the end-effector parallel-jaws gripper and the environmental setup (See Fig. 1) is different than the one used in [7], so direct comparison is not possible. In addition, because we used non-deformable object in simulation, we wanted to try with a real soft-

TABLE II
Optimization Results with/without Meta Learning -
Comparison with [7] using Budget of 68 iterations vs 30 here
and Transfer Learning instead of Meta Learning

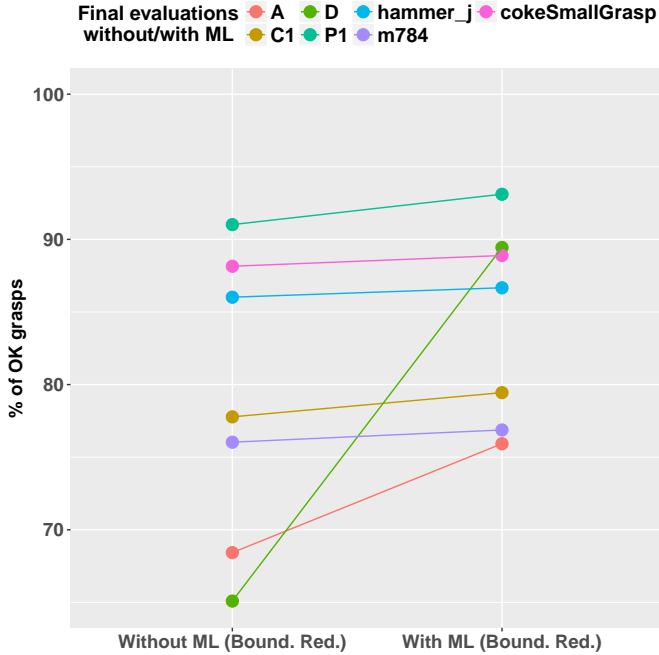| Reference	Budget	% succes all run mean±sd, median	% succes (worst run)	% succes (best run)
A [7]	68	65.47±27.3, 73.3	-	78.9
A	30	68.4±7.09, 66.4	61.7	81.1
A_ML_C2	30	75.9±2.37, 75.8	73.3	80.0
A_TL_C2 [7]	68	76.1±10.19, 76.7	-	82.8
C1 [7]	68	78.95±10.87, 80	-	83.9
C1	30	77.6±6.00, 77.5	68.3	85.0
C1_ML_C2	30	79.4±5.44, 79.4	70.6	85.0
C1_TL_C2 [7]	68	81.3±11.04, 80	-	82.5
D [7]	68	86.9±9.45, 86.67	-	91.1
D	30	65.1±25.7, 76.4	28.3	88.3
D_ML_D'	30	89.4±6.78, 90	78.9	96.1
D_TL_D' [7]	68	87.3±7.44, 86.7	-	90.6
P1	30	91.0±6.06, 91.4	83.3	99.4
P1_ML_P2	30	93.1±3.25, 91.7	91.1	98.9
ham_j	30	86.0±4.8, 84.7	80.0	92.2
ham_j_ML_ham_t	30	86.7±2.06, 86.7	83.3	90.0
m784	30	76.0±6.65, 76.7	66.7	86.7
m784_ML_m782	30	76.9±4.27, 77.8	71.1	83.3
coke	30	88.1±2.69, 87.8	84.4	91.1
coke_ML_detergent	30	88.9±3.06, 88.9	85.6	93.3

Fig. 7. Final mean performance of all runs, grouped by objects and paired on both conditions: without meta-learning and with meta-learning. This shows the systematic gain of performance when using meta-learning strategy, with a greater benefit where the initial performance was lower (object D and A)

body object in order to check if the method can obtain good results with such physical property. Therefore, we created an homogenous heap of highly cluttered elbowed rubber tube pieces as a test. With the 30 iterations budget runs, we have observed again a benefit of the meta-learning feature, with

an increase from 75.6% of mean performance with the real robot (sd=5.46, min=70.6, max=82.8) without meta-learning, to 84.6% (sd=2.5, min=82.2, max=87.2) with meta-learning.

VI. Conclusion and Future Work

This work explored how a robot can take advantage of its experience and long-term memory in order to utilize a meta-learning method and enhance the results of Bayesian Optimization algorithm for tuning constrained and continuous hyper-parameters, in bin-picking objects type of tasks (6 different objects extracted from 3 different shape objects database). With a very small fixed optimization budget of 30 trials, we are able to optimize 9 continuous parameters of an industrial grasping algorithm and achieve good performance, even with a very noisy evaluation function as encountered during this task. The meta-learning method, based on the reduction of the search space using reduced parameters bounds from the best iterations of object similar to the new one, guarantees overall a faster and better optimization with a mean grasping success of 84.3% vs 78.9% without meta-learning. Moreover, the increase in the mean expected performance from the optimization with meta-learning is consistent for every object tested, simulated or real (75.9% vs 68.4%, 79.4% vs 77.6%, 89.4% vs 65.1%, 93.1% vs 91.0%, 86.7% vs 86.0%, 76.9% vs 76.0%, 88.9% vs 88.1%, and 84.6% vs 75.6%), and is stronger for object presenting a higher challenge. When considering only the best run for each object among the 6, the optimization with meta-learning reaches 80.0%, 85.0%, 96.1%, 98.9%, 90.0% and 83.3% and 93.3% for respectively object A, C1, D, P1, *hammer_j*, *m784* and *cokeSmallGrasp*, which represents a mean score of 89.5%.

One of the assumption in this work was that the default parameters bounds where large enough to include optimized values within the range, that is why the Parameters Bounds module has been designed to only reduced them. However, future work will investigate the possibility of the parameters bounds to also be extended, which can be useful in particular when the manually defined default bounds are too constrained for a specific task.

We aim also to use this developmental learning framework from simulation into a transfer learning setup, where the reduced parameters bounds and the optimized parameters of a simulated object O will be used when optimizing the same object O but with a real robot, as explored for grasping problems recently [27]. The robot will use its simulated experiences in order to warm-start and simplify the optimization of the bin-picking of the same object when confronted in reality. The use of the simulation applied to transfer learning has the benefit of allowing the robot to always train and learn "mentally" (*i.e.* when one computer is available, and can even "duplicate" itself and run multiple simulation from several computers) even if the physical robot is already used or is costly to run, which is the case usually for industrial robots *in-situ*.

Eventually, this work can be extended toward the developmental embodied aspect of the robotics field, when reduced

parameters bounds might potentially be linked to embodied symbols or concept emergence [28] related to physical properties of the manipulated objects. A possible method to investigate such properties would be to find co-occurrences between sub-set of reduced parameters bounds and human labels or description of the object (*e.g.* "flat", "heavy") or of the manner the task has been achieved (*e.g.* "fast"), in a similar way that was done to discover pronouns [29] or body-parts and basic motor skills [30]. This would allow in return a possible human guidance in an intuitive manner to the robot by constraining the search space based on the label provided by the human operator.

REFERENCES

[1] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[2] T. Rückstiess, F. Sehnke, T. Schaul, D. Wierstra, Y. Sun, and J. Schmidhuber, "Exploring parameter space in reinforcement learning," *Paladyn*, vol. 1, no. 1, pp. 14–24, 2010.

[3] J. Mockus, "The bayesian approach to local optimization," in *Bayesian Approach to Global Optimization*. Springer, 1989, pp. 125–156.

[4] ——, "Application of bayesian approach to numerical methods of global and stochastic optimization," *Journal of Global Optimization*, vol. 4, no. 4, pp. 347–365, 1994.

[5] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[6] D. Yogatama and G. Mann, "Efficient transfer learning method for automatic hyperparameter tuning," in *Artificial Intelligence and Statistics*, 2014, pp. 1077–1085.

[7] M. Petit, A. Depierre, X. Wang, E. Dellandréa, and L. Chen, "Developmental bayesian optimization of black-box with visual similarity-based transfer learning," in *The 9th Joint IEEE Int. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-Epirob)*, 2018.

[8] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans, "Automatic gait optimization with gaussian process regression." in *IJCAI*, vol. 7, 2007, pp. 944–949.

[9] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1-2, pp. 5–23, 2016.

[10] B. Yang, G. Wang, R. Calandra, D. Contreras, S. Levine, and K. Pister, "Learning flexible and reusable locomotion primitives for a microrobot," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1904–1911, 2018.

[11] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, p. 503, 2015.

[12] A. Maesani and D. Floreano, "Viability principles for constrained optimization using a (1+ 1)-cma-es," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2014, pp. 272–281.

[13] A. Maesani, G. Iacca, and D. Floreano, "Memetic viability evolution for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 125–144, 2015.

[14] E. Tulving, "Memory and consciousness." *Canadian Psychology/Psychologie canadienne*, vol. 26, no. 1, p. 1, 1985.

[15] B. Bischl, J. Richter, J. Bossek, D. Horn, J. Thomas, and M. Lang, "mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions." [Online]. Available: http://arxiv.org/abs/1703.03373

[16] V. Picheny, D. Ginsbourger, Y. Richet, V. Picheny, D. Ginsbourger, and Y. Richet, "Optimization of Noisy Computer Experiments with Tunable Precision," *Technometrics*, vol. 55, no. 1, pp. 2–13, 2013.

[17] G. Pointeau, M. Petit, and P. F. Dominey, "Successive developmental levels of autobiographical memory for learning through social interaction," *IEEE Transactions on Autonomous Mental Development*, vol. 6, no. 3, pp. 200–212, 2014.

[18] M. Petit, T. Fischer, and Y. Demiris, "Lifelong augmentation of multimodal streaming autobiographical memories," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, no. 3, pp. 201–213, 2016.

[19] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

[20] E. J. Dudewicz and E. C. Van Der Meulen, "Entropy-based tests of uniformity," *Journal of the American Statistical Association*, vol. 76, no. 376, pp. 967–974, 1981.

[21] K. Mamou, "Volumetric hierarchical approximate convex decomposition," in *Game Engine Gems 3*, E. Lengyel, Ed. A K Peters, 2016, pp. 141–158.

[22] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[23] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. IEEE, 1996, pp. 312–317.

[24] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.

[25] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Proceedings Shape Modeling Applications, 2004.* IEEE, 2004, pp. 167–178.

[26] A. Kasper, Z. Xue, and R. Dillmann, "The kit object models database: An object model database for object recognition, localization and manipulation in service robotics," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 927–934, 2012.

[27] M. Breyer, F. Furrer, T. Novkovic, R. Siegwart, and J. Nieto, "Flexible robotic grasping with sim-to-real transfer based reinforcement learning," *arXiv preprint arXiv:1803.04996*, 2018.

[28] T. Taniguchi, T. Nagai, T. Nakamura, N. Iwahashi, T. Ogata, and H. Asoh, "Symbol emergence in robotics: a survey," *Advanced Robotics*, vol. 30, no. 11-12, pp. 706–728, 2016.

[29] G. Pointeau, M. Petit, G. Gibert, and P. F. Dominey, "Emergence of the use of pronouns and names in triadic human-robot spoken interaction," in *Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on*. IEEE, 2014, pp. 146–152.

[30] M. Petit and Y. Demiris, "Hierarchical action learning by instruction through interactive grounding of body parts and proto-actions," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3375–3382.