Smart Self-Adaptive Cyber-Physical Systems: How can Exploration and Learning Improve Performance in a Partially Observable Multi-Agent Context?

Ana Petrovska, Malte Neuss, Sebastian Bergemann, Martin Büchner, M. Ansab Shohab
Department of Informatics
Technical University of Munich
email: ana.petrovska, malte.neuss, sebastian.bergemann, martin.buechner, ansab.shohab@tum.de

***Abstract*—Cyber-physical systems (CPSs) are software-intensive systems that are embedded in the physical world to monitor, control and coordinate a variety of processes in both the physical and the digital world. As a result, they often operate in complex, dynamic, and unanticipated environments with various potential sources of run-time changes and uncertainties, that could potentially lead the CPSs to faults, and even to complete system failures. To cope with these changes, the systems should have the capabilities to self-adapt in order to continue meeting their functional specifications. In this paper, we investigate how creating self-adaptive CPSs which are able to collaborate and learn in a dynamic, partially observable, multi-agent context, can not only preserve but also improve the performance, despite all the changes introduced to the system at run-time. We evaluate the proposed methodology on an in-house developed, multi-agent system from the robotics domain.**

Keywords*—*self-adaptive systems, cyber-physical systems, collaboration, learning, partial observability

I. Introduction

In recent years, the widespread availability of cost-effective embedded systems with increasing computation power and the expansion of wireless networks have led to a solid foundation for emergence and advancement of the pervasive Cyber-Physical Systems (CPSs) in a multitude of different domains, with progressively increasing technological and social influence. Modern CPSs, which lie in the intersection of the control, computation and communication area [1], are composed of many interacting and interconnected components, while inheriting all the complexities of large-scale distributed systems [2]. Also, they need to be able to operate efficiently and reliably within a continually changing, uncertain, and unanticipated environments or *execution contexts* [3] [4] [5]. Furthermore, they need to be able to collaborate and cooperate with another CPSs towards realizing common goals, which a single system or agent itself would not be able to achieve on its own. To successfully cope with the change introduced at run-time (and cannot be predicted during the design of the system), these systems should be therefore engineered with properties to learn, and to automatically and independently modify themselves without any external human involvement [6] [7] . The run-time changes can originate from 1) the CPSs themselves and 2) from the context where these agents are operating.

A. Motivation

The need for self-adaptive systems stems from the ideas initially introduced in "The Vision of Autonomic Computing" [8] by Kephart and Chess, where the authors envision the systems from the future to only manage themselves accordingly to high-level business goals given by human administrators. In a future world, where the present-days engineers and developers become obsolete, systems organize and manage themselves in a completely autonomous manner. These ideas, anticipating fully autonomous self-engineering and self-managing systems, still remain "ideas that are not science fiction, but elements of the grand challenge [8]". From a current time-point, it is impossible to argue on how the systems from the future will be engineered, but instead, continuous step-by-step integration of the contemporary concepts and ideas is necessary. Consequently, self-adaptive systems can be considered as an intermediate step toward complete autonomicity.

On a conceptual level a self-adaptive system, is comprised of a *managed element* and *adaptation logic*, as shown in Figure 1. The managed element is the entity that acquires self-adaptation capabilities, given by the adaptation logic. A common approach to realize the adaptation logic of a self-adaptive system is through the MAPE-K (Monitor, Analyze, Plan, Execute) [8] feedback loop, with shared Knowledge among all the components of the loop. The self-adaptive system interacts with the context, which is the relevant part of the environment, or the external world, for that particular system.

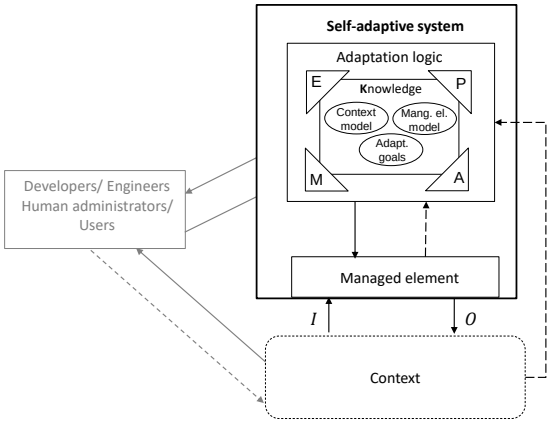

Figure 1. Updated conceptual model of a self-adaptive system from [4].

The internal changes in the managed element(s)—in our case CPS(s); and/or changes in the context during *run-time*, are triggers for the system to self-adapt. Additionally, it is

 ISBN: 978-1-61208-848-8

essential for the system to know why it is adapting for, or what the adaptation goals are. As a result, as shown in the figure, the knowledge in the adaptation logic presents an abstraction of relevant aspects of the managed element(s), the context and the system's adaptation goals.

B. Background

Russel and Norvig in [9] categorize environments, or contexts (as we refer to in this work) accordingly to a few informally defined dimensions: fully observable vs. partially observable, single-agent vs. multiagent, deterministic vs. stochastic, episodic vs. sequential, discrete vs. continuous and known vs. unknown. Unknown context does not refer to the context itself, but it refers to the robots' knowledge about *the laws of physics* of the context [9]. These defined dimensions, to a large extent, determine the appropriate system design and implementation. According to the authors, the hardest case is designing and implementing solutions for systems operating in partially observable, multiagent, stochastic, sequential, dynamic, continuous and unknown [9], or abbreviated, PMSSDCU context. This exactly how we would classify the context in which our multi-agent systems are operating. In this paper, we propose a methodology that provides an engineering solution for self-adaptive multi-agent CPSs operating in PMSSDCU context.

C. Gaps and Contributions

The majority of the previous works in the self-adaptive systems community provide approaches where 1) the adaptation logic is predetermined and its structure does not change over time, e.g., [10], or 2) the operational context in which the self-adaptive CPSs operate is predetermined and static, and does not change during run-time, e.g., [11]. Having an adaptation logic that is predefined at the design of the system and does not improve over time, cannot provide adequate and accurate adaptation, when the self-adaptive systems and the context in which they operating are dynamic and changing in an unpredictable manner during run-time. As a result, the adaptation logic should have mechanisms to modify itself in order to reflect the run-time changes in the context where the agents are operating. In this paper, we tackle this issue by proposing a methodology for building adaptation logic for self-adaptive CPSs that operate in a dynamic, partially observable, multi-agent context. Precisely, we focus on building a self-adaptive system, for multi-agent CPSs, with shared adaptation logic, in which the knowledge in the adaptation is continuously updated at run-time. In our work, the adaptation logic does not only adapts the behaviour of the systems (the managed elements), but it changes its own knowledge during run-time. The contributions of the paper are the following:

1) We propose an approach for modeling the context in the knowledge of the adaptation logic, based on globally aggregated observations of from all the agents. It is based on learning two probabilistic maps by storing the past contextual encounters, which enable the agents to over time gain knowledge about *the laws of physics* of the context.
2) For multi-agent tasks allocation, which provides close-to-optimal solution, is computationally feasible, and is dynamically adaptable during run-time, we apply *Prim Allocation* algorithm using minimum spanning forests (MSF), proposed in [12].
3) We propose local path planning that minimizes the distance to the assigned task and maximize the context exploration.
4) Additionally, for evaluating the ideas, in this paper, we have developed an in-house, ROS-based, multi-agent simulated system from the robotics domain. The robotic system is based on a reference problem proposed in the following section, which explains and motivates the need for self-adaptivity.

The paper is organized as follows: Section II explains the reference problem and motivates the need for self-adaptation. The reference problem is additionally used as a running example throughout the paper. Section III elaborates in more depth the challenges that we are addressing in this work. The three-step methodology is proposed in Section IV. The details of the implementation of the multi-agent system are presented in Section V. Subsequently, the benefits of collaboration, exploration, and learning are evaluated in Section VI. In Section VII we conclude the paper.

II. Reference problem

Our reference problem comes from the robotics domain, and aims to motivate and support the need for self-adaptivity. The reference problem is used as a running example in the paper. Additionally, based on the reference problem, we have built the robotic system presented in Section V.

The setting of our reference problem consists of the following: (1) the context: a room with static obstacles (for example, walls and interior) where dirt appears perpetually in different places at different points in time; and (2) agents—CPSs: autonomous, ground robots. The robots need to explore and detect dirt tasks in the room, and attain them in the most *efficient* way (in the shortest period of time) despite different run-time uncertainties [13], including the limited sensor range (see Figure 2). Namely, the sensors of the robots have a limited sensing range and they observe the context or the room where they are deployed only partially. Consequently, the agents can detect dirt that is only within their range of observations. Once the dirt locations are detected and identified, they become *goals* for the robots. Additionally, each robot is equipped with a map of the room, and they use Adaptive Monte Carlo Localization (AMCL) for navigation and localization.

In our reference problem, the robots monitor or observe the context, and discover new tasks in a distributed manner. There is no global view of the room, meaning that the robots only discover tasks which are within their range of observation. In this case, the agents not knowing what is happening in the local surroundings of the other agents, brings inefficiency to the overall performance, for example, when one

 ISBN: 978-1-61208-848-8

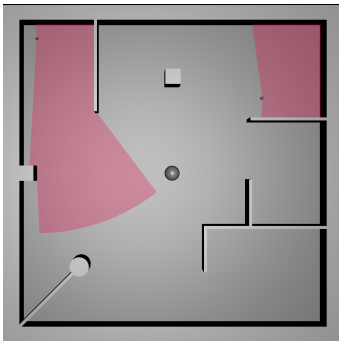

Figure 2. View of the room and the deployed agents.

part of the room is getting dirtier than the other. The effects of having a partially observable context, is that as a result only a local performance maximum is possible. To achieve global performance maximum, a cooperative aggregation of the contextual observations of all the robots deployed in the room is necessary.

A self-adaptive system has *business* or *mission goals*, related to the functional requirements of the agents; and *self-adaptation* goals related to the quality objectives or the non-functional requirements [6]. The business goal of the system in our reference problem is keeping the room clean by first detecting, and then removing the dirt. The self-adaptation goals are the following: 1) increasing the performance by minimize the time needed for the room to be cleaned and be kept clean, and 2) increasing its fault-tolerance by avoiding failures (for example, collision with other robots) and deadlocks. The self-adaptation goals need to be satisfied despite the internal or the external (contextual) changes and uncertainties that emerge at run-time. In our specific reference problem the following changes trigger the need for self-adaptation:

Internal changes: Imperfect sensors. As explained above, the sensors have restricted range of observation; therefore, the agents have only a partial view of the physical space or the room in which they operate. Furthermore, they can detect new dirt task only if it appears within their range of observation. Additionally, there are sensor uncertainties that originate from hardware and software limitations of the sensors, for example, sensor imprecision, noise, ambiguity, inconsistency and inaccuracy, and even sensor failures [13]. This means that even if a robot observes dirt (subsequently referred to as task) within the range of its sensor, we cannot be 100% certain in the accuracy and the precision of the observation.

Context-changes: Multiple agents operating in the same room. For CPSs, it is highly probable and more realistic scenario to have multiple agents deployed in a relative proximity, for instance, a platoon of autonomous cars or a fleet of robots. In our example, when the agents need to localize themselves and navigate in a room, the other agents deployed in their relative proximity indirectly influence their actions. This can potentially lead to different AMCL localization and navigation issues, which can later result as sources of failures. For example, collisions or deadlocks that directly impact the overall system performance.

Context-changes: Continuous appearance of new dirt. As previously explained in Section I-B the agents do not have knowledge about *the laws of physics* of the context. In our case that would mean that when new tasks are continuously spawned in the room, they will be spawned at random locations, with location patterns unknown to the robots in advance. The run-time decisions on *how* the new tasks are assigned to the agents, and *what* path the robots take to reach to those tasks can significantly influence the system performance.

To sum up, our reference problem introduces and identifies *run-time* changes and uncertainties that are characteristic of a real multi-agent robotic systems. These changes and uncertainties trigger the self-adaptation, and cannot be specified beforehand during design time of the system. However, they need to be dealt with during the run-time, without affecting the system performance and system's functional goals, as well as the quality objectives.

III. Challenges

In this work, we make our contributions by addressing the following challenges and the corresponding emerging questions:

Challenge 1: *Distributed observation and reliable detection of continuously appearing tasks in a partially observable context, and learning the context by collaboratively building aggregated context models in the knowledge of the adaptation logic based on the previous observations.*

All the agents deployed in the room observe the context in a distributed manner. As explained in the previous section, the CPSs have limited sensor range, and therefore, they make only partial observations of the room. Consequently, when a new task or dirt is being spawned, it can only be detected once it is within the range of observation of at least one of the agents. Additionally, there are other run-time sensor uncertainties, like sensor imprecision, and sensor ambiguities, which imply that we cannot be fully sure in the true position of a task, even when a task is detected by the agents. Furthermore, if there is no mechanism for the robots to share their observations with each other, then achieving a global performance maximum is not possible due to the likelihood of one part of the room getting dirtier than the other. Hypothetically, developing more complex adaptation logic—through collaboratively built knowledge—based on globally aggregated context models built jointly by all the agents, could enables us to achieve a global maximum of the performance of the overall system (in combination with close-to-optimal solutions from the other challenges).

In our reference problem, learning the context (the room) would mean storing the past context states, which potentially lead towards learning the patterns in which the tasks appear in the context. The built knowledge of the context in the adaptation logic can be considered as an input to the local path planning (further explained in Challenge 3), which, for example, would enable the CPSs to choose paths with higher

 ISBN: 978-1-61208-848-8

probabilities of new tasks appearing, over paths with lower probabilities.

Question 1: How to ensure reliable task detection?

Question 2: How to build global aggregated context models from what the robots are independently observing from the context?

Challenge 2: *Global multi-agent task allocation that has a close-to-optimal solution, which is computationally feasible, and dynamically adapts during run-time.*

Once the dirt tasks are detected, they become goals and they need to be assigned to the CPSs in the most optimal way. The requirements are the following: 1) the algorithm should suffice optimality criteria and 2) it should exhibit high efficiency concerning distance and time travelled. Since we need to ensure a true real-time capability of the goal allocation (new task gets detected, or assigned goal is reached), the algorithm should be dynamically adaptable during run-time. The problem of computational feasibility needs to be considered, since we cannot assume the complexity of the environment and the number of dirt locations beforehand, and they can increase during run-time. Also, the notion of completeness is also essential since we need to consider all possible goal locations detected, and finally, find a suitable path to reach the locations. Thus, the algorithm should terminate with a solution when one exists. According to Lagoudakis et al. [12] finding an allocation of goals to multiple agents, as in our setting: identical robots, symmetric and uniform traversing costs, operating on the Euclidean plane; where the total cost (the sum of the travel costs of all robots over time) of all the paths that the robots traverse to their goals is minimized, is an NP-hard problem, because it is the multi-agent version of the Euclidean Traveling Salesman Problem [14].

Question 3: How to allocate continuously appearing tasks, as goals to many agents in a close-to-optimal, computationally feasible and dynamically adaptable during run-time way?

Challenge 3: *Local path planning to the assigned goal that minimizes the traversed distance and maximizes the context exploration.*

Once the goals (the explored tasks) are assigned to the CPSs, the agents should plan how to reach to the locations of their goals in a way that maximizes the space exploration while traversing paths that have the highest probability of new tasks appearing. Maximizing the space exploration is essential, due to the partial observability of the context, as explained in Challenge 1. Having partially observable context might lead to situations where the agents have enough tasks to complete in the parts of the room that they have already observed and explored; but maybe the other unobserved parts of the room get dirtier with a higher rate. As a result, in this case, exploring and accomplishing tasks the unexplored parts of the room brings better the performance-time ratio. Consequently, the robots should be incentivized to explore more space.

Additionally, as previously explained in Challenge 1, the knowledge from the past context situations should be taken into consideration during the local planning towards the currently assigned goal. Namely, even if the robots do not observe any tasks on a particular path, but the knowledge tells that there is a high probability of new dirt a appearing, then this path should be preferred over the others.

Question 4: What local path planning can minimize the distance to the assigned task and maximize the context exploration?

IV. Methodology

Our methodology for engineering the adaptation logic in self-adaptive CPSs that operate in PMSSDCU context is shown on Figure 3. In our proposed solution, we address the challenges described in the previous section, in three different phases: two local (i.e., decentralized and distributed in every CPSs), and a global phase shared among all the CPS, e.g.., robots. Every phase in the proposed solution is performed by the self-adaptive CPS autonomously. In the following subsections, we address the three challenges, respectively.

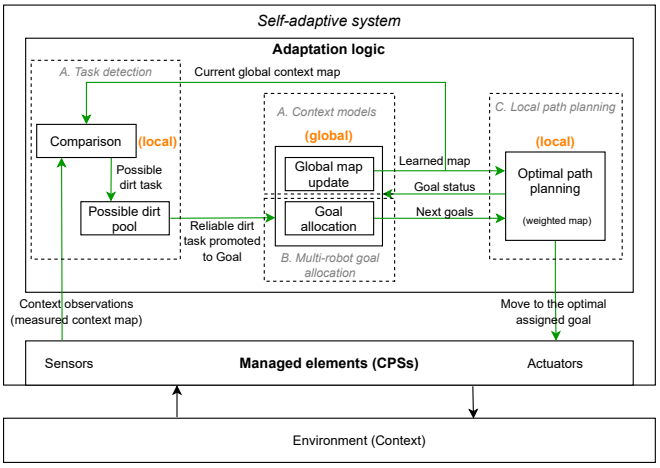

Figure 3. General overview of our methodology.

A. Tasks detection, knowledge representation and context models (Addressing Challenge 1.)

1) Reliable tasks detection: Once new dirt or task is sensed it is considered as a new *potential* goal object. In our implementation, the new potential goal object is stored in a list, with a unique identifier, a pose (containing the x, y coordinates of the detected task in the map), and an initial confidence value. We chose the initial confidence value for every new goal object stored in the list to be 10%. If the same task is detected again (with a small position tolerance, due to sensor uncertainties), by the same or the other agents in the room, then the confidence value is increased by 10% more. The potential goal object is published as a reliable goal, once the confidence value exceeds 90%.

2) Context models and knowledge representation in the adaptation logic: In Section I-A, we explained that the knowledge in the adaptation logic presents an abstraction of several relevant aspects, including the context where the system is operating. We model the context as a global, centralized grid map with a size equal to the size of the room. Each cell in the grid is either free or occupied. The occupied cells are occupied either by static obstacles, for example, the walls; or by dynamic obstacles: the agents deployed in the space, or the continuously appearing dirt or tasks. Additionally, in our solution, each cell can hold multiple tasks. The motion of the robots is discretized, and with each time-stamp the robots can move *up*, *down*, *left* and *right* to the centre of their neighbouring cells. As mentioned before, the operation of the robots is limited to their sensor capabilities, meaning that the agents only hold a partial observation of the context at a given point in time. As the robots move in the room, they gather their partial observations in the centralized grid map, where the multiple observations from the robots are aggregated to produce a new, common, global knowledge about the context in the adaptation logic. The global aggregated knowledge contains all the tasks detected by the partial observations of all the agents that are operating in the room.

3) Updating context models based on probabilistic models: In the following section, we explain how we build the knowledge and update the context models during runtime, based on two probabilistic maps. In order to minimize the time taken for detection and completing the tasks, a probabilistic analysis of the environment is necessary. Using this, a mechanism for predicting where the next dirt patch is most likely to appear will be developed. Such predictions will help in minimizing the time needed to clean the room, thus improving the efficiency of the system. The approach for carrying out a probabilistic analysis of the room consists of maintaining two probability maps, which we call *Probability Map* and *Cumulative Map*.

Probability Map. The Probability Map associates with every cell of the grid-map a value quantifying the probability $P_{i,j}$ of dirt appearing in that location in the next time step. It is updated accordingly to Algorithm 1, where $N_{i,j}$ is the number of tasks in a respective cell ij, T is the number of time-steps until a specific point in time, and Δt is the frequency in which new tasks appear. $\frac{N_{i,j}}{T}$ is the division of the number of dirt tasks found since $t = 0$ with the number of time steps until that point in time, resulting in the probability for a specific cell. In case no dirt task has appeared in a specific cell, the expected value for that cell is calculated, and we check whether its value is less than 1 or not. The calculation performed under the else statement is meant to reduce the probability but to never let it reach zero.

Cumulative Map. The Cumulative Map is calculated by making use of the Probability Map. It measures the probability $CP_{i,j}(T)$ that there is at least one task in a specific cell, and it is calculated by the following equation:

$$CP_{i,j}(T) = 1 - (1 - P_{i,j}(T-1))(1 - CP_{i,j}(T-1))$$

Algorithm 1 Update Probability Map

For every cell ij:
if $N_{i,j} = 0$ **then**
  if $\sum_{t=0}^{T} P_{i,j}(t) \cdot \Delta t < 1$ **then**
    $No\,Change$
  else
    $P_{i,j}(T) = \frac{P_{i,j}(T-1)}{2}$
  end if
else
  $P_{i,j}(T) = \frac{N_{i,j}}{T}$
end if

where $(1 - P_{i,j}(T-1))(1 - CP_{i,j}(T-1))$ calculates the probability that there is not a single of task in a cell.

B. Multi-robot goal allocation (Addressing Challenge 2.)

For the multi-robot goal allocation, we need to find a goal-allocation algorithm, which apart from the fact that it can be centralized (and the means and the cost of communication are neglected), we can undoubtedly say that it needs to provide a solution to a problem of utmost complexity. Namely, in our approach, to minimize the overall sum of travel costs of all robots when visiting all detected targets (the identified goals) where finding an optimal allocation is an NP-hard problem, we employ a greedy principle termed *Prim Allocation*. This auction-based approach, derived from operations research and adapted to a multi-agent context, provides the following guarantee on the quality of its allocations [12]: in the worst-case, the total cost of this principle is at most twice the cost of the optimal solution, but in average-case it is close to the optimal solution.

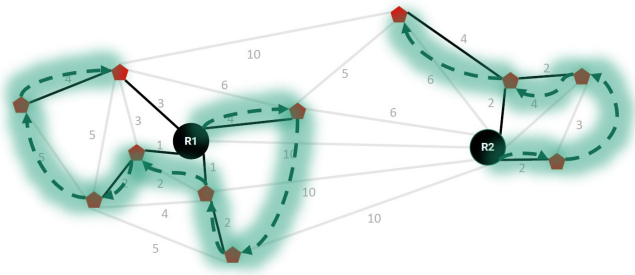

Figure 4. Minimum Spanning Forests

The pseudo-code is given in Algorithm 2, and in a nutshell, it works as follows: an interconnected graph between all the tasks (shown in red pentagons) is woven, and the robot locations in close vicinity to this graph are determined. Then the algorithm finds the shortest connecting link to a tree, initially starting only contains the robot itself. The shortest links are then pair-wise compared and the minimal cost-link is returned and added to the particular tree. This step is repeated until all tasks or dirt locations are assigned. According to this principle, the *minimum-spanning trees* that grow together form *minimum-spanning forests*, as shown in Figure 4. Finally, the

last step is to determine optimal trajectories to the previously allocated goals by the *Prim Allocation*, per agent. In our case, we use a depth-first search algorithm for finding the paths from the sub-trees (marked with green-dashed directed arrows in Figure 4).

Algorithm 2 Prim Allocation from [12]

1) For each robot i, construct a tree T_i that contains only the corresponding robot vertex from V_R
2) While ($V_T \neq \emptyset$) do
   a) For all i, c_i = $min_{v \in V_T}\ min_{w \in T_i}$ {c(v,w)}
   b) j = $argmin_i\ c_i$
   c) v_j = $min_{v \in V_T}\ min_{w \in T_j}${c(v,w)}
   d) Attach v_j to T_j
   e) V_T = V_T - $\{v_j\}$
3) For all i, use the MSF heuristic on T_i to construct the path for robot i.

C. Local path planning (Addressing Challenge 3.)

We model the explorational aspect of local planning as a local optimization problem. The local path planning tries to find the optimal path between the current position of the robot and its next assigned goal. The optimality depends on two factors: 1) minimization of the distance traveled by the robot to the allocated goal, and 2) maximization of the context exploration. For the local path planning we use the probabilistic models previously explained in Section IV-A.

Similarly as the grid-map, the vision of the robot is also discretized as shown in Figure 5. The red circle represents the range of the sensors of the robot and the area in which the robot can detect new task, and the shaded grid is the discretized region corresponding to this area. Using the discretization of the motion of the robot, mentioned previously in the paper, one can construct a search tree (shown in Figure 6) that iterates through the robot's possible actions: starting from the robot's current position until the robot reaches its next assigned goal, looking for the optimal path. For the path search in this case we use uniform cost search.

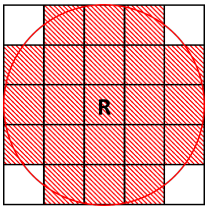

Figure 5. Robot's observation range

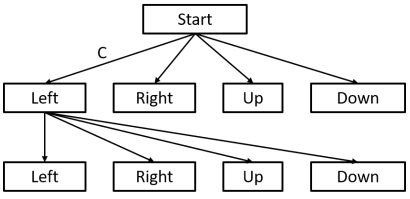

Figure 6. One possible representation of the search tree.

The costs for each transition C is calculated with the following formula:

$$C = d(action) - \alpha * E(x, y, action, t)$$

where $d(action)$ calculates the Euclidean distance of performing a certain action, for example, left right, up and down; and $E(x, y, action)$ is a function that calculates the exploration gain based upon the current position (x, y) and the action taken.

We further introduce a constant α which we can use to empirically tune the relative magnitudes of the Euclidean distance and exploration gain. This allows us to determine the relative importance of the two factors and hereby the path taken by the robot.

For quantifying the exploration gain we use the following formula:

$$E(x, y, action, t) = \sum_{i,j \epsilon S} CP_{ij}(t)$$

which sums all the cumulative probabilities of a set S, where S is the set of currently unseen grid cells that will become visible when a specific action is taken.

V. Implementation

In this section we discuss the implementation of the ROS-based, multi-agent system based on the reference problem, which was previously explained in Section II. We have created simulated, yet realistic implementation of a multi-robot system, which itself presents a challenge. In our implementation, the entire communication is based on Robot Operating System (ROS), and Gazebo [15] [16] is used for simulating the robotics system. Gazebo relies on well-established physics engines, which enables high physical, functional and visual fidelity. In this paper, we evaluate all the concepts considering only two robots, in particular two *Turtebots 3 Burger* [17]. However, our implementation allows increasing the number of robots deployed in the room. Additionally, we simulate 360 degrees 2D LIDAR sensor is mounted on top of the robots. The laser scanners can detect obstacles up to a distance of 3.5 m. [18] and [19] contain the source code of the implementation, together with installation instructions, more detailed architecture of the implementation for each of the sub-tasks, the complete ROS computation graph and illustrative concepts—videos of the implementation and some of the results. The robotic system can serve as a basis for various experiments for other researchers, and can be modified accordingly to their distinct scientific needs.

VI. Evaluation

In this section we show some of the preliminary results. For data collection and analysis, a series of *rosbag*-records were performed. *rosbag*-records subscribe to topics and enable recording of the content of all the messages published on those topics. We have conducted one long-term experiment of approximately 40 minutes, and seven shorter 10-minute experiments. During all the experiments, the exploration parameter α (explained in Section 4.3), the time-interval of dirt spawned Δt, and the use of prior learned knowledge gained in time T are varied. Specifically, the prior learned knowledge comes in the form of probability task distribution that is learned for 1000 time-steps before the actual measurements are collected. Furthermore, the start-time is used to denote the recordings. The parameter specifics are given in the Table 7.

 ISBN: 978-1-61208-848-8

Sample	α	Δt	start-time	knowledge
LONG TERM				
#1	0.0	10-10	15-49-49	FALSE
EXPLORATION				
#1	0.75	10-10	17-15-00	FALSE
#2	0.75	25-25	17-27-12	FALSE
#3	0	25-25	17-39-28	FALSE
#4	0.75	25-25	17-58-57	TRUE
#5	0.75	25-25	19-12-42	FALSE
#6	0.75	15-15	19-31-39	FALSE
#7	0.75	10-10	14-09-16	TRUE

Figure 7. Experiments parameters specifics.

It is important to point out that for testing purposes and better replication of the scenarios in the experiments, we have fixed the frequency Δt and used a random seed for the appearance of the tasks. Additionally, the multi-robot global task allocation node runs at $1Hz$—detection and allocation of new goals are re-calculated every second. In the following, we are showing results for three different cases:

1) no exploration $\alpha = 0$, and no prior knowledge $T = 0$;
2) exploration $\alpha = 0.75$, and no prior knowledge $T = 0$;
3) exploration $\alpha = 0.75$, and prior knowledge $T = 1000$.

Figure 8 and Figure 9 show whether the robots have good coverage in the partially observable context with regards to the detection of tasks, with $\alpha = 0$, $T = 0$ and $\alpha = 0.75$, $T = 1000$, respectively. Concretely, in both of the graphs, we compare the amount of spawned (in orange) vs. the amount of detected tasks (in blue color). We can see that the advanced approach combining exploration $\alpha = 0.75$ and prior knowledge $T = 1000$, shows a much better approximation of the spawned tasks by the detected tasks over time.

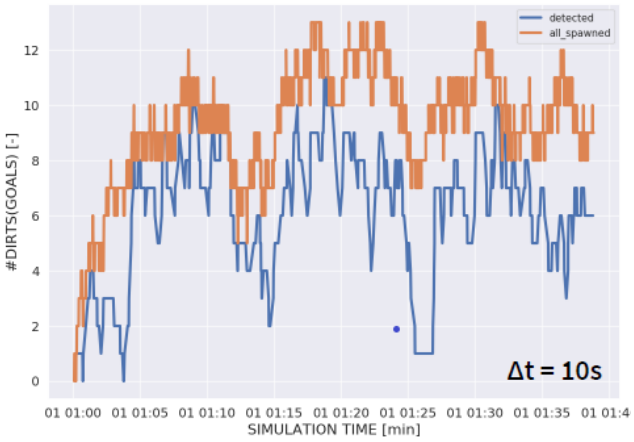

Figure 8. Spawned vs. detected tasks ($\alpha = 0$, $T = 0$).

The graph in Figure 10 shows how many goals are assigned to both of the robots over time. From the graph, we can see that with time, concretely in the second half of the simulation time, the number of assigned goals increases when we have exploration and prior knowledge (depicted in orange),

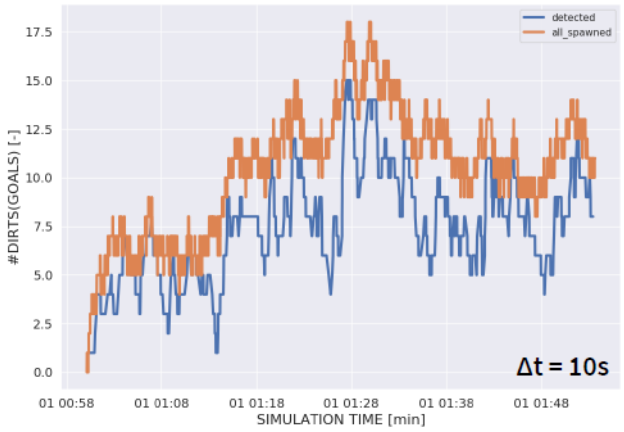

Figure 9. Spawned vs. detected tasks ($\alpha = 0.75$, $T = 1000$).

in comparison, when there is no exploration and no prior knowledge given (depicted in blue).

Figure 10. Number of currently assigned goals.

The succeeded goals graph in Figure 11 shows an accumulated number of succeeded goals over time. The results show that the approach which combines the exploration and the prior knowledge (in green color) performs the best over time, in average completing 5 goals/minute, following the approach with no exploration and no prior knowledge (in blue color) with 3.75 goals/minute, and at the end with only 2 goals/minute, the approach with exploration but no prior knowledge. Interestingly, our experiments revealed that the exploration benefits are only noticeable when the exploration is combined with the previously learned knowledge about the context. Otherwise, when the system explores without prior knowledge, it performs almost half worse than when the system did not explore and did not learn. From the results, we can conclude that a self-adaptive system benefits by a more extensive exploration of the partially observable context, only if the exploration is guided by the previous learning of the system.

 ISBN: 978-1-61208-848-8

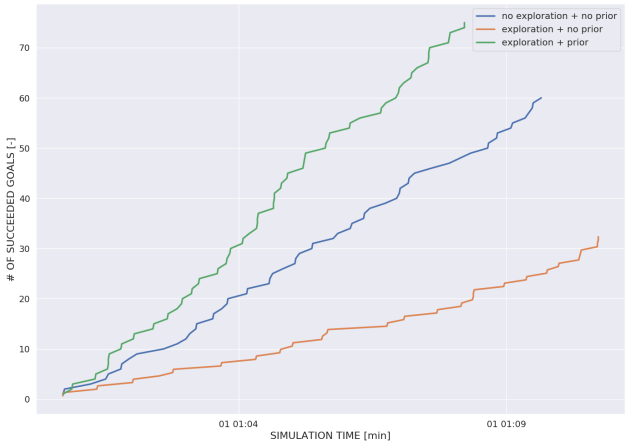

Figure 11. Succeeded goals (cumulative).

VII. Conclusion and Future Work

The objective of this work was to investigate how self-adaptive systems that establish their adaptation on incorporating human-like activities like collaboration and learning can preserve or even improve their performance—despite the continuous, run-time changes in the context that could not be specified during the design time. The systems operate in partially observable, multi-agent contexts. We proposed an approach for building adaptation logic, which improves over time and tackles different challenges of self-adaptive cyber-physical systems. The collaboration was enabled through run-time cooperative aggregations of the contextual observations and run-time collaborative tasks assignment. The learning was achieved by storing the past contextual encounters, which later were reused in a predictive manner, to help the systems make better, *smarter* decisions. To evaluate our approach, we built a self-adaptive system testbed from the robotics domain. As part of our future work, we intend to evaluate the applicability of the methodology on another use case from a different domain. Additional future enhancements should also comprise learning and optimal hyper-parameters search for different parameters, and changing the number of robots, for different contextual setups.

References

[1] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2016.

[2] H. Muccini, M. Sharaf, and D. Weyns, "Self-adaptation for cyber-physical systems: a systematic literature review," in *Proceedings of the 11th international symposium on software engineering for adaptive and self-managing systems*, pp. 75–81, 2016.

[3] P. Jamshidi, J. Cámara, B. Schmerl, C. Käestner, and D. Garlan, "Machine learning meets quantitative planning: Enabling self-adaptation in autonomous robots," in *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 39–50, IEEE, 2019.

[4] D. Weyns, "Software engineering of self-adaptive systems: an organised tour and future challenges," *Chapter in Handbook of Software Engineering*, 2017.

[5] A. Petrovska, S. Quijano, I. Gerostathopoulos, and A. Pretschner, "Knowledge aggregation with subjective logic in multi-agent self-adaptive cyber-physical systems," in *SEAMS '20: IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Seoul, Republic of Korea, 29 June - 3 July, 2020*, pp. 149–155, ACM, 2020.

[6] A. Petrovska and A. Pretschner, "Learning approach for smart self-adaptive cyber-physical systems," in *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, pp. 234–236, IEEE, 2019.

[7] M. Broy, M. V. Cengarle, and E. Geisberger, "Cyber-physical systems: imminent challenges," in *Monterey workshop*, pp. 1–28, Springer, 2012.

[8] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[9] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[10] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, 2004.

[11] V. Matena, T. Bures, I. Gerostathopoulos, and P. Hnetynka, "Model problem and testbed for experiments with adaptation in smart cyber-physical systems," in *2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 82–88, IEEE, 2016.

[12] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 1, pp. 698–705, IEEE, 2004.

[13] A. J. Ramirez, A. C. Jensen, and B. H. Cheng, "A taxonomy of uncertainty for dynamically adaptive systems," in *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 99–108, IEEE Press, 2012.

[14] E. L. Lawler, "The traveling salesman problem: a guided tour of combinatorial optimization," *Wiley-Interscience Series in Discrete Mathematics*, 1985.

[15] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2149–2154, IEEE, 2004.

[16] C. E. Agüero *et al.*, "Inside the virtual robotics challenge: Simulating real-time robotic disaster response," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 494–506, 2015.

[17] W. Garage and T. Foote, "TurtleBot 3 Burger." https://www.turtlebot.com/, 2016. [Online; accessed 19-July-2018].

[18] A. Petrovska, "Smart Self-Adaptive Cyber-Physical Systems Simulation." https://github.com/tum-i4/ssacps_simulation, 2019. [Online; accessed 08-Sept-2019].

[19] A. Petrovska, "Smart Self-Adaptive Cyber-Physical Systems Packages." https://github.com/tum-i22/ssacps_packages, 2019. [Online; accessed 08-Sept-2019].

 ISBN: 978-1-61208-848-8