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

Technical University of Munich, Department of Informatics

ana.petrovska@tum.de

About presenter: Ana Petrovska

Ana Petrovska since 2017 is a Ph.D. candidate at the Chair of Software and Systems Engineering led by Prof. Alexander Pretschner at the Technical University of Munich, Department of Informatics. Her research interest focuses in understanding and defining self-adaptivity, self-adaptive cyber-physical systems operating in dynamic environments, and system reasoning under uncertainties.

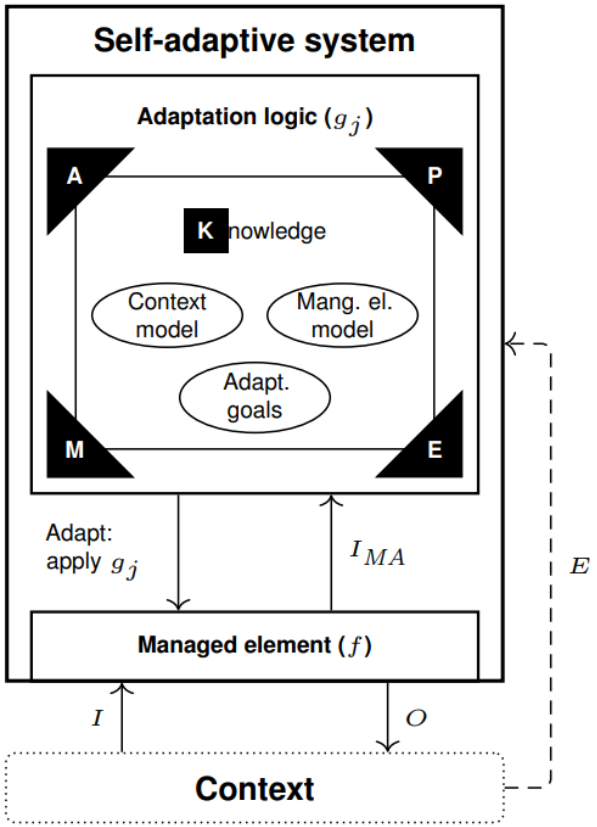

ana.petrovska@tum.de


https://www.in.tum.de/i04/petrovska/


Munich, Germany

Motivation

- Modern complex and dynamic systems as Cyber-Physical Systems (CPSs):
  - are composed of many interacting and interconnected components
  - inherit all the complexities of modern large-scale distributed systems [1]
  - deliver complex functionality [2]
  - cooperate and collaborate with other CPSs.
- A common approach to deal with run-time changes and uncertainties is to make the CPSs self-adaptive.
  - MAPE-K [6, 7, 8]
  - The Knowledge component comprises *models* of the CPS(s) and the context where they are operating.



Motivation cont.

- Additionally, modern CPSs:
  - operate in continually changing, uncertain, and unanticipated environments (*operational context*) [3, 4, 5]
  - additionally, CPSs often operate in:
    - partially observable context
    - multiagent
    - stochastic
    - sequential
    - dynamic
    - continuous, and
    - unknown context*.
  - abbreviated, PMSSDCU context.

* Unknown context does not refer to the context itself, but it refers to the knowledge that the CPSs have about the laws of physics of the context [6].

Identified problems

The prior works in the self-adaptive systems domain provide approaches where

1) the adaptation logic is predetermined and its structure does not change over time, e.g., [7], 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., [8].

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.

Paper contribution

- We tackle this issue by proposing a methodology for building adaptation logic for self-adaptive CPSs that operate in a PMSSDCU context. The context changes in a way that cannot be predicted during the design of the system.

- 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.

Reference problem: Cleaning robots

Setting:

- Dirt appears perpetually in different places at different points in time in the room
  - Unknown locations and frequency patterns
- One or more ground, mobile robots deployed in a room
  - Discover new dirt tasks, and later attain the tasks
- Adaptive Monte Carlo Localization (AMCL) for navigation and localization

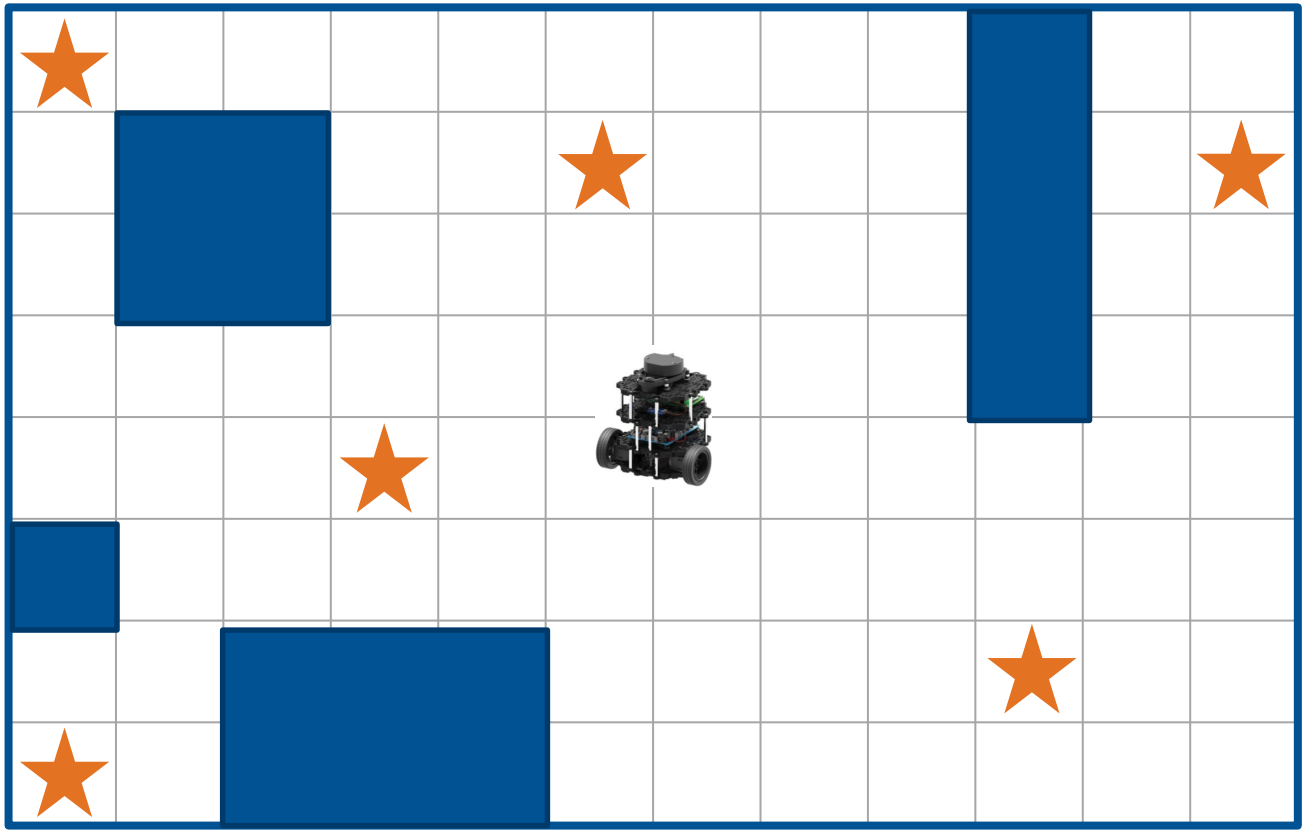
Mission goal:

- Keep the room clean by removing the dirt

Adaptation goals:

1. Minimizing the time needed for the room to be cleaned and be kept clean
2. Increasing its fault-tolerance by avoiding failures (e.g. collision with other robots) and deadlocks.

Reference problem: In a perfect world…



- Single robot
- Perfect sensors
- Static obstacles
- Predefined number of dirt tasks
- Limitless sensor range – complete overview of the room
- Sensors do not introduce any uncertainties
- Robot never fails
- Perfect AMCL

However, this does not resemble reality!

Reference problem: The problems are…

1.

Sensors are not perfect: they have a partial observation range, and introduce different run-time uncertainties, e.g., sensor imprecision, noise, ambiguity, inconsistency and inaccuracy, and even sensor failures [9]

2.

Continuous appearance of new tasks with unknown patterns: The run-time decisions on *how* the new tasks are assigned to the robots, and *what* path the robots take to reach to those tasks can significantly influence the system adaptation goals.

3.

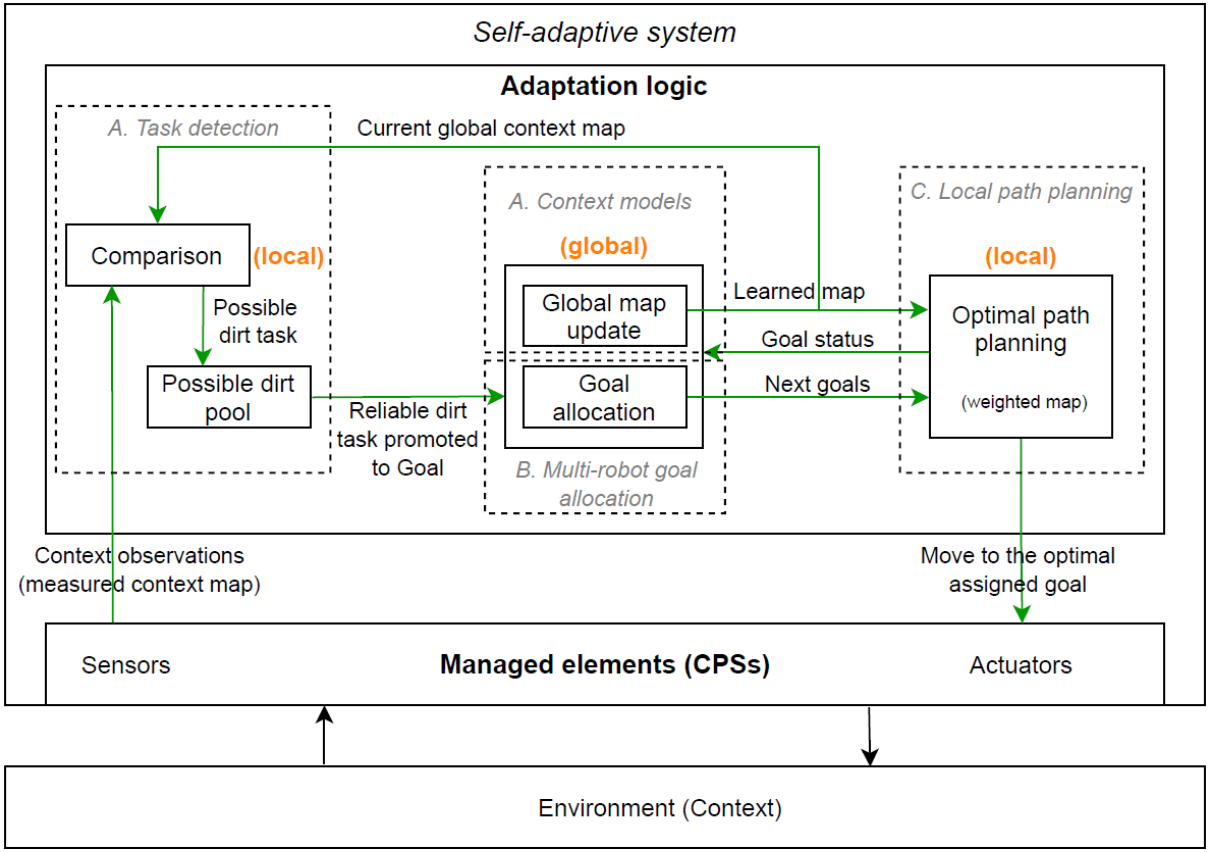
Navigation and localization do not work perfectly: when multiple agents need to localize themselves and navigate in a room, the other agents deployed in their relative proximity indirectly influence their actions. This can potentially lead to different AMCL localization and navigation issues, which can later result as sources of failures.

caused by internal system changes

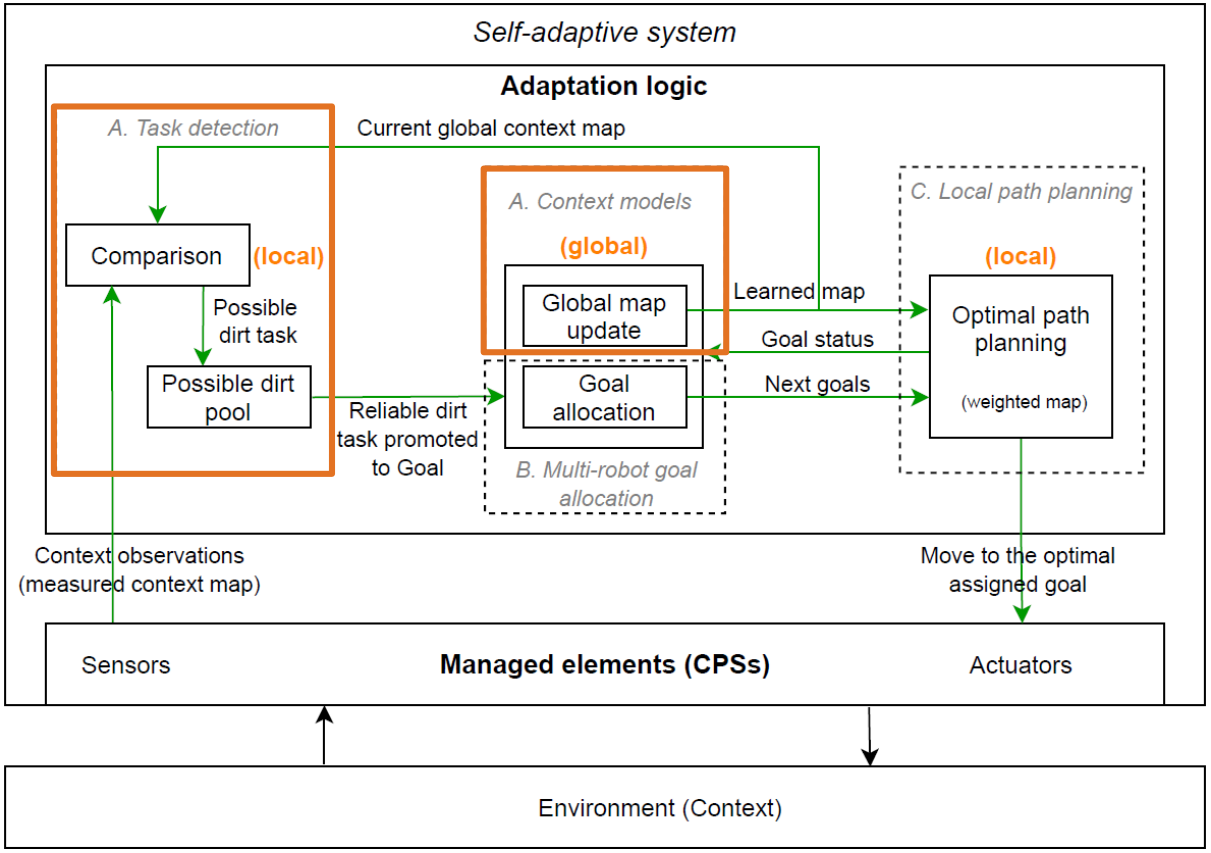
caused by changing context

These changes are triggers for system self-adaptation.

Methodology



Methodology



Tasks detection, knowledge representation and context models

- Task detection: a local phase, decentralized and distributed in every CPSs, i.e., robot.
  - Every agent independently detects the newly appearing dirt tasks, with a confidence value of 10% upon detection.
  - The dirt tasks are published as goals, for all the robots, once the confidence value exceeds 90%.

- The knowledge representation via context models in the adaptation logic: global phase, shared among all the CPSs, i.e., the robots.
  - It is the best possible representation of the actual context, i.e., the run-time state of the room in which the robots operate.
  - 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
  - Every CPSs updates the same, shared knowledge, based on the tasks that the systems discover on a local level

Updating context models based on probabilistic models

We update the context models during run-time, based on two probabilistic maps:

PROBABILITY GRID MAP

Probability of a dirt appearing in grid cell i, j in the next time step:

$$P_{ij}(T) = \frac{N_{ij}}{T}$$

with $N_{i,j}$ = number of dirt tasks found since $T = 0$

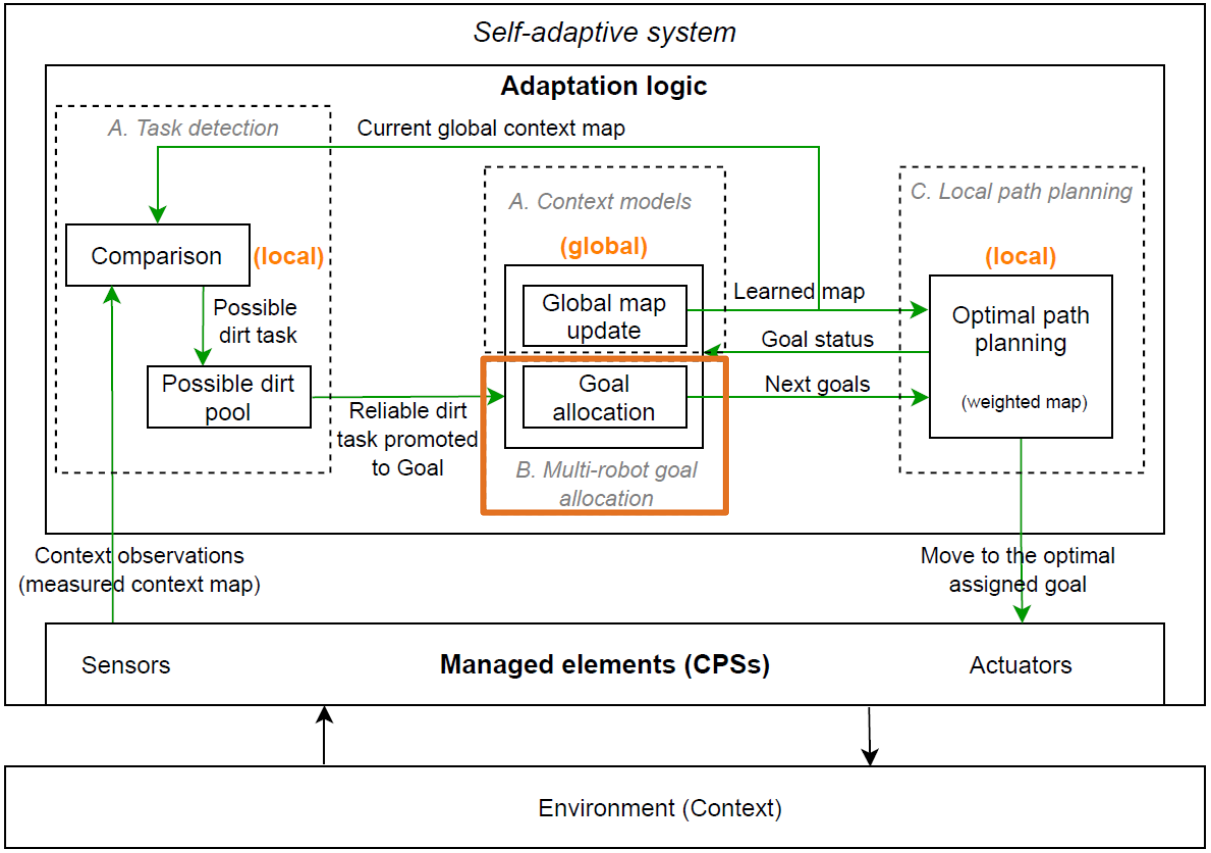
CUMULATIVE PROBABILITY GRID MAP

Probability of at least one dirt tasks being within the grid cell i,j:

$$CP_{ij}(T) = 1 - (1 - P_{ij}(T-1))(1 - CP_{ij}(T-1))$$

Probability that there is not a single task

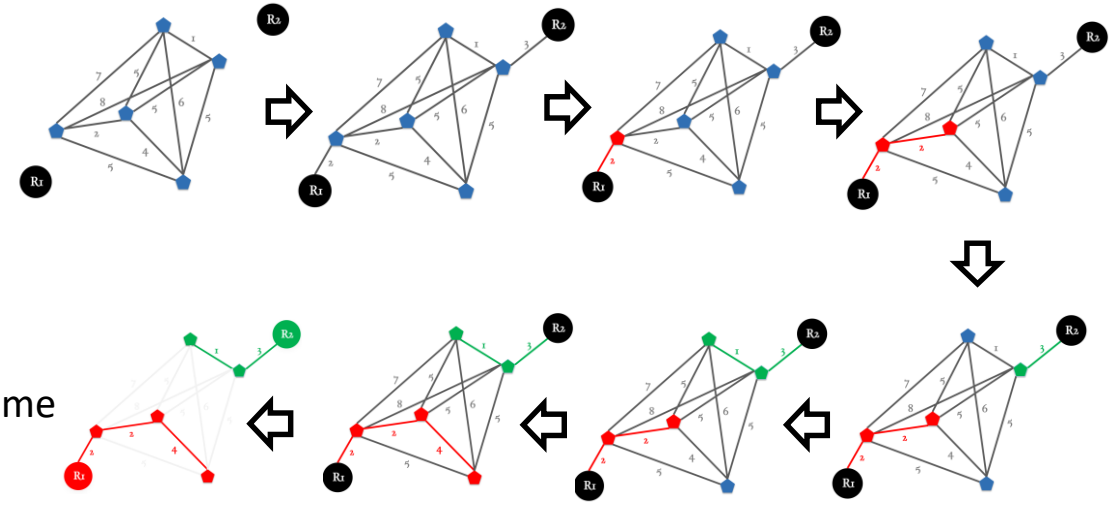
Methodology



Multi-robot goal allocation

GOAL

Finding an optimal allocation in which the overall sum of travel costs of all robots when visiting all detected tasks is minimized.



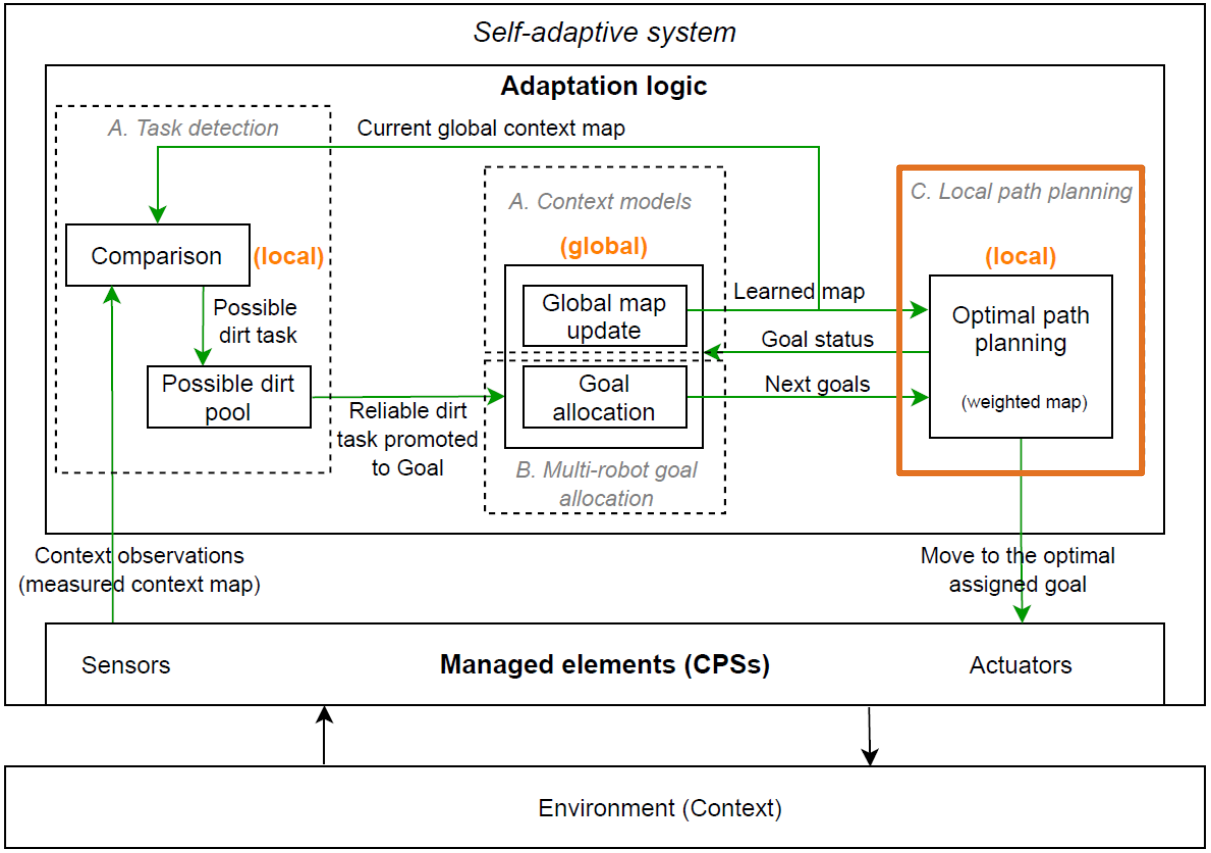
REQUIREMENTS

- Close-to-optimal solution
- Computationally feasible
- Dynamically adaptable during run-time

APPROACH

Multi-robot task allocation using minimum spanning forests (MSF), based on the greedy principle termed Prim Allocation [10], using Euclidean distance heuristic

Methodology



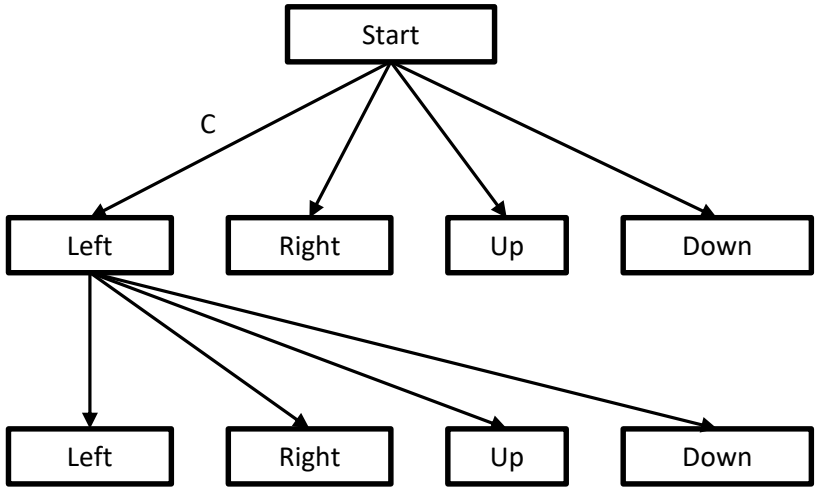
Local path planning

- Path planning: a local phase, decentralized and distributed in every CPSs, i.e., robot.
  - Compromise with the multi-robot goal allocation from the previous phase
  - Minimum distance (from the goal allocation) + maximum exploration (from the path planning)
  - The motion of the agents is discrete, and with each timestamp they move *up*, *down*, *left* and *right*

GOAL

Balance time minimization with exploration

- Use uniform cost graph search for path planning



Cost function:

$$C = d(left) - \alpha \cdot E(x, y, left)$$

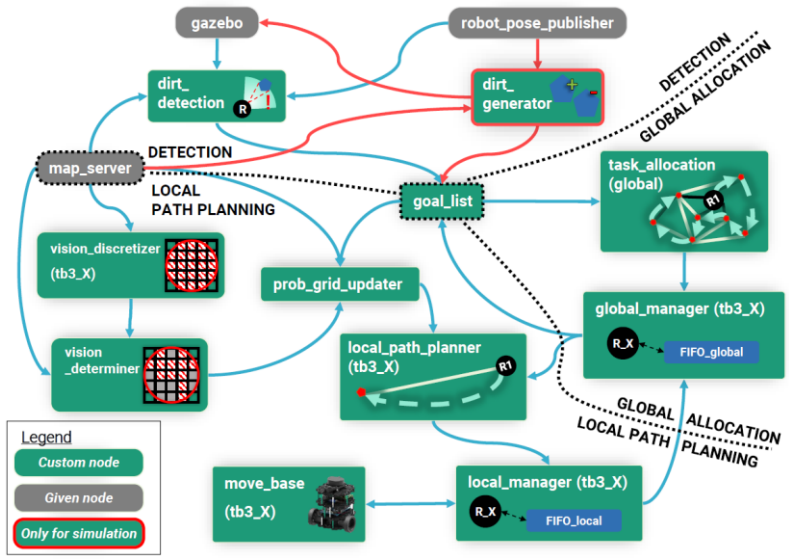
distance when moving left

tuning parameter

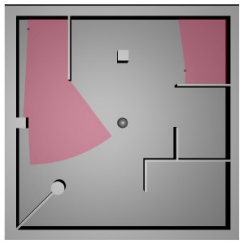
exploration gain (by moving left from position x,y)

Implementation

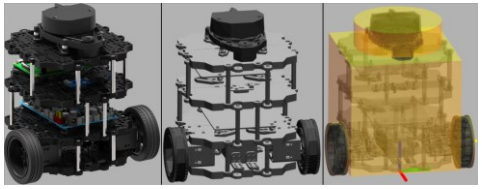
- ROS-based communication
- Simulated in Gazebo [11, 12]
  - TurtleBot3 Burgers[1]
  - closely resemble the real world
- Simulate many robots, and different room maps can be used
  - The evaluation is done with a setup of two robots in a single room
- Deployment to real robots without any modification
- Implementation is open-source[2]



ROS implementation architecture



Map of the room, the deployed robots and their observation range



TurtleBot3 Burger

1 https://www.turtlebot.com/
2 https://github.com/tum-i22/ssacps_simulation, https://github.com/tum-i22/ssacps_packages

Evaluation

- Experimental setup:

- 8 experiments in total: 1 long term (40 minutes) and seven short-term (10 minutes)
- during all the experiments we vary:
  - the exploration parameter α
  - the time-interval of dirt task spawn Δt
  - the use of prior learned knowledge gained in time T*
- for a better replication of the experiments, we fixed the frequency Δt and used the random seed for the appearance of the tasks.

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

Experiments parameters specifics

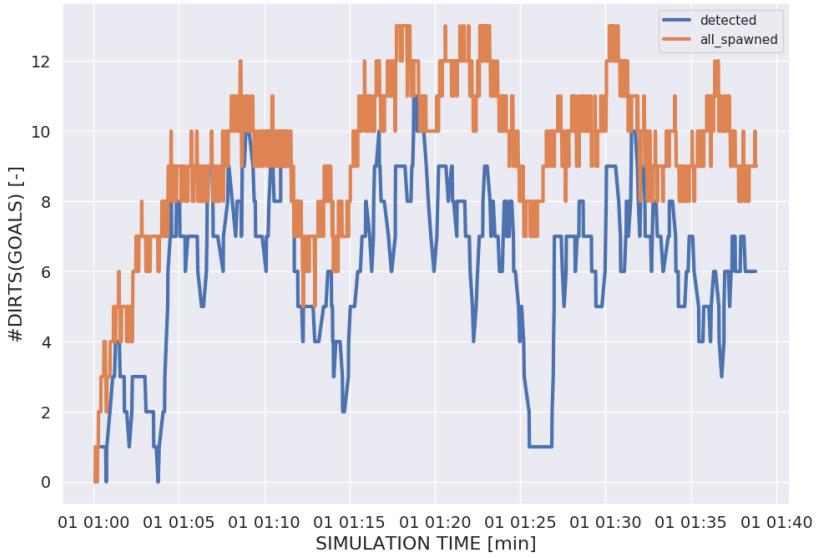
* the prior learned knowledge comes in the form of probability task distribution that is learned for 1000 time-steps (in seconds) before the actual measurements are collected.

Results: Spawned vs. detected tasks

- Investigate whether the robots have a good coverage in the partially observable context with regards to the detection of tasks:

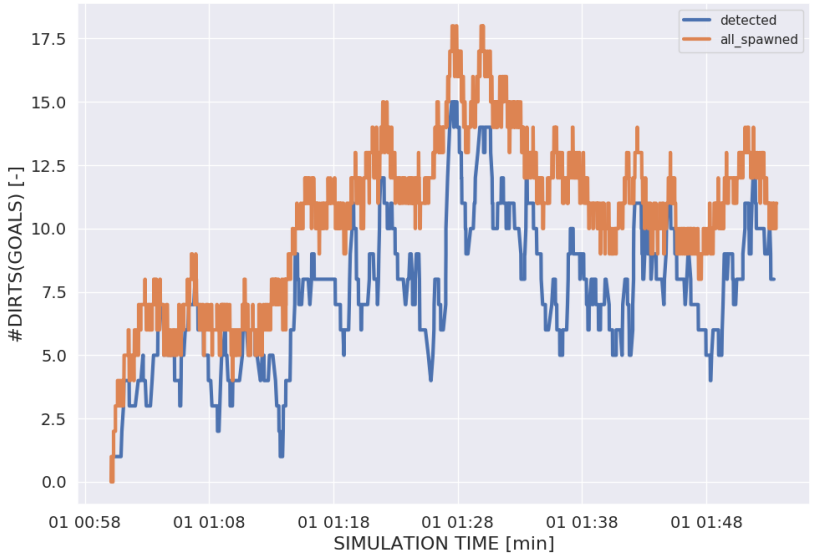
- No exploration and no prior knowledge

$\alpha = 0, T = 0$



- With exploration and prior knowledge

$\alpha = 0.75, T = 1000$



RESULTS: With exploration and prior knowledge shows a much better approximation of the spawned tasks by the detected tasks over time.

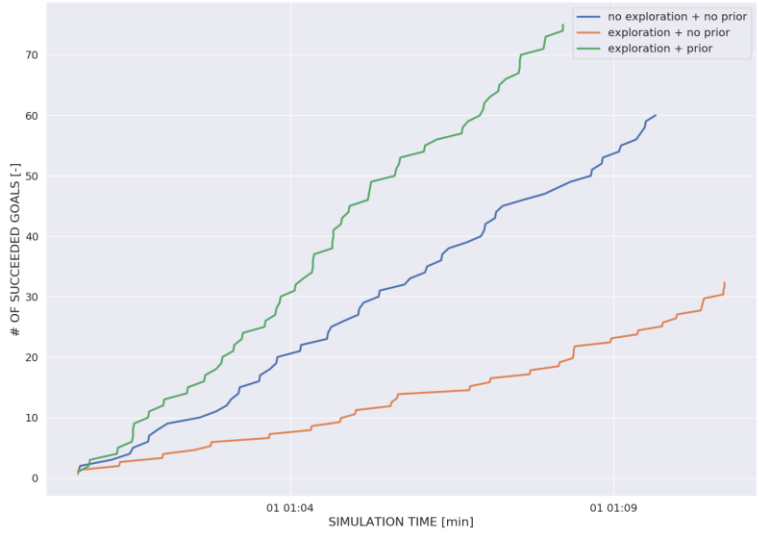
Results: Goals assigned over time

α = 0, T = 0 vs. α = 0.75, T = 1000

RESULTS: We can observe that the number of assigned goals increases when we have exploration and prior knowledge (depicted in orange), in comparison, when there is no exploration and no prior knowledge given (depicted in blue).

Results: Succeeded goals (cumulative)

α = 0, T = 0 vs. α = 0.75, T = 0 vs. α = 0.75, T = 1000



RESULTS: 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.

Conclusion

- We investigated 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 PMSSDCU context that could not be specified during the design time.
- As part of this paper, we proposed an approach for building adaptation logic, which improves over time and tackles different challenges of self-adaptive CPSs.
- The collaboration is enabled through run-time cooperative aggregations of the contextual observations and run-time collaborative tasks assignment.
- The learning is achieved by storing the past contextual encounters, which later are 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.

References

[1] Muccini, Henry, Mohammad Sharaf, and Danny Weyns. "Self-adaptation for cyber-physical systems: a systematic literature review." *Proceedings of the 11th international symposium on software engineering for adaptive and self-managing systems*. 2016.
[2] Weyns, Danny, and Radu Calinescu. "Tele assistance: A self-adaptive service-based system exemplar." *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 2015.[1] SEAMS, "Self-Adaptive Systems Artifacts and Model Problems. Tele Assistance System (TAS)."
[3] Mahdavi-Hezavehi, Sara, Paris Avgeriou, and Danny Weyns. "A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements." *Managing Trade-Offs in Adaptable Software Architectures*. Morgan Kaufmann, 2017. 45-77.
[4] Quin, Federico, et al. "Efficient analysis of large adaptation spaces in self-adaptive systems using machine learning." *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2019.
[5] Petrovska, Ana, and Alexander Pretschner. "Learning Approach for Smart Self-Adaptive Cyber-Physical Systems." *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE, 2019.
[6] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.
[7] 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.
[8] V. Matena, T. Bures, I. Gerostathopoulos, and P. Hnetynka, "Model problem and testbed for experiments with adaptation in smart cyberphysical systems," in 2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 82–88, IEEE, 2016.
[9] 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.
[10] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multirobot 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.
[11] Nathan Koenig and Andrew Howard. 2004. 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).
IEEE, 2149–2154. https://doi.org/10.1109/IROS.2004.1389727
[12] Carlos E. Aguero, Nate Koenig, Ian Chen, Hugo Boyer, Steven Peters, John Hsu, Brian Gerkey, Steffi Paepcke, Jose L. Rivero, Justin Manzo, Eric Krotkov, and Gill Pratt. 2015. Inside the Virtual Robotics Challenge. 12 (2015), 494–506. Issue 2.
https://doi.org/10.1109/TASE.2014.2368997

Chair of Software and Systems Engineering
TUM Department of Informatics
Technical University of Munich

Thank you!

Contact me:

Ana Petrovska

ana.petrovska@tum.de