Leveraging Multi-Level Modelling to Automatically Design Behavioral Arbitrators in Robotic Controllers

Cyrill Baumann, Hugo Birch, Alcherio Martinoli

***Abstract*— Automatic control design for robotic systems is becoming more and more popular. However, this usually involves a significant computational cost, due to the expensive and noisy evaluation of candidate solutions through high-fidelity simulation or even real hardware. This work aims at reducing the computational cost of automatic design of behavioral arbitrators through the introduction of a two-step approach. In the first step, the structure of the finite state machine governing the behavioral arbitrator is optimized. To this purpose, a more abstracted model of the robotic system is leveraged in order to significantly reduce the computational cost. In the second step, the close-to-hardware, behavioral parameters are fine-tuned using a high-fidelity model. We show that, for a scenario involving a single robot and multiple tasks to be solved sequentially, using the proposed method results in a significant decrease of the computational cost while reaching the same controller performance both in simulation and reality.**

I. Introduction

Effectively designing and optimizing control algorithms for robotic systems plays a crucial role in enabling them to accomplish increasingly complex tasks. While many robotic systems are still designed manually, automatic control design is gaining popularity [1], [2]. Today, Artificial Neural Networks (ANN) in combination with Reinforcement Learning (RL) [3], evolutionary algorithms [2], or other meta-heuristic optimizers such as Particle Swarm Optimization [4], are highly popular and used for a wide variety of tasks. However, not only is the learning of ANNs non-trivial, but ANNs also have significant downsides. Notably, they cannot be easily read, understood, verified, or analyzed for formal properties, though improvements have been made in this direction [5]. Another unsolved challenge lies in the crossing of the reality gap, that is the transition from simulation to reality [6].

An alternative way of controlling robots is through Finite State Machines (FSM), which are fully readable and verifiable. The method that uses FSMs to switch between continuous-time states is formally known as hybrid automata, first proposed in [7]. In [8] [9], a FSM-generating framework has been introduced for swarm robotic controllers. It translates a FSM into an optimization problem whose associated cost can then be minimized. In [10], a similar framework is proposed, which evolves a finite-state controller based on basic behaviors. A key advantage of these frameworks lies in their use of (calibrated) basic behaviors. While limiting partially the exploration space of the machine-learning algorithm applied, it drastically reduces the reality gap [8] [10].

However, while these frameworks are shown to produce competitive solutions, their computational cost is high. Indeed in [8] but also [11], the best performances were obtained using a design budget of 200'000 evaluations of candidate solutions using a high-fidelity simulator. Assuming an evaluation takes 1 s, each design would take more than 2 days. There exist different techniques for reducing the computational cost of evaluations. In [12], for example, the evaluation length is reduced, maintaining only the minimal number of learning opportunities necessary. However, in our opinion, the most promising method to overcome this shortcoming lies in further abstraction of the models leveraged for learning.

There exists numerous single robot modeling techniques, often based on dynamic equations. However, we are mainly interested in modeling techniques that support both environmental interactions as well as interactions among multiple robots. Note that while this work concentrates on a single robot scenario to ease the demonstration of our approach, we designed our method with multi-robot scenarios in mind. Multi-Robot Systems (MRS) are not only more versatile and able to solve more complex scenarios than individual robots, they are also more robust to hardware failures thanks to mutual redundancy.

There exist only few such relatively general approaches. In [13] the authors use Probabilistic FSMs (PFSMs) with transition probabilities calibrated on physical experiments in order to faithfully represent a multi-robot system in time-discrete models. They also introduced a multi-level hierarchical representation of distributed robotic systems, involving submicroscopic, microscopic, and macroscopic models: at the submicroscopic level, details of the robotic nodes are faithfully represented in a high-fidelity robotic simulator. At the microscopic level, details of individual robotic nodes are simplified and aggregated but each physical robot in the system is represented separately in the model. At the macroscopic level, the whole group of robots is represented directly, typically using differential equations.In this framework, called Multi-Level Modeling, model structure and parameters are generated manually in a bottom-up fashion, from the physical reality to the highest abstraction level (i.e. the macroscopic representation). Another, similar framework is proposed in [14], which was then extended with formal model checking in [15]. In [16], constraint logic programming was introduced to reduce the model size. A convenient

The authors are with the Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
firstname.lastname@epfl.ch.
This work is financially supported by Mitsubishi Electric Corporation, Japan.

summary of the overall framework with examples can be found in [17].

In both [18] and [19], the authors use a multi-level modelling approach to guide manual control design. There have also been numerous contributions where modelling is used to speed-up the optimization processes of robotic controller parameters, including, for example, the works in [20] and [21]. However, to the best of our knowledge, leveraging multi-level modelling has never been applied to the optimization problem involved with the generation of both structure and parameters of a FSM.

Building upon the improved FSM-generation framework of [8] and subsequent works, and leveraging the modelling framework of [13], we combine two different modeling levels, microscopic and submicroscopic, of the very same robotic system in order to significantly reduce the computational cost required for generating the FSM-based behavioral arbitrator of the robot controller. This is especially interesting as the framework in [8] is based on basic behaviors which are agnostic of the targeted application. Therefore, once a basic behavior is adequately represented in the modelling framework, its model can be reused in future applications.

An additional contribution of this work is the extension of the probabilistic framework of [13] for a case where the controller structure cannot be directly leveraged as blueprint for the structure of the model. We further introduce the use of a Mixed-Discrete Particle Swarm Optimization (MDPSO) [22] in combination with Optimal Computing Budget Allocation (OCBA) [23] as alternative optimization algorithm for optimizing FSMs.

In the following, we first introduce the problem of automatically generating FSM-based behavioral arbitrators and define a concrete application scenario used throughout the paper as illustration. Then, we describe the methods used, including the leveraged models. Finally, we present and discuss our results on a concrete application, ending with some conclusive remarks.

II. Problem Statement and Scenario

The challenge tackled in this work is to automatically synthesize FSM-based behavioral arbitrators for a given scenario which involves solving a sequential single-robot task. The robot is placed in an arena populated with obstacles as illustrated in Figure 1. The robot is then tasked to find any of the two black floor patches present in the arena, before going to the white area in the top right corner. A light source is installed in this corner, allowing the robot to efficiently navigate to the white area using a light-following behavior.

Accordingly, the scenario-dependent cost function σ can be defined as:

$$\sigma = \frac{1}{T_E}\sum_{k=1}^{T_E}(1-\sigma_{mv}(k)\cdot(1-\sigma_{oa}(k)))\cdot\sigma_F(k) \qquad (1)$$

with T_E the experiment run-time,

$$\sigma_{mv,k} = \frac{||pos_k - pos_{k-1}||}{V_{max} * dT}$$

the travelled distance at t=k,

$$\sigma_{oa,k} = \frac{min(detected\_obstacle\_distances)}{max\_sensor\_range}$$

the distance to the closest obstacle at t=k. Both movement (mv) and obstacle avoidance (oa) metrics are adapted from [24] and are normalized between 0 and 1. σ_F is defined as:

$$\sigma_F(k) = \begin{cases} 0 & \text{on white floor at } t = k \text{ and on black floor at } t < k \\ 0.5 & \text{if on black floor at t} \leq k \\ 1 & \text{otherwise} \end{cases}$$

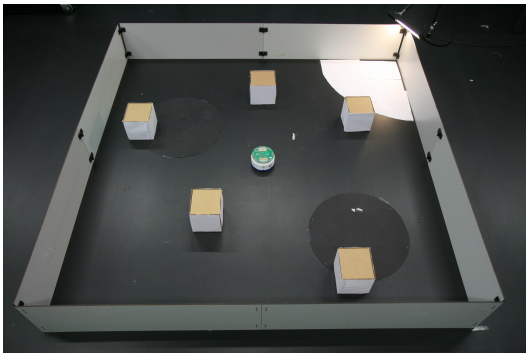


Fig. 1: The real 2 x 2 m arena including five obstacles and the Khepera IV robot in its initial position, as well as both black and white floor patches.

III. Methods

In order to translate the synthesis of a FSM-based behavioral arbitrator into an optimization problem we use the FSM-encoding grammar described in Section III-A.

Using a suitable optimization algorithm (c.f., Section III-C), we then aim at minimizing a scenario-dependent cost function σ, which is evaluated leveraging two different modeling levels, microscopic and submicroscopic. As elaborated in Section III-B, we consider three different approaches in this work, including a two-step approach, which leverages both modeling levels, microscopic in a first step, and submicroscopic in a second step, in order to reduce the overall computational cost of the optimization procedure.

As this work focuses on illustrating the methodological procedures, it uses a specific, relatively simple scenario presented in Section II. The implementation of the scenario at submicroscopic and microscopic level, according to the modelling framework introduced in [13], is then described in Sections III-D.1 and III-D.2 respectively.

A. FSM-Encoding Grammar

A generic FSM $\mathcal{F}$, with N_s states and $N_D = N_S(N_S-1)$ possible state transitions, can be encoded as a vector of both discrete and continuous numbers as follows:

$$\mathcal{F} = [[B]\ [D]\ [C]\ [A_B]\ [A_C]] \qquad (2)$$

where:

- B consists of $B_1 ... B_{N_S} \in \mathcal{B}$, corresponding to the selected behaviors for the N_S behavior slots, where $\mathcal{B}$ is the set of available behaviors reported in Table I.

- D consists of $D_1...D_{N_D} \in \{1, .., N_S\}$, corresponding to the destinations or target behavioral slots of every transition.
- C consists of $C_1...C_{N_C \cdot N_D} \in \mathcal{C}$ corresponding to the conditions for each transition to happen, where $\mathcal{C}$ is the set of available transition conditions and N_C to the number of conditions allowed per transition.
- A_B consists of $A_1...A_{N_{BP} \cdot N_S} \in [0, 1]$, corresponding to the parameters of the behaviors, where N_{BP} is the number of parameters allowed per behavior.
- A_C consists of $A_1...A_{N_{CP} \cdot N_C \cdot N_D} \in [0, 1]$, corresponding to the parameters of the conditions, where N_{CP} is the number of parameters allowed per condition.

As the length of vector $\mathcal{F}$ corresponds to the length of the cardinality of the sets above, the total number of variables for a FSM can thus be expressed as $N_V = N_S(N_S + N_{BP} + N_C(N_S - 1)(1 + N_{CP}))$. The numerical values used in this work are $N_C = 2$, $N_{BP} = 2$, $N_{CP} = 1$ and $N_S = 2$, resulting in a 16-dimensional mixed-integer vector $\mathcal{F}$, as illustrated in Figure 2.

The behaviors and conditions used in this work are summarized in Tables I and II. Figure 2 gives an example of a FSM and its corresponding 16 dimensional vector. The translation from a FSM to a corresponding vector or vice-versa can be achieved using Equation 2 as well as Tables I and II. It is worth noting that transitions onto itself, as well as multiple same states or conditions with the same or different arguments, are possible.

The optimization problem can be defined as

$$\min_{\mathcal{F}} \sigma(\mathcal{S}(\mathcal{F})) \tag{3}$$

where $\mathcal{S}$ corresponds to the resulting robot behavior in the microscopic or submicroscopic model, respectively, for a given $\mathcal{F}$ and σ is the cost function, depending on the targeted scenario. Note that due to the stochastic nature of robotic applications, a precise mathematical formulation of $\mathcal{S}$ is usually impossible, justifying the use of appropriate simulation tools to study it.

TABLE I: Set $\mathcal{B}$ of predefined basic behaviors.

Behavior-ID	Behavior	Parameter 1	Parameter 2
1	Braitenberg (BB)	agressivity	distance threshold
2	Light Following (LF)	agressivity	unused
3	Stop (SP)	unused	unused

TABLE II: Set $\mathcal{C}$ of predefined transition condition

Condition-ID	returns true at time T if	Parameter
0	always true	unused
1	above grey floor at T	sensor treshold
2	above white floor at T	sensor treshold
3	obstacles in range at T	sensor treshold
4	no obstacles in range at T	sensor treshold
5	above black floor at T	sensor treshold
6	above black floor at any $t \in [0...T]$	sensor treshold

B. FSM-Synthesizing Approaches

In the framework of simulation-based optimization processes, the computationally most expensive step is typically concerned with the evaluation of candidate solutions. Therefore, any technique that allows for the reduction of wall-clock evaluation time has a high impact on the overall computational cost, thus the wall-clock time is needed to complete the optimization process. As mentioned above, multiple techniques to this purpose exist. In this paper, we focus on the exploitation of more abstracted and thus computationally cheaper models for assessing the performance of the candidate solutions, before finalizing the optimization with more faithful and therefore expensive models. In particular, by using a microscopic model characterized by a reduced number of states and capturing probabilistic state transitions, a decrease of computational cost of up to four orders of magnitude in comparison to real robot experiments is possible for evaluating candidate solutions, as reported in [13]. In contrast, a submicroscopic model implemented in a high-fidelity simulator usually only achieves speed-ups of one or two orders of magnitude. However, due to the abstraction and aggregations used in the creation of the microscopic model, including a partial removal of spatiality, not all components of $\mathcal{F}$ can be explicitly represented. Notably the parameters of the individual behaviors, which are often directly linked to sensor values, are not explicitly representable within a microscopic model without inducing a significant increase in both model complexity and computational cost.

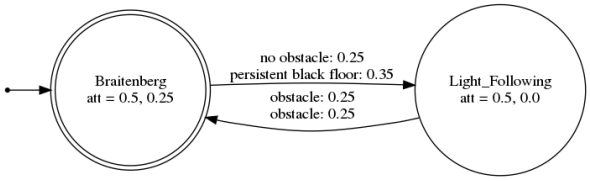


Fig. 2: Example of a FSM with $N_S = 2$, $N_C = 2$, $N_{BP} = 2$ and $N_{CP} = 1$, corresponding to the manual solution. The initial state is displayed by a double circle, other states by a single circle. State transitions are indicated by arrows. The FSM displayed here is encoded through the mixed-integer vector given by: [1 2 1 0 4 6 3 3 0.5 0.25 0.5 0.0 0.25 0.35 0.25 0.25]

We therefore consider three different FSM-synthesizing approaches for this task, leveraging different modeling levels:

- **A1**: Optimization of the full, 16-dimensional mixed-integer vector using the submicroscopic model.
- **A2**: Optimization of the 8-dimensional, discrete part of the FSM-descriptor vector using the submicroscopic model, using manually defined default values for the continuous parameters, followed by the optimization of the 8-dimensional, continuous part of the FSM-descriptor vector using the submicroscopic model, with the previously optimized discrete vector values frozen.
- **A3**: The two-step approach proposed in this work. First, optimization of the 8-dimensional, discrete part of the FSM-descriptor vector using the microscopic model. Then, optimization of the 8-dimensional, continuous part of the FSM-descriptor vector using the submicroscopic model, with the previously optimized discrete

vector values frozen.

C. Optimization Algorithm

The optimization algorithm used in this work is the Mixed-Discrete Particle Swarm Optimization (MDPSO) algorithm presented in [22]. In order to handle the noise inherent to robotic scenarios better, this algorithm has been enhanced with an Optimal Computing Budget Allocation (OCBA) scheme [23], using the centralized version introduced in [25]. It is worth noting that MDPSO degenerates to a regular PSO if no discrete variables are present. The number of MDPSO particles used, N_P, follows the rule of thumb of one particle per problem dimension, which is increased to two particles if discrete states are present. The maximum evaluation budget per approach in both modeling levels, N_E, is summarized in Table III. Note that higher evaluation budgets are necessary when using the microscopic model due to the higher stochasticity present in the model. It is further worth noting that, while we do not use the same computational budget for different approaches, a fair comparison has been one of our key concerns for this work. Given the relatively simple nature of the task, the 2-step approach proposed in this work quickly reaches a minimum, due to the noise inherent to robotic benchmarks, which is indistinguishable from a global minimum. However, a single step approach exclusively relying on high-fidelity simulation cannot converge to any minimum with the very same computational budget for the evaluation of candidate solutions. We therefore believe that the cost evolution over the computational budget dedicated to the evaluation of candidate solutions (Figure 6), in our case expressed as equivalent evaluations in a high-fidelity simulator, is not only fair but also more illustrative of the differences between the various optimization variants.

The OCBA parameters $n0$ and Δ are kept constant at 2 and 4 respectively following the recommendations in [25]. The optimization budget per MDPSO iteration is set to $4N_P$. Early convergence is achieved after 20 MDPSO iterations without change of the global best solution.

TABLE III: Maximum evaluation budgets used in the sub-microscopic model (sM) and the microscopic model (μM) for the different approaches.

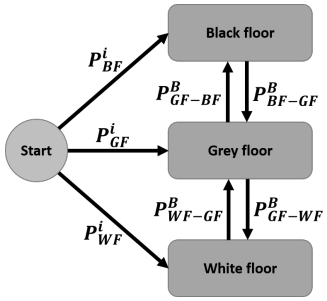
	$\mathbf{N_{P,sM}}$	$\mathbf{N_{E,sM}}$	$\mathbf{N_{P,\mu M}}$	$\mathbf{N_{E,\mu M}}$
A1	32	4000	-	-
A2	16, 8	2000, 1000	-	-
A3	8	1000	16	40000

D. Multi-Level Modeling

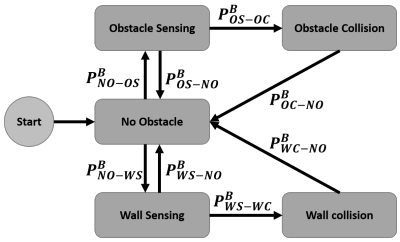
The multi-level hierarchical representation of distributed robotic systems introduced in [13] includes three increasingly abstract modeling levels: submicroscopic, microscopic, and macroscopic. However, in this work we are only using the first two levels: submicroscopic and microscopic.

1) Submicroscopic Modeling: The submicroscopic model has been implemented using Webots [26], a high-fidelity, open-source robotics simulation platform. For each candidate solution evaluation, the robot is randomly placed in the 2 x 2 m arena populated with five to seven randomly placed, non-overlapping obstacles.

2) Microscopic Modeling: In both reality and submicroscopic simulation, changes of environmental states happen through physical displacement of the robot. Following the probabilistic framework of [13], we can abstract such displacements through calibrated transition probabilities and definition of states of interest, typically capturing interactions between the robot and the environment. In other words, we can capture spatiality in an approximated way through the use of a calibrated PFSM. However, two difficulties prevent us from applying the previously mentioned framework in a straighforward way. First, the controller structure in this work is not established a priori and therefore, in contrast to [13], it cannot be directly leveraged as blueprint for the structure of the microscopic model. As a consequence, the microscopic model must incorporate a new dimension in its transition probabilities, namely that of the chosen behavior (upper index "B") for any particular transition. Second, the spatiality can only be partially abstracted as the robot navigates from specific zones to others in the arena, and well-mixing conditions (due to stochastic interactions and randomized initialization over repeated runs) can be only partially assumed. As a result, the rough spatiality based on the differently colored floor zones must be taken into account, as well as the geometric and estimation reasoning for the different transition probabilities.



(a) PFSM of the floor color state



(b) PFSM of the collision state

Fig. 3: Perceptual PFSMs forming the structure of the microscopic model used in this work. While both PFSMs could be merged in a single, nested, large PFSM, for sake of clarity, we have preferred to illustrate them as two separate representation for the two states concerned.

Model structure: In our example, the perceptual state of a robot captured with exteroceptive sensors consists of the combination of two independent states: *floor color* and

collision. Consequently, two PFSMs, as illustrated in Figure 3, are leveraged to model the environmental interactions, meaning that the robot has both an active floor color state and collision model state (e.g., white floor and wall collision) at every moment. For both PFSMs, an independent probabilistic decision is taken at every simulation timestep to determine any potential perceptual state changes.

- *Floor color*: The robot is initialized randomly with a probability P^i_{BF}, P^i_{WF} and P^i_{GF} of being on black, white or grey floor respectively. These probabilities are based on their respective geometric probabilities. Thereafter, the probability to transition from color X to color Y is expressed as P^B_{X-Y}, where B corresponds to the currently active behavior. Note that it is impossible to transition directly from black floor to white floor and vice-versa, as this is not physically possible in our experimental setup (see Figure 1).
- *Collision*: The robot is initialized in No Obstacle (NO) state. It is assumed that the robot cannot collide with an obstacle without having first sensed the obstacle. As a result, the Obstacle Collision (OC) state can only be reached from the Obstacle Sensing (OS) state. Moreover, depending on the robot behavior, the robot's interaction with the walls differs from the interaction with the obstacles. Therefore, in contrast to [13], separate transition probabilities P^B_{NO-WS} and P^B_{WS-WC} have to be used for Wall Sensing (WS) and Wall Collision (WC) respectively. In order to keep the model simple, it is assumed that obstacle- and wall-events are mutually exclusive, which is not necessarily always true.

Model parameters: Each behavior in $\mathcal{B}$ needs its representation in the microscopic model in the form of a set of state transition probabilities for the two PFSMs described above.

Consistent with previous works [13], [27], whenever the well-mixing assumption (or homogeneous spatial distribution in this case) is fulfilled, the transition probabilities P_i are calculated using geometric encountering rates:

$$P_i = P_{geo} = r_i \cdot T_{\mu M} = \frac{v^B W_{S,j}}{A_{S,j}} g_i \cdot T_{\mu M} \qquad (4)$$

with $T_{\mu M}$ the simulation timestep and r_i their respective encountering rates given by v^B the robot's speed depending on the currently active behavior, $A_{S,j}$ the detection area of the smallest object of type of j present in the arena, $W_{S,j}$ the detection width for the smallest object of type j, where $j \in \{floor, obstacles\}$, depending on the nature of object i. $g_i = \frac{A_i}{A_a}$ is the geometrical probability using A_i the detection area of object i and A_a is the whole arena area.

In case the well-mixing assumption is not fulfilled, calibrated probabilities representing time delays are used: $P_i = P_{delay} = \frac{1}{T_i}$ with T_i the average time delay for transition i.

The type of probability for each of the three behaviors for all perceptual transition probabilities is summarized in Table IV with the numerical parameters listed in Table V, in the appendix. Note that, in order to match more faithfully the results of the submicroscopic model, all transitions to grey floor and no obstacle state are calibrated on actual mean delays achieved in the more faithful representation.

TABLE IV: Perceptual state transition probability types and analytical expressions used in the microscopic model

Perc. probability	Braitenberg	Light following	Stop [i]
P_{WF-GF}	$P_{delay} - \frac{2R_{WF}}{v^{BB}T_{\mu M}}$	/ − 0 [ii]	$P_{geo} - 0$
P_{GF-WF}	P_{geo}	$P_{delay} - \frac{\mathbb{E}[dist]}{v^{LF}}$ [iii]	$P_{geo} - 0$
P_{GF-BF}	P_{geo}	P_{geo} [iv]	$P_{geo} - 0$
P_{BF-GF}	$P_{delay} - \frac{2R_{BF}}{v^{BB}T_{\mu M}}$	P_{geo} [iv]	$P_{geo} - 0$
P_{NO-OS}	P_{geo}	P_{geo}	$P_{geo} - 0$
P_{OS-OC}	/ - 0 [v]	P_{geo}	$P_{geo} - 0$
P_{NO-WS}	P_{geo}	P_{manual} [vi]	$P_{geo} - 0$
P_{WS-WC}	/ - 0 [v]	P_{manual} [vi]	$P_{geo} - 0$
P_{OC-NO}	/ [v]	$P_{delay} - \frac{1}{T^{LF}_{OC-NO}T_{\mu M}}$	$P_{geo} - 0$
P_{OS-NO}	$P_{delay} - \frac{1}{T^{BB}_{OS-NO}T_{\mu M}}$	P_{geo}	$P_{geo} - 0$
P_{WC-NO}	/ [v]	$P_{delay} - 0$ [vi]	$P_{geo} - 0$
P_{WS-NO}	$P_{delay} - \frac{1}{T^{BB}_{OS-NO}T_{\mu M}}$	P_{geo}	$P_{geo} - 0$

[i] $v = 0$, thus no perceptual changes happen
[ii] Light following does not leave white floor due to the light's position.
[iii] $\mathbb{E}[dist] \sim 1.4$m is the average distance from every position within the area to the beginning of the white zone.
[iv] The geometric probability used is scaled with respect to P_{GF-WF}.
[v] Braitenberg obstacle avoidance is assumed to be perfect.
[vi] As the robot heads towards the light, the probability of sensing and colliding with a wall is zero, until after it reaches the white zone when it is certain to collide with the wall, as the light is located beyond the arena.

IV. Results

In this section we report the performances of the FSM-synthesizing approaches introduced in Section III-B on the scenario introduced in Section II.

A. Experimental Campaign Details

The overall physical set-up is shown in Figure 1. Experiments have been carried out with a Khepera IV [28], a differentially driven robot with a diameter of 14 cm and a maximal speed of $\sim$ 81 cm/s. For this work, we exclusively leveraged the infrared-sensors of the Khepera IV robot. The eight sensors placed in regular distances of 45° around the edge of the robot are used both as proximity sensors for the Braitenberg behavior and as ambient light sensor for the light-following behavior. The four sensors underneath the robot are used to detect the current floor color. Our Khepera IV robot is also equipped with an active marker module featuring two LEDs for enabling accurate tracking with the SwisTrack software [29].

Every FSM-synthesizing approach is repeated 20 times. In order to assess the quality of their respective behavioral arbitrators, the final candidate of each optimization run is re-evaluated 20 times in the Webots simulator. A fully manually defined FSM-based behavioral arbitrator, as illustrated in Figure 2, evaluated in the same way, provides a reference performance. Furthermore, the performance of the resulting best performing candidate solution for every approach is validated 10 times in reality.

It is worth noting that, in order to achieve a consistent validation in reality across the approaches, obstacle numbers

and positions as well as the initial position of the Khepera IV robot have been fixed as shown in Figure 1.

B. Numerical Results

Figure 4 reports the re-evaluated performance of the optimized FSM-based arbitrator the discrete component re-evaluated in the Webots simulator (blue), with both discrete and continuous components in Webots (orange) and in reality (green). It is apparent that all three approaches lead to highly competitive solutions in comparison to the manual one, both in Webots and using real robots. However, we also note that the one-step approach using Webots (A1) leads to the highest spread in performance. This is supposedly due to the difficulty of optimizing both the continuous and discrete part in parallel. The performance of both two-step approaches, A2 and A3, is as good as the manual solution both in simulation and reality, with only slightly higher variability in simulation. Given the natural ability of humans to design FSMs and the designer's previous knowledge about the experiment, we can consider the manual solution to be almost ideal, highlighting even more the good performance of A2 and A3.

The re-evaluated costs in reality (green boxes in Figure 4) are lower for all approaches. This can be explained by the fact that for each approach only the best performing candidate solution has been evaluated ten times in reality, improving thus the expected performance. Another difference between real and simulated experiments is that in reality, both the robot's initial position and the environment have been kept static across all evaluations, thus avoiding to the stochasticity introduced through the initialization.

Figure 5 provides a fair comparison between simulation and reality. Here, only the best performing candidate solutions of each approach have been re-evaluated 100 times in simulation and 10 times in reality. Both the initial position and the environment were kept static and identical to the real experiments, in an attempt to a comparison as fair as possible. However, we can still observe a small, but non negligible reality gap as well as an increase in stochasticity in the real robot experiments.

Figure 6 reports the learning curve of the three approaches. For representing purposes, the evaluations in the microscopic model (uM) are represented as time-equivalent evaluations in the submicroscopic model (Webots, WB). A3, the two-step uM-WB approach obtains competitive results most quickly, followed by A2, the two-step WB-WB approach. This is shown even more precisely in Figure 7 which reveals the time spent on an Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz with 16 GB of RAM and a NVIDIA GeForce GTX 960M graphics card for each optimization stage for all approaches, without considering possible speedups through parallelization of the candidate evaluations. In comparison to A1, the one-step WB approach, A2, the two-step WB-WB approach, is able to save ∼25% and A3, the two-step uM-WB approach, ∼75% of the computational cost.

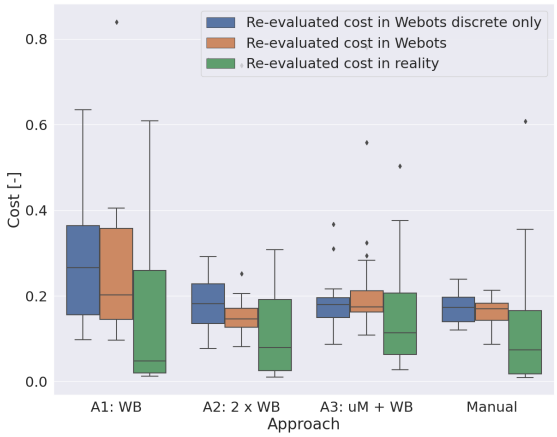


Fig. 4: Comparison of the solutions generated using the different synthesizing strategies re-evaluated using only the discrete part of the solutions in Webots (blue), using the full solutions in Webots (orange), and in reality (green). Only the best solution for each approach was re-evaluated in reality.

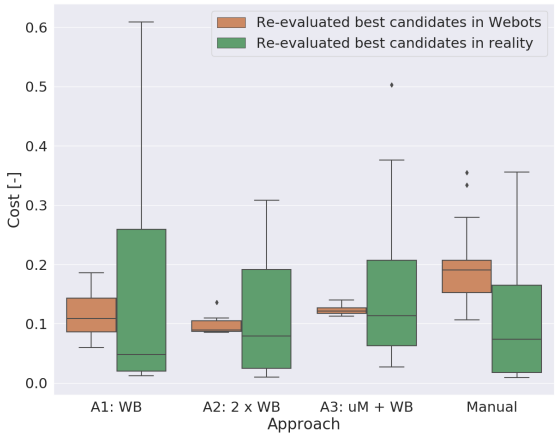


Fig. 5: Comparison of the best candidate solutions generated using the different synthesizing strategies re-evaluated in a static environment with constant initial positions in both Webots (orange) and real experiments (green).

V. Discussion

We find that all synthesizing approaches result in competitive solutions in terms of the performance of the resulting FSM. Both A2 and A3 manage to increase the performance on A1 by decreasing the variability of the resulting solution, achieving more consistently performances similar to the manual one. We further note that in comparison to the one-step WB approach, significant time gains can be achieved through two-step approaches. Notably the time gain of ∼75% for A3 is worth highlighting.

These results could be even more impactful for cases where a high-fidelity simulation environment is absent or unreliable, and thus high-resolution assessments of a candidate solution require real experiments, which are cumbersome

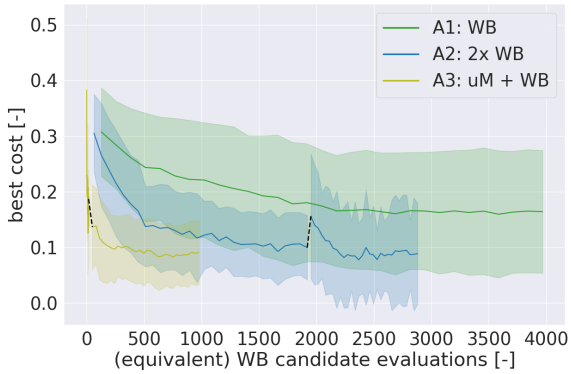
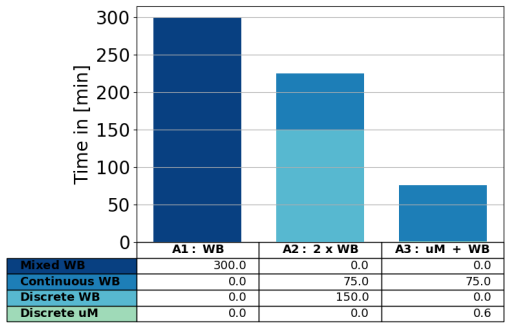


Fig. 6: Comparison of the solutions performance as reported during their learning process. The evaluations in the microscopic model are represented as wall-clock time-equivalent evaluations of the submicroscopic model (Webots). Dotted lines correspond to a change from the first to the second optimization step. Shaded areas indicate standard deviations.



	A1 : WB	A2 : 2 x WB	A3 : uM + WB
Mixed WB	300.0	0.0	0.0
Continuous WB	0.0	75.0	75.0
Discrete WB	0.0	150.0	0.0
Discrete uM	0.0	0.0	0.6

Fig. 7: Comparison of the wall-clock time needed per synthesizing strategy in minutes.

and time-consuming. Indeed for the scenario used here, one might get away with roughly 500 experiments using A3 versus over 2000 using A1.

The reality gap observable between the re-evaluated cost of the best performing solutions in Webots and reality, apparent in Figure 5, is small but not negligible. However, this work focuses on leveraging multiple levels of modeling for control generation, with the real robot experiments validating the results obtained in our (calibrated) high-fidelity simulation. Nonetheless, further reality gap reducing techniques such as including the simulation-to-reality disparity measure in the optimization process, as introduced in [30], could be used between microscopic and submicroscopic modeling levels or between the submicroscopic level and reality, in particular if the goal is to optimize the performance in reality rather than speeding up the optimization process.

A major limitation of this work is the need of an initial (manual) modelling effort for A3. However, it is worth noting that any level of modelling used, including submicroscopic models (i.e. high-fidelity simulators), requires an initial modelling effort for both elements of the scenario (i.e. representation of the physical world and robot), which can then be partially re-used in future scenarios. In the case of the microscopic model used in A3, on the one hand, both the perceptual model and the behavioral representations in the form of a set of state transition probabilities are, from a methodological perspective, not specific for a given scenario. On the other hand, their numerical values are depending on the specific parameters of the environment (e.g., P_{GF-WF} depends on the size of the arena). Thus, such modeling approach makes the resulting models partially re-usable and thus more viable.

Another limitation of this work is concerned with the low complexity of the considered scenario. However, this is the outcome of a deliberate choice: in this paper, we wanted to illustrate the methodological aspect with a relatively simple scenario. Nonetheless, from a methodological point of view, the approach can be directly applied to more complex problems including scenarios with multiple robots if the modelling assumptions in [13] are fulfilled. Indeed, both the control design and modelling framework used were originally designed for multi-robot systems. However, the applicability of our approach to more challenging and complex tasks, possibly involving multiple robots, needs to still be demonstrated in future work. Tackling more complex missions will also allow us to demonstrate the potential reusability of the models for each basic behavior.

VI. Conclusion

In this work we have leveraged the FSM generation framework introduced in [8] and combined it with the modelling framework in [13]. Through a proper parametrization and corresponding calibration exploiting a submicroscopic model, we were able to leverage a microscopic model to synthesize the discrete part of the FSM for a single robot scenario. Due to the low-level nature of the continuous parameters of the FSM, they could not be represented explicitly in the microscopic model. As a result, these parameters had to be learned in a second step using the submicroscopic model. We compared this two-step approach with another two-step approach using the submicroscopic model for both steps, a standard one-step approach using only the submicroscopic model, and a manually designed FSM.

We have shown that the two-step approach using the microscopic model results in a significant reduction of the computational cost (75%) in comparison to the standard one-step approach, while maintaining the performance and decreasing the variability of the resulting solution.

We further note that, even though this work focuses on a relatively simple single-robot scenario to demonstrate the viability of the approach, the method introduced here has been designed with more complex scenarios in mind, including multi-robot systems, in mind. Consequently, to evaluate its general viability, future work will include applying the proposed approach to those scenarios.

References

[1] M. Birattari, A. Ligot, D. Bozhinoski, M. Brambilla, G. Francesca, L. Garattoni, D. Garzón Ramos, K. Hasselmann, M. Kegeleirs, J. Kuckling, F. Pagnozzi, A. Roli, M. Salman, and T. Stützle, "Automatic Off-Line Design of Robot Swarms: A Manifesto," *Frontiers in Robotics and AI*, vol. 6. art. 59, 2019.

[2] F. Mukhlish, J. Page, and M. Bain, "Evolutionary-learning framework: improving automatic swarm robotics design," *International Journal of Intelligent Unmanned Systems*, vol. 6, no. 4, pp. 197–215, 2018.

[3] G. Sartoretti, Y. Wu, W. Paivine, T. K. S. Kumar, S. Koenig, and H. Choset, "Distributed Reinforcement Learning for Multi-robot Decentralized Collective Construction," in *Distributed Autonomous Robotic Systems*, pp. 35–49, 2019.

[4] M. Chavoshian, M. Taghizadeh, and M. Mazare, "Hybrid Dynamic Neural Network and PID Control of Pneumatic Artificial Muscle Using the PSO Algorithm," *International Journal of Automation and Computing*, vol. 17, no. 3, pp. 428–438, 2020.

[5] M. Borg, C. Englund, K. Wnuk, B. Duran, C. Levandowski, S. Gao, Y. Tan, H. Kaijser, H. Lönn, and J. Törnqvist, "Safely Entering the Deep: A Review of Verification and Validation for Machine Learning and a Challenge Elicitation in the Automotive Industry," *arXiv:1812.05389*, 2018.

[6] F. Silva, M. Duarte, L. Correia, S. M. Oliveira, and A. L. Christensen, "Open Issues in Evolutionary Robotics," *Evolutionary Computation*, vol. 24, no. 2, pp. 205–236, 2016.

[7] T. A. Henzinger, "The theory of hybrid automata," in *IEEE Symposium on Logic in Computer Science*, pp. 278–292, 1996.

[8] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari, "AutoMoDe: a novel approach to the automatic design of control software for robot swarms," *Swarm Intelligence*, vol. 8, pp. 89–112, 2014.

[9] M. Birattari, A. Ligot, and G. Francesca, "AutoMoDe: A Modular Approach to the Automatic Off-Line Design and Fine-Tuning of Control Software for Robot Swarms," in *Automated Design of Machine Learning and Search Algorithms*, Natural Computing Series, pp. 73–90, 2021.

[10] E. Ferrante, E. Duenez-Guzman, A. Turgut, and T. Wenseleers, "GESwarm: grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics," in *ACM Genetic and Evolutionary Computation Conference*, pp. 17–24, 2013.

[11] K. Hasselmann, F. Robert, and M. Birattari, "Automatic design of communication-based behaviors for robot swarms," in *International Conference on Swarm Intelligence (ANTS)*, Lecture Notes in Computer Science, pp. 16–29, 2018.

[12] E. Di Mario and A. Martinoli, "Distributed Particle Swarm Optimization for limited-time adaptation with real robots," *Robotica*, vol. 32, no. 02, pp. 193–208, 2014.

[13] A. Martinoli, K. Easton, and W. Agassounon, "Modeling Swarm Robotic Systems: a Case Study in Collaborative Distributed Manipulation," *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 415–436, 2004.

[14] M. B. Egerstedt, K. H. Johansson, J. Lygeros, and S. Sastry, "Behavior based robotics using regularized hybrid automata," in *IEEE Conference on Decision and Control*, pp. 3400–3405, 1999.

[15] U. Furbach, J. Murray, F. Schmidsberger, and F. Stolzenburg, "Hybrid Multiagent Systems with Timed Synchronization – Specification and Model Checking," in *Programming Multi-Agent Systems*, Lecture Notes in Computer Science, pp. 205–220, 2008.

[16] A. Mohammed and U. Furbach, "Multi-Agent Systems: Modeling and Verification Using Hybrid Automata," in *Programming Multi-Agent Systems*, Lecture Notes in Computer Science, pp. 49–66, 2010.

[17] A. Mohammed, U. Furbach, and F. Stolzenburg, "Multi-Robot Systems: Modeling, Specification, and Model Checking," *IntechOpen Robot Soccer*, 2010.

[18] M. Brambilla, A. Brutschy, M. Dorigo, and M. Birattari, "Property-Driven Design for Robot Swarms: A Design Method Based on Prescriptive Modeling and Model Checking," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 9, no. 4, pp. 1–28, 2014.

[19] A. Reina, G. Valentini, C. Fernández-Oto, M. Dorigo, and V. Trianni, "A Design Pattern for Decentralised Decision Making," *PloS one*, vol. 10. art. e0140950, pp. 1-18, 2015.

[20] G. Mermoud, M. Mastrangeli, U. Upadhyay, and A. Martinoli, "Real-time automated modeling and control of self-assembling systems," in *IEEE International Conference on Robotics and Automation*, pp. 4266–4273, 2012.

[21] A. Vergnano, C. Thorstensson, B. Lennartson, P. Falkman, M. Pellicciari, F. Leali, and S. Biller, "Modeling and Optimization of Energy Consumption in Cooperative Multi-Robot Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 423–428, 2012.

[22] S. Chowdhury, W. Tong, A. Messac, and J. Zhang, "A mixed-discrete particle swarm optimization algorithm with explicit diversity-preservation," *Structural and Multidisciplinary Optimization*, vol. 47, pp. 367–388, 2013.

[23] C.-H. Chen, J. Lin, E. Yücesan, and S. Chick, "Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization," *Discrete Event Dynamic Systems: Theory and Application*, vol. 10, pp. 251–270, 2000.

[24] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 3, pp. 396–407, 1996.

[25] E. Di Mario, I. Navarro, and A. Martinoli, "A distributed noise-resistant Particle Swarm Optimization algorithm for high-dimensional multi-robot learning," in *IEEE International Conference on Robotics and Automation*, pp. 5970–5976, 2015.

[26] O. Michel, "WebotsTM: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, pp. 39–42, 2004.

[27] N. Correll and A. Martinoli, "Collective Inspection of Regular Structures using a Swarm of Miniature Robots," in *Ninth International Symposium on Experimental Robotics*, June 2004. Springer Tracts in Advanced Robotics (2006), pp. 375-385.

[28] J. M. Soares, I. Navarro, and A. Martinoli, "The Khepera IV mobile robot: performance evaluation, sensory data and software toolbox," in *Second Iberian Robotics Conference*, vol. 417, pp. 767–781, 2016.

[29] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli, "SwisTrack - a flexible open source tracking software for multi-agent systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4004–4010, 2008.

[30] S. Koos, J.-B. Mouret, and S. Doncieux, "The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 122–145, 2013.

APPENDIX

TABLE V: Numerical values of parameters.

Description	Num. Value
A_A Arena area	4 m^2
R_{k4} Robot radius	0.07 m
R_S Sensor range	0.09 m
R_{obs} Obstacle radius	0.106 m
N_{obs} Number of obstacle	7
R_{WF} Radius of white zone area	0.4 m
R_{BF} Radius of a black zone area	0.3 m
A_{WF} white zone area	0.126 m^2
A_{BF} black zone area	0.565 m^2
$A_{S,floor}$ Detection area of the smallest floor object	0.43 m^2
$A_{S,obstacles}$ Detection area of the smallest obstacle object	0.22 m^2
$W_{S,floor}$ Detection width for smallest floor object	0.532 m^2
$W_{S,obstacles}$ Detection width for smallest obstacle object	0.14 m^2
$T_{\mu M}$ Timestep of microscopic model	0.1 s
T^{BB}_{OS-NO} Avg. time within sensing range of an obstacle	4.6 s
T^{LF}_{OC-NO} Avg. time in collision with an obstacle	1 s
v^{LF} Avg. Robot speed in LF	0.2 m/s
v^{BB} Avg. Robot speed in BB	0.08 m/s
$P^i_{WF} = \frac{A_{WF}}{A_A}$	0.032
$P^i_{BF} = \frac{A_{BF}}{A_A}$	0.141
$P^i_{GF} = 1 - P^i_{BF} - P^i_{WF}$	0.827
$A_{OS} = N_{Obs}(R_{k4} + R_{Sen} + R_{Obs})^2\pi$	1.54m^2
$A_{OC} = N_{Obs}(R_{k4} + R_{Obs})^2\pi$	0.68m^2
$g_{BF} = \frac{A_{BF}}{A_A}$	0.14
$g_{WF} = \frac{A_{WF}}{A_A}$	0.033
$g_{OS} = \frac{A_{OS}}{A_A}$	0.33
$g_{OC} = \frac{A_{OC}}{A_{OS}}$	0.33
$g_{WS} = \frac{A_{WS}}{A_A}$	0.2944