Learning Optimal Hyper-Parameters in Smart Self-Adaptive Cyber-Physical Systems

Master Thesis

Supervisor: Prof. Dr. Alexander Pretschner
Advisor: Ana Petrovska
Email: {alexander.pretschner, ana.petrovska}@tum.de
Phone: +49 (89) 289 - 17830

Starting date: 15.11.2019

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17830
https://www4.in.tum.de

Context

Cyber-physical systems (CPSs) have to adapt more and more to dynamic environments, to be able to meet the needs and challenges of the 21st century and beyond. Therefore it is necessary to design software which optimizes the behavior in changing environments. One approach for self-adaptation in a multi-robot scenario was developed and evaluated by Andreas Mütter, Ece Tanova, and Julian Weick in the practical course *Smart Self-Adaptive Cyber-Physical Systems–A Road towards Autonomous Systems (SSACPS)* during the summer term 2019 [1]. The goal of this practical course was to make a set of robots collaborate and to do this adaptively, for example, by adjusting the behavior of the robots to the run-time changes in the environment that could not be specified during design time. In our use case, a set of two cleaning robots is supposed to collaborate and self-adapt in order to clean (or complete tasks) in a given map as efficient as possible [2].

The implementation of the adaptation logic as part of the practical course consisted of several modules based on a shared knowledge base, covering different levels of behavior, such as best-first search and collaboration via entropy repulsion. The entropy repulsion allows the robots to work nearby if the tasks are in proximity, or it separates the agents when the tasks are widely distributed in the space. Further, modules to emphasize the exploration of unseen areas, learning/predicting of new task appearance, and task importance indication via clustering have been additionally implemented.

The overall attractiveness of a task (or area to explore) is the weighted sum of the above-described modules:

$$policy = \alpha M1(a) + \beta M2(b) + \gamma M3(c) + \dots$$

where $M1, M2, M3$ are the modules referred above; α, β, γ are the modules' weights; and a, b, c are the hyper-parameters from the modules. In our case, the weights $\alpha, \beta, \gamma, \dots$ are hyper-parameters themselves. Further, we have some hyper-parameters $a, b, c, \dots$ in the modules themselves. For example, the numbers of the overall clusters (which need to be specified because the clustering algorithm is K-means), the decaying of the importance of previously learned tasks, and several more. Overall it sums up to >10 hyper-parameters.

For the evaluation purposes in the practical course, the hyper-parameters have been adjusted manually, and several points in the multidimensional parameter space have been probed. The evaluation results of the implemented self-adaptive system showed a significant performance increase compared to a simple, non-adaptive approach. In the non-adaptive approach, the tasks' priorities and how they are assigned to the robots were, by default, always proportional to their distance from the robots. Nevertheless, several questions such as 1) how to optimize the hyper-parameters automatically, 2) how to improve the policy, and 3) how to evaluate the performance for different setups, have arisen at the end of the initial evaluation. This thesis wants to complete the work by addressing these questions and using a learning approach for the hyper-parameters, rather than setting them manually. It will open the gate for a real and complete evaluation of the approach, and ideally identify the need for further research and contributions in this field.

Goal

The goal of this work is first to develop an algorithm that learns and continuously optimizes the hyper-parameters and second, to evaluate how much the hyper-parameters deviate in different contextual setups. We model the context as a grid map of the space where the robots are operating. Therefore, the setup consists of different scenarios: maps of the different rooms (with varying sizes and a varying number of static obstacles in it) and changing distribution, total number, and appearance rate of the new tasks for the robots. The overall approach can be seen in Figure 1. In this thesis we want to consider two different type of maps: artificial

and realistic. Artificial maps will be designed to have a wide configuration variety of room and obstacle placement and shape. This is supposed to maximize the hyper-parameter spread, which further is described below. The realistic maps, on the other hand, will represent real-world indoor space maps, for example, the Magistrale in the MI building, or different seminar rooms in the same building.

First, as mentioned above, we will implement a learning module, which will learn the hyper-parameters for several randomly generated artificial maps and will accordingly store them for evaluation, as seen in Figure 2a. Evaluating the difference between multiple artificial setups will give us a distribution of optimal hyper-parameters. The variance of the distribution should lead to an upper bound on the spread for every hyper-parameter.

Second, we want to evaluate the optimal hyper-parameters for a set of realistic maps. This opens the possibility to investigate several points:

1. Comparing and finding the differences between the hyper-parameter distributions of the real-world and artificial scenarios. We expect different magnitudes of congruence for the different hyper-parameters and hope to get more insight into how they influence the overall behavior of the system.
2. Transferring the optimal hyper-parameters between artificial and realistic maps to investigate the performance behavior of the model trained on synthetic data in a real-world scenario. This performance difference, together with the upper bounds of the spread of every hyper-parameter ideally should indicate, if with our approach the expected performance shortage is negligible in practice, and can be used for every scenario, or if it is necessary to invest in further research on parameter adaption on the context.

Finally, we want to give an outline of how much the uncertainties, errors (FP and FN rates), and parameters such as sensors' range and noise influence the optimal behavior. For that purpose, a sensor noise module in the simulator will add different types of noise on the different setups. For each setup and noise set, the optimal hyper-parameters will be again calculated by the algorithm and re-evaluated (Figure 2b).

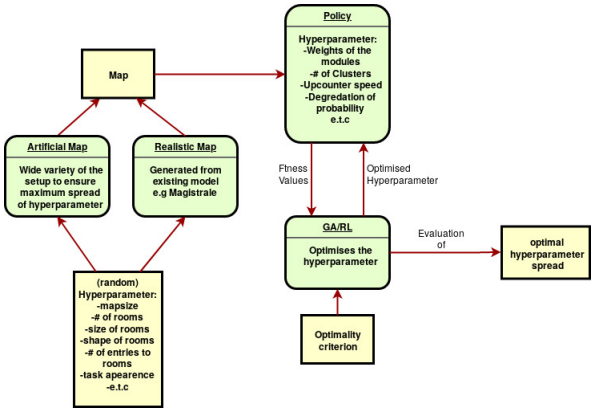
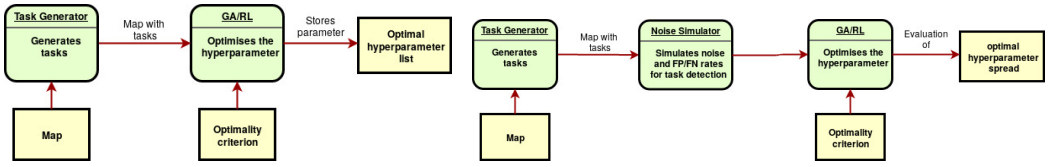


Figure 1: Overall Approach



(a) Random generation (b) Evaluate behaviour under noise

Figure 2: Schematic of the two evaluation approaches

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17830
https://www4.in.tum.de

Working Plan

1. Familiarize ourselves and evaluate the benefits of using generic algorithms over reinforcement learning approaches
2. Programming and optimization of the supporting modules
    - Implement ROS hyper-parameter server which supports updates from an non ROS script
    - Refining and extending the policy
    - Define performance criterion and the procedure for the validation
3. Implementation of the GA or reinforcement learning approach
    - Setting up the artificial and realistic maps [1]
    - Generalize an existing dirt generator to be able to adapt to changing setups
    - Hyper-parameter optimization using GA or reinforcement learning.
4. Evaluation:
    - Evaluate the spread of the optimal hyper-parameters on the artificial maps and identify the magnitudes of congruence.
    - Transferring the optimal hyper-parameters between artificial and realistic maps to investigate the performance behavior.
    - Run tests on how much the uncertainties, sensor errors ,and parameters such as sensor range influence the optimal behavior.
5. Documentation
    - Introduction and overview of the work the thesis is based on
    - Software architecture and communication
    - Analysis of the testing
    - Conclusion and future work

Deliverables

- Source code of the implementation.
- Technical report with comprehensive documentation of the implementation, i.e. design decision, architecture description, API description and usage instructions.
- Final report written in conformance with TUM guidelines.

References

[1] J. W. Ece Tanova and A. Mütter, “Ssacps team 4: Approach description,” 10 2019.

[2] A. Petrovska and A. Pretschner, “Learning approach for smart self-adaptive cyber-physical systems,”

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17830
https://www4.in.tum.de

[1] Co-developed with Guan Erjiage.