CO-SIMULATION OF BIO-INSPIRED MULTI-AGENT ALGORITHMS

Cinzia Bernardeschi
Andrea Domenici
Maurizio Palmieri

Department of Information Engineering
University of Pisa
Via Caruso 16, Pisa, Italy
{name.surname}@ing.unipi.it

Adriano Fagiolini

Department of Energy, Information Engineering
and Mathematical Models
University of Palermo,
Viale delle Scienze, Palermo, Italy
{adriano.fagiolini}@unipa.it

ABSTRACT

This paper reports on the co-simulation of a team of robots deployed in an exploration task, coordinated by a bio-inspired exploration algorithm. The co-simulation integrates the high-level exploration algorithm with detailed implementations of the robot controllers and kinematic models. Co-simulation results are used to find and correct mismatches between submodels.

Keywords: Co-simulation, Cyber-Physical Systems, bio-inspired algorithms, map exploration.

1 INTRODUCTION

Cyber-Physical Systems (CPSs) are complex systems that integrate digital controllers and physical components. Simulation is one of the techniques that are usually applied, together with testing, in the analysis of system behaviors. Due to the existence of continuous and discrete components, simulation of CPSs often takes the form of co-simulation (Gomes et al. 2018), which allows the coordinated execution of different sub-systems, each modeled and simulated with its most appropriate tool.

Bio-inspired cooperative algorithms are exploited in several robotic applications for spatial exploration in an unknown environment, i.e., where some target must be reached, and each robot has no a priori knowledge of the environment and can only acquire local information from its sensors and possibly from communication with other robots. In most cases, safety-related targets are involved, such as fires, mines, human casualties to rescue, or dangerous materials.

In this work, a coordination algorithm (Palmieri, Yang, Rango, and Marano 2017) based on Ant Colony Optimization (ACO) (Dorigo et al. 2006) has been implemented. This algorithm has been co-simulated on the INTO-CPS platform (Larsen et al. 2018) together with a control algorithm for point-to-point movement of the robot and with a model of the robot kinematics. The implementation has been validated by simulation, and the results of various simulations made it possible to tune both the coordination and the control algorithms' implementation.

The main contribution of this paper is showing the value of using co-simulation and design space exploration in the development of a CPS obtained by composing a high-level control algorithm with pre-existing closed

SummerSim, July 20-July 22, 2020, Madrid, Spain; ©*2020 Society for Modeling & Simulation International (SCS)*

implementations of lower-level robot kinematics. Co-simulation helped developers pinpoint issues arising from different assumptions on the two models.

The rest of this paper is structured as follows: a selection of related works is presented in Sec. 2, basic notions on the algorithms and co-simulation framework are introduced in Sec. 3, and Sec. 4 illustrates the simulation workflow as applied in the case at hand. Sec. 5 concludes the paper.

2 RELATED WORK

This section presents a selection of works on multi-robot coordination and on CPS co-simulation.

A very large and growing body of literature is available on the topic of multi-robot system coordination. Many works follow *bio-inspired* approaches, focused on various forms of *swarm intelligence* emerging from the behavior of natural living societies, such as flocks of birds, ant or bee or bacterial colonies.

Work (De Rango et al. 2015) provides a description of possible approaches where a bio-inspired exploration algorithm has been applied to a swarm of robots in order to explore a mined region and to start a cooperative task to secure the area. Also Masár (Masár 2013) proposes biologically inspired methods and algorithms for space exploration and surveillance.

Many works are based on the concept of *stigmergy*, i.e., the mechanisms whereby the movement of individual members of an insect society is guided by the intensity of olfactive tracks left by other individuals. For example, (Alfeo et al. 2019) report on the swarm coordination of Unmanned Aerial Vehicles, using a differential evolutionary algorithm. Paper (Dorigo et al. 2006) presents the ACO approach as a bio-inspired strategy for optimization problems, and the research in (Chen et al. 2013) exploits pheromone-based exploration with the ACO method.

Karaboga and Akay (Karaboga and Akay 2009) report on a bio-inspired approach based on Artificial Bee Colony for optimization of a large set of numerical test functions. Their results have been compared with those obtained by genetic algorithms, Particle Swarm Optimization (PSO) algorithms, differential evolution algorithms, and evolution strategies. PSO has been applied by Pugh et al. (Pugh et al. 2005) to optimization problems with noisy perturbancies.

The fundamental concepts of co-simulation are exposed in many works, such as (Gomes et al. 2018), and the Functional Mock-up Interface (FMI) standard is defined in (Blochwitz et al. 2012). In the following, some works more closely related to the present paper are referenced.

A methodology for formal analysis and modeling of software components in CPSs is presented in (Bernardeschi et al. 2018), together with supporting tools. The methodology describes how to integrate a simulation of logic-based specifications of software components with Simulink models of continuous processes.

Previous work (Palmieri, Bernardeschi, and Masci 2017) takes into account possible critical applications and aims at exploiting co-simulation to perform preemptive analysis, in order to detect conditions that may cause the system to violate safety requirements. The Line Following Robot case study (The INTO-CPS Association 2019) available from the INTO-CPS project (Larsen et al. 2018) is used as an example. Moreover, a framework for the co-simulation of CPSs involving human-machine interfaces was presented by the authors in (Palmieri et al. 2020). The framework supports formal methods experts in the analysis of safety-critical aspects of user interfaces, and generates interactive FMI-compliant prototypes.

Design Space Exploration (DSE) is used in (Palesi and Givargis 2002) to explore efficiently a parametric system in order to find optimal configurations in a multi-objective design space.

3 BACKGROUND

This section describes the bio-inspired exploration algorithm, the controller for the movement of a single robot and the co-simulation framework.

3.1 A bio-inspired exploration algorithm

One category of bio-inspired algorithms is *stigmergic algorithms*. These algorithms are abstractions of behaviors exhibited by social insects: Each individual releases an amount of substances called pheromones, whose presence is sensed by other individuals, which make choices based on that information.

Stigmergic exploration algorithms maintain a map of the area to explore. The map is partitioned into cells, e.g., by a rectangular grid. The map is shared among all agents involved in the exploration, but each agent can access only a limited area around its current position. Each cell has a value, *pheromone level* in the following, used to mimic the release and sensing of pheromone. An agent "releases pheromone" by increasing the pheromone level in its current and nearby cells. The release of pheromone decreases with distance from the current cell, and afterwards the pheromone *evaporates*, i.e., its pheromone level decreases with time.

In this work, the exploration algorithm presented in (Palmieri, Yang, Rango, and Marano 2017) has been considered as a case study. The goal of this discrete-time algorithm is to maximize the area explored by a number $\mathrm{N_R}$ of robots. The cited algorithm specifies (i) the computation of the pheromone level (Equations (1–4)), and (ii) the exploration strategy (Equations (5) and (6)).

Each robot releases pheromone in the cells contained within a radius $\mathrm{R_s}$ from the current position (the cell center). If a robot r at time t is at point $(x_k^{(t)}, y_k^{(t)})$, its distance from a cell c is

$$r_{kc}^{(t)} = \sqrt{(x_k^{(t)} - x_c)^2 + (y_k^{(t)} - y_c)^2} \tag{1}$$

and the amount of pheromone released in cell c is

$$\Delta\tau_{kc}^{(t)} = \Delta\tau_0 e^{-\frac{r_{kc}^{(t)}}{a_1}} + \frac{\varepsilon}{a_2} \quad \text{if } r_{kc}^{(t)} \leq \mathrm{R_s}; \qquad \Delta\tau_{kc}^{(t)} = 0 \quad \text{otherwise}\ , \tag{2}$$

where the constant $\Delta\tau_0$ is the maximum pheromone release, ε is a uniform random variable representing noise, and a_1 and a_2 are tuning parameters of the algorithm.

With $\rho_c^{(t)}$ the evaporation fraction and $\tau_c^{(t)}$ the pheromone level for cell c at time t, the pheromone evaporated at time t , named $\xi_c^{(t)}$, is shown in Equation 3. The evaporation fraction depends on the *evaporation rate per time unit* coefficient ($\mathrm{ERTU_{\%}}$) and on the time elapsed since the last time t_v the cell has received pheromone. The value of t_v is set to 0 at the beginning for all the cells.

$$\rho_c^{(t)} = (t - t_v)\mathrm{ERTU_{\%}}\ ; \qquad \xi_c^{(t)} = \rho_c^{(t)} \tau_c^{(t)}\ . \tag{3}$$

The pheromone level of a cell is then given by

$$\tau_c^{(t)} = 0 \quad \text{for} \quad t = 0\ ; \qquad \tau_c^{(t)} = \tau_c^{(t-1)} - \xi_c^{(t-1)} + \sum_{k=1}^{\mathrm{N_R}} \Delta\tau_{kc}^{(t)} \quad \text{for} \quad t > 0\ . \tag{4}$$

The exploration strategy is based on a repulsive reaction to pheromone by the robots, i.e., each robot tries to avoid cells that have already been visited. This strategy lets robots move towards unexplored regions of the map with a higher probability, with the objective of improving the map coverage in a given time.

If $c_k^{(t)}$ is the cell occupied by robot k at time t, the set of cells accessible from $c_k^{(t)}$ is denoted by $A(c_k^{(t)})$. This set contains only the cells whose center is within the radius $\mathrm{R_S}$ and are not occupied by an obstacle or a robot. In typical applications of the ACO algorithm, the pheromone level has an attractive effect and individual agents tend to follow or join each other. This attractive behavior can be modeled by expressing the probability that the robot moves to a cell $c \in A(c_k^{(t)})$ as

$$p(c|c_k^{(t)}) = \frac{(\tau_c^{(t)})^{\phi}(\eta_c^{(t)})^{\lambda}}{\sum_{b\in A(c_k^{(t)})}(\tau_b^{(t)})^{\phi}(\eta_b^{(t)})^{\lambda}}\ , \tag{5}$$

where η is a heuristic correction factor used to escape local minima, and ϕ and λ are tuning parameters. To obtain a repulsive behavior, the exploration algorithm chooses a cell $\bar{c}^{(t)}$ satisfying the condition

$$p(\bar{c}^{(t)}|c_k^{(t)}) = \min_{c\in A(c_k^{(t)})}[p(c|c_k^{(t)})]\ . \tag{6}$$

3.2 Robot control

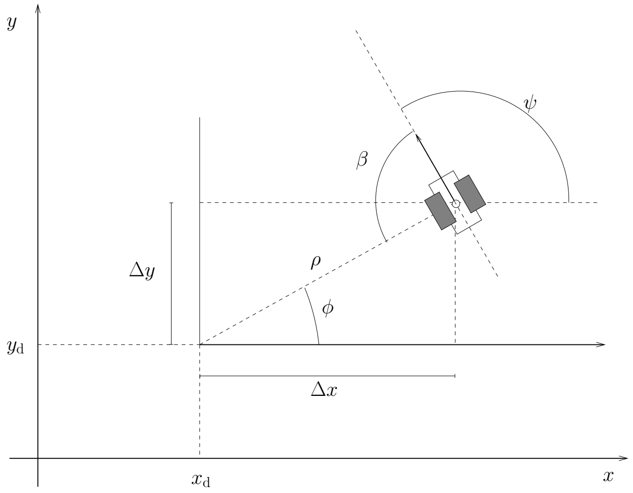


Figure 1: Reference frames for robot movement.

In this work, it is assumed that an exploration task is carried out by a number of single-axle robotic vehicles, controlled in linear speed v and turning speed ω. At each step of the algorithm described above, each robot moves from its current cell to the next one, chosen by the algorithm. To control the robot in this point-to-point movement, a method from (Siciliano and Khatib 2016) has been chosen for its simplicity. With this method, the robot coordinates are transformed from the map's Cartesian frame to a local polar frame (ρ,ϕ) originated in the target waypoint $(x_\mathrm{d},y_\mathrm{d})$ (Fig. 1). In this frame, angle ϕ is the robot's bearing as seen from the target waypoint, while ψ is the robot's heading in the map's frame. Angle β is the error between the instantaneous heading and the desired heading:

$$\beta = \arctan\left(\frac{y-y_\mathrm{d}}{x-x_\mathrm{d}}\right) - \psi + \pi\ , \tag{7}$$

It is worth recalling that the "arctan" function is well-defined over the entire real domain. However, its input argument can diverge as x approaches x_d when $y \neq y_\mathrm{d}$. An accepted solution to this issue is to use the alternative function "atan2", which accepts numerator and denominator as two separate arguments and thus is numerically stable.

The robot movement is split in two phases: a *maneuver* phase to align the heading to the waypoint, and a *moving* phase to reach it. If k_β and k_v are the feedback gains for error β and speed v, respectively, the control law in the maneuver phase is

$$\begin{cases} v = 0 \\ \omega = \mathrm{k}_\beta \beta \end{cases} \tag{8}$$

and the control law in the moving phase is

$$\begin{cases} v = \mathrm{k}_v \rho (\rho^2 \cos\beta + \beta^2 \sin\beta) \\ \omega = \mathrm{k}_\beta \beta \; . \end{cases} \tag{9}$$

The rotational speeds ω_l and ω_r of the left and right wheel depend on wheel radius R and axle length L:

$$\begin{cases} \omega_\mathrm{l} = \dfrac{\mathrm{L}\omega + 2v}{2\mathrm{R}} \\ \omega_\mathrm{r} = \dfrac{\mathrm{L}\omega - 2v}{2\mathrm{R}} \; . \end{cases} \tag{10}$$

3.3 FMI-based co-simulation

The FMI (Blochwitz et al. 2012) is a tool-independent standard for co-simulation and model exchange of dynamic models. It defines a container, called Functional Mock-up Unit (FMU), and an interface defining interaction and data exchange between FMUs. An FMU contains the executable implementation of a model, created with the tool chosen by its developers, and the information or tools needed to execute it.

An FMI-complying host environment provides a master program that orchestrates the FMUs, implementing a co-simulation algorithm and providing a communication framework. The host environment used in this work is the INTO-CPS Application (Larsen et al. 2018). The INTO-CPS application has a form-based user interface to define the interconnections between FMUs and to display the evolution in time of the simulated variables.

4 SIMULATION WORKFLOW

This section describes the activities involved in the co-simulation workflow, which can be summarized as follows: (i) implementing the coordination algorithm, the control algorithm, and the kinematic model of the robots; (ii) defining the co-simulation architecture; (iii) performing simulation; and (iv) assessing the results and, if needed, revising choices and simulating again.

In this work, co-simulation has been used to model an exploration task performed by robot vehicles, coordinated by the algorithm in Sec. 3.1. The algorithm is executed by a supervisory controller, called *planner* in the following, to visit all regions of the grid. More specifically, the goal of the planner is to have all the cells visited by at least one robot. Co-simulation made it possible to integrate the off-the-shelf robot implementations with the planner implementation.

4.1 Implementation of the exploration algorithm

As described in Sec. 3.1, the exploration algorithm is a very general model of a bio-inspired behavior. When the algorithm is used as a planning strategy for a practical application, appropriate values must be chosen for its parameters. These choices may be based on experimental data, or heuristics, or a combination of both.

Simulation can play an important role in this task, especially as it makes it possible to introduce system features in the model that are necessarily absent in the algorithm. In particular, the physical behavior of the agents (e.g., robot vehicles) is a crucial factor in the algorithm's implementation.

The algorithm is structured in three high-level tasks, executed in sequence for each robot at each simulation step, that operate on a shared data structure representing the map: (a) choose the next cell, according to equation (6); (b) compute evaporation in all cells, according to equation (3); and (c) release pheromone in accessible cells, according to equation (2). Task (a) is further composed of three subtasks, which (i) identify the cell whose center is the current robot position, (ii) find the set of accessible cells, and (iii) choose the next robot position.

4.2 Implementation of the robot controller

Algorithm 1 Robot control 1.

```
compute $\beta$; {Eqn. 7}
if moving then
    if $|\Delta x| + |\Delta y| < \delta_\rho$ then
        switch to maneuver;
        stop robot;
    end if
else if maneuver then
    if $|\beta| < \delta_\beta$ then
        switch to moving;
    end if
end if
```

The control laws in Sec. 3.2 are implemented by Algorithm 1, where δ_ρ is a tolerance on the distance from the waypoint, and δ_β is a tolerance on the alignment error. However, co-simulation shows that in a few cases the algorithm becomes unstable as Δx or Δy are close to zero. The reason is that the control laws assume that the robot position does not change in the maneuver phase, i.e., they assume the robot turns around its center of mass without any translation and so Δx and Δy do not change their sign. Instead, the mechanics of turning of the single-axle vehicle used in the co-simulation causes the center of mass to move, so that Δx or Δy, but not both at the same time, can change their sign, causing instability in the algorithm.

This is an example of how experimental validation complements formal modeling. Splitting the robot movement into clearly separated phases of pure rotation and pure translation is an appealing modeling choice and a reasonable approximation in many cases. But in this case, deviations from the ideal case are noticeable, requiring the control algorithm to be changed. The deviations have been detected by visual inspection of simulation results, and further work could involve the development of an automatic procedure to compare simulation results for the actual robot to results for an ideal one.

This problem is corrected by checking if the above condition on Δx and Δy holds (comparing Δx and Δy to a threshold δ_{thr}), and in that case truncating the value of Δx or Δy to zero. The alignment error β is then computed. Accordingly, Algorithm 2 performs the check and chooses the next control phase, and Algorithm 3 applies the corresponding control law.

4.3 Co-simulation architecture

The co-simulation model is structured in three levels: (i) the exploration planner, (ii) the robot controllers, and (iii) the robot plants. As shown in Fig. 2, each robot is simulated with one FMU for the robot controller

Algorithm 2 Robot control 2a.

```
compute $\beta$; {Eqn. 7}
if moving then
    if $|\Delta x| + |\Delta y| < \delta_\rho$ then
        switch to maneuver;
        stop robot;
    end if
else if maneuver then
    if ($|\Delta x| < \delta_{\text{thr}}$ or $|\Delta y| < \delta_{\text{thr}}$) then
        if $|\Delta x| < \delta_{\text{thr}}$ then
            $\Delta x \leftarrow 0$;
        end if
        if $|\Delta y| < \delta_{\text{thr}}$ then
            $\Delta y \leftarrow 0$;
        end if
    end if
    if $|\beta| < \delta_\beta$ then
        switch to moving;
    end if
end if
```

Algorithm 3 Robot control 2b.

```
if maneuver then
    linear speed $\leftarrow 0$; {Eqn. (8)}
else
    compute linear speed; {Eqn. (9)}
end if
compute turning speed; {Eqn. (8) or (9)}
compute wheel speeds; {Eqn. (10)}
```

(Sec. 4.2) and one for the plant. The latter is an off-the-shelf FMU available from an INTO-CPS example project (The INTO-CPS Association 2019). The FMU was designed to model a small sized two-wheel robot, with parametric axle length and wheel radius. For the co-simulation shown in this section, the default values in the example have been used for all the parameters, with the exception of the initial battery level, which has been considerably increased to avoid an abrupt stop caused by loss of energy. The 20-sim model of the robot is available on the INTO-CPS application GitHub repository.

A central FMU (the *planner*) contains the implementation of the exploration algorithm (Sec. 4.1). The planner FMU receives the current position of each robot that has reached its waypoint, and returns the new waypoint to the robot's controller FMU, closing the higher-level feedback loop. In the low-level loop, the plant FMU provides the controller with the current position and heading, and the controller returns the control linear and turning speeds.

4.4 Graphical Interface

A Graphical User Interface (GUI) has been implemented starting from a pre-existent NodeJS (OpenJS Foundation 2020) project of the authors to visualize a two-dimensional representation of the robot movements as the simulation progresses (Palmieri et al. 2020). The graphical interface communicates with the planner through websockets. The planner sends the updated values to the GUI every co-simulation step.

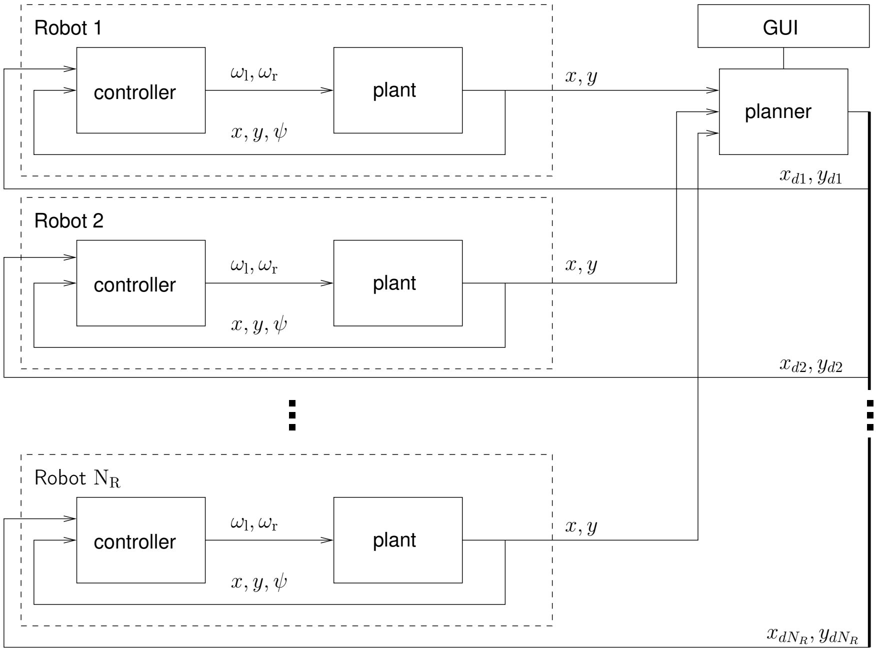
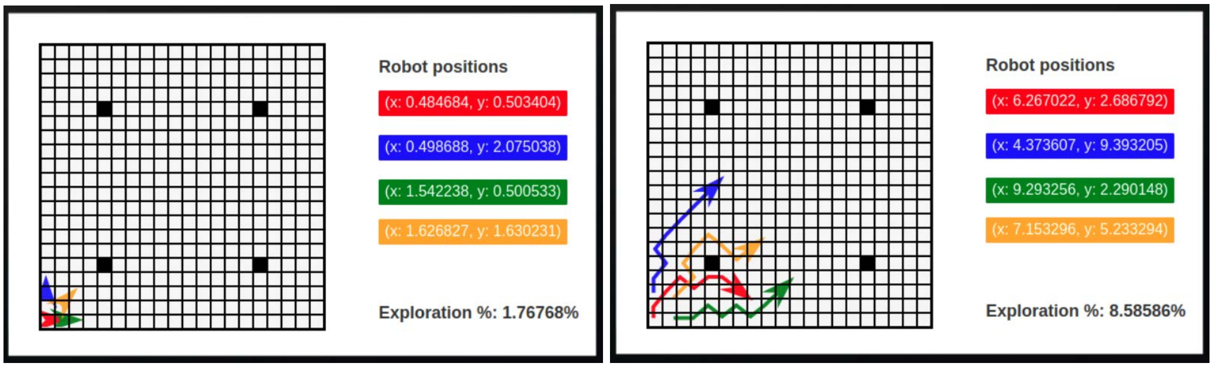


Figure 2: Co-simulation architecture.

The view of the map is displayed in a web page as a square grid, with color-coded lines representing robot paths and black squares representing obstacles. Next to the map, a legend identifies the robots and displays their co-ordinates, together with the percentage of visited cells.

Fig. 3 shows a twelve by twelve map, with four obstacles uniformly distributed around the center of the quadrants and four robots placed in cells located near the left bottom corner. In the left picture, the initial environment is shown; in the right picture two robots that avoid the obstacle are shown. Another part of the interface is a set of forms to enter application-specific data, such as initial values or parameters. This interface hides the native INTO-CPS application interface, which is rather complex, as it is designed to support general-purpose experiments and provide access to the co-simulation infrastructure, in particular to the FMUs and their interconnections. The interface for the bio-inspired exploration algorithm is tailored for the co-simulation architecture shown in Fig. 2, and it relieves the users from the task of specifying the interconnections between the FMUs.



Robot positions

(x: 0.484684, y: 0.503404)

(x: 0.498688, y: 2.075038)

(x: 1.542238, y: 0.500533)

(x: 1.626827, y: 1.630231)

Exploration %: 1.76768%

Robot positions

(x: 6.267022, y: 2.686792)

(x: 4.373607, y: 9.393205)

(x: 9.293256, y: 2.290148)

(x: 7.153296, y: 5.233294)

Exploration %: 8.58586%

Figure 3: Graphical interface: two simulation snapshots.

Table 1: Parameter values from (Palmieri, Yang, Rango, and Marano 2017).

Parameter	Value	Parameter	Value
R_s	4	η	0.9
$\Delta\tau_0$	2	a_1	0.5
φ	1	a_2	0.5
λ	1	ε	uniform in [0, 1]

4.5 Co-simulation Results

The FMUs described above have been deployed on the INTO-CPS application to run simulations aimed at showing the effects of different algorithm parameter values and map dimensions. A first result of the experiments is the validation of the entire system model (the bio-inspired algorithm, the controller algorithm and the kinematics of the robots). Co-simulations show that the robots actually move towards adjacent cells and explore the map. Two series of simulations have been run, one for a map size of twelve by twelve and one for a size of twenty by twenty. In both series, the $\mathrm{ERTU}_{\%}$ parameter varied in the set $\{0.2, 0.002, 0.00002\}$. The other parameters, taken from (Palmieri, Yang, Rango, and Marano 2017), are in Table 1.

Simulations are executed with the DSE tool of the INTO-CPS application. For the smaller map, two cases are simulated, one without obstacles and one with four obstacles arranged as in Fig. 3. For the larger map, only the case with obstacles is reported. Each simulation lasts 20 simulated minutes. The values in the Table 2 have been obtained by generating 10 different runs for each parameter combination (map size, obstacles and $ERTU_{\%}$). At each run, the random number generator has been initialized with a different seed.

The results show that, independently of the map size and the number of obstacles, a low value of exploration rate was achieved for $ERTU_{\%} = 0.2$, while the exploration rate increases for lower values of $ERTU_{\%}$. This is due to the interplay between the assumptions in the configuration of the exploration algorithm and those in the robot kinematics. In fact, a high value of $ERTU_{\%}$ means that, during the movement of a robot, the pheromone levels in the neighboring cells vanish quickly, so that the robots soon loose the pheromone trail and begin to move randomly.

The value of 0.2 was taken from (Palmieri, Yang, Rango, and Marano 2017), where the simulation's goal was studying energy consumption, so that the robot speed was not included in the physical model. In the present work, instead, the physical model of the robot is included, and the relationship between evaporation rate and robot speed significantly affects the simulation. In this case, the experiments showed that an $\mathrm{ERTU}_{\%}$ value of 0.002 is appropriate for the physical characteristics of the chosen robots, because the average exploration percentage significantly increases with respect to 0.2.

We may recall that the robot kinematics were embodied in an off-the-shelf FMU that was not meant to be modified, a common constraint in real-world development. Reuse of off-the-shelf components is a fundamental issue in all industrial development processes. While the concept of reuse has always been implicit in engineering practice, software engineering is arguably the field where it has been studied more extensively, leading to the definition of such concepts as *software modules* and *components*, and *design patterns*. In this framework, software reuse is based on *interface* compatibility between components, and to methods to verify and validate interface compatibility.

Taking these ideas and methods back to the field of systems engineering requires carrying them from software components to more general models. In particular, simulation is analogue to software integration, as it assists in revealing interface-related issues. This approach has proved useful in the present work, but deeper issues remain to be dealt with. First, the very concept of interface must be adapted to cyber-physical systems, where continuous signal flows, as opposed to discrete messages or operation calls, are the rule. Then, languages to formally define the interfaces must be devised. Last (and not least), theories must be

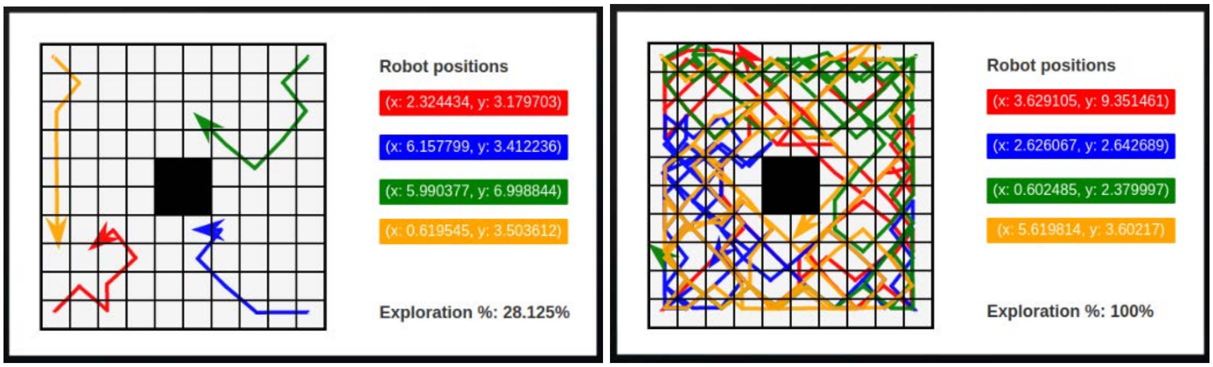
Table 2: DSE results.

			Exploration rate (%)		
map size	obstacles	ERTU%	Average	Minimum	Maximum
12 x 12	0	0.2	81.04	70.83	92.36
12 x 12	0	0.002	96.45	91.66	98.61
12 x 12	0	0.00002	97.56	94.44	100
12 x 12	4	0.2	80.29	71.43	90.71
12 x 12	4	0.002	96.21	88.57	99.29
12 x 12	4	0.00002	97.00	91.43	99.29
20 x 20	4	0.2	37.93	29.29	45.96
20 x 20	4	0.002	66.21	60.10	71.46
20 x 20	4	0.00002	72.95	70.45	75.51

developed for the analysis of systems defined in terms of component interfaces, upon which analysis and synthesis tools may be built.

Co-simulation of the exploration algorithm with the physical model allows better calibration of the $\mathrm{ERTU}_{\%}$ parameter. Another situation where co-simulation has allowed us to discover possible instability issues due to different assumptions in different submodels has been discussed in Sec. 4.2. More precisely, when Δx and Δy approach zero, the control law (control subsystem) assumes that the robot is not moving during the maneuvering phase, while the mechanics (plant subsystem) can make small motions of the robot's center of mass, with positive or negative signs, thus leading to instability of the algorithm.

Simulations had shown that a 100% exploration rate is reached in the case of no obstacles. Another experiment achieved 100% exploration rate in the presence of obstacles, at the cost of longer operation time. The map is ten by ten, four obstacles are placed at the center of the map, and in the initial condition each robot is located at a different corner of the map. The $\mathrm{ERTU}_{\%} = 0.002$ and the simulation lasts 60 simulated minutes. Fig. 4 shows two screenshots: the first steps on the left, and the complete paths on the right.



Robot positions

(x: 2.324434, y: 3.179703)

(x: 6.157799, y: 3.412236)

(x: 5.990377, y: 6.998844)

(x: 0.619545, y: 3.503612)

Exploration %: 28.125%

Robot positions

(x: 3.629105, y: 9.351461)

(x: 2.626067, y: 2.642689)

(x: 0.602485, y: 2.379997)

(x: 5.619814, y: 3.60217)

Exploration %: 100%

Figure 4: Another experiment with 100% exploration coverage.

5 CONCLUSIONS

In this paper a bio-inspired exploration algorithm, which coordinates multiple robots, has been co-simulated together with the physical model of the robots. Co-simulation allows subsystems modeled with different tools to be simulated. The algorithm and the controller of the robot have been written in C, while the kinematics of the robot is modeled in 20-sim. One of the advantages of co-simulation is that complex systems made up of heterogeneous subsystems can be modeled and simulated in a straightforward way, building each respective submodel with its specific language. This applies to many weakly-coupled systems such as the one considered in this work, which does not address strongly-coupled ones, e.g., exhibiting multi-body interactions, which require an accurate choice of time-stepping algorithms (Lacoursière and Härdin 2017).

This work also shows that co-simulation provides developers with the means to investigate the implied assumptions underlying the various submodels, and detect problems arising from subtle semantic mismatches, as discussed in Sec.s 4.2 and 4.5. DSE tools currently support semi-automatic exploration of design space or input space subsets that developers consider useful to reveal such mismatches. This is particularly important with complex CPSs whose submodels are developed independently and at different levels of abstraction.

Many lines of further research are open, such as the issues mentioned in Sec. 4.2, on detecting deviations from expected behaviors, and in Sec. 4.5, on defining interfaces for subsystem models and analyze the results of their composition. The authors propose to tackle these issues by the integration of formal verification and simulation (Bernardeschi et al. 2019, Domenici et al. 2018).

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their useful comments and suggestions. Work partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

REFERENCES

Alfeo, A. L., M. G. Cimino, and G. Vaglini. 2019. "Enhancing biologically inspired swarm behavior: Metaheuristics to foster the optimization of UAVs coordination in target search". *Computers & Operations Research* vol. 110, pp. 34 – 47.

The INTO-CPS Association 2019. "case-study_line_follower_robot". https://github.com/INTO-CPS-Association/Documentation/blob/master/examples_compendium/INTO-CPS_Examples_Compendium.pdf.

Bernardeschi, C., A. Domenici, and P. Masci. 2018. "A PVS-Simulink Integrated Environment for Model-Based Analysis of Cyber-Physical Systems". *IEEE Transactions on Software Engineering* vol. 44 (6), pp. 512–533.

Bernardeschi, C., A. Domenici, and S. Saponara. 2019. "Formal Verification in the Loop to Enhance Verification of Safety-Critical Cyber-physical Systems". *Electronic Communications of the EASST* vol. 77, pp. 1–9.

Blochwitz, T., M. Otter, J. Åkesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel. 2012. "Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models". In *Proceedings of the 9th International Modelica Conference*, pp. 173–184, The Modelica Association.

Chen, X., Y. Kong, X. Fang, and Q. Wu. 2013. "A fast two-stage ACO algorithm for robotic path planning". *Neural Computing and Applications* vol. 22 (2), pp. 313–319.

De Rango, F., N. Palmieri, X. S. Yang, and S. Marano. 2015. "Bio-inspired exploring and recruiting tasks in a team of distributed robots over mined regions". In *2015 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pp. 1–8.

Domenici, A., A. Fagiolini, and M. Palmieri. 2018. "Integrated Simulation and Formal Verification of a Simple Autonomous Vehicle". In *Software Engineering and Formal Methods*, edited by A. Cerone and M. Roveri, Volume 10729 of *Lecture Notes in Computer Science*, pp. 300–314.

Dorigo, M., M. Birattari, and T. Stutzle. 2006. “Ant colony optimization”. *IEEE Computational Intelligence Magazine* vol. 1 (4), pp. 28–39.

OpenJS Foundation 2020. “Node.js web site”. https://nodejs.org.

Gomes, C., C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe. 2018. “Co-simulation: a survey”. *ACM Computing Surveys (CSUR)* vol. 51 (3), pp. 1–33.

Karaboga, D., and B. Akay. 2009. “A comparative study of Artificial Bee Colony algorithm”. *Applied Mathematics and Computation* vol. 214 (1), pp. 108 – 132.

Lacoursière, C., and T. Härdin. 2017. “FMI Go! A simulation runtime environment with a client server architecture over multiple protocols”. In *Proceedings of the 12th International Modelica Conference*, pp. 653–662.

Larsen, P. G., H. D. Macedo, M. Neghina, C. König, S. Basagiannis, J. Woodcock, C. Gomes, and John-Fitzgerald. 2018. “The INtegrated TOolchain for Cyber-Physical Systems (INTO-CPS): a Guide”. Technical report, INTO-CPS Association.

Masár, M. 2013. “A biologically inspired swarm robot coordination algorithm for exploration and surveillance”. In *IEEE 17th International Conference on Intelligent Engineering Systems*, pp. 271–275.

Palesi, M., and T. Givargis. 2002. “Multi-Objective Design Space Exploration Using Genetic Algorithms”. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign*, CODES ’02, pp. 67–72. New York, NY, USA, Association for Computing Machinery.

Palmieri, M., C. Bernardeschi, and P. Masci. 2017. “Co-simulation of Semi-autonomous Systems: The Line Follower Robot Case Study”. In *Software Engineering and Formal Methods - SEFM 2017 Collocated Workshops, Revised Selected Papers*, pp. 423–437.

Palmieri, M., C. Bernardeschi, and P. Masci. 2020. “A framework for FMI-based co-simulation of human-machine interfaces”. *Software and Systems Modeling* vol. 19 (3), pp. 601–623.

Palmieri, N., X.-S. Yang, F. D. Rango, and S. Marano. 2017. “Comparison of bio-inspired algorithms applied to the coordination of mobile robots considering the energy consumption”. *Neural Computing and Applications* vol. 31, pp. 263–286.

Pugh, J., A. Martinoli, and Y. Zhang. 2005. “Particle swarm optimization for unsupervised robotic learning”. In *Proceedings IEEE Swarm Intelligence Symposium, SIS 2005*, pp. 92–99.

Siciliano, B., and O. Khatib. 2016. *Springer handbook of robotics*. Springer.

AUTHOR BIOGRAPHIES

CINZIA BERNARDESCHI is an Associate Professor with the Department of Information Engineering at the University of Pisa. Her main research interests are in the area of dependable systems and formal methods for verification of safety-critical systems. Her email address is cinzia.bernardeschi@unipi.it.

ANDREA DOMENICI is Assistant Professor with the Department of Information Engineering at the University of Pisa. He is working on theorem-proving and CPSs. His email address is andrea.domenici@unipi.it.

ADRIANO FAGIOLINI is Assistant Professor at the University of Palermo, Italy. His research interests are in estimation and control for mobile and distributed robotics . His email address is adriano.fagiolini@unipa.it.

MAURIZIO PALMIERI is a researcher with the Department of Information Engineering at the University of Pisa. His research interests include application of formal methods and co-simulation techniques to the analysis of cyber-physical systems. His email address is maurizio.palmieri@ing.unipi.it.