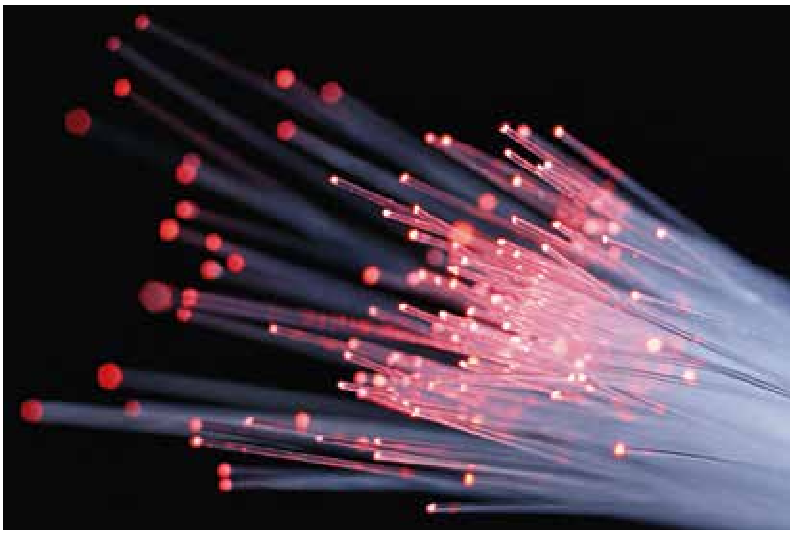
MÄLARDALEN UNIVERSITY
SWEDEN

DPAC Newsletter
Spring 2021



Dependable Platforms for
Autonomous systems and Control

D P A C

About DPAC

The DPAC profile establishes a leading research profile targeting dependable platforms for autonomous systems and control, hosted at Mälardalen University. DPAC is organized through close collaboration and interaction between several research groups at MDH and a set of participating industrial companies. The profile leverages our solid track record of close cooperation to conduct excellent research, knowledge transfer, and support commercialization with industrial partners. DPAC shall create synergy effects between the partners and a significant increase in coproduction is expected.

The ultimate goal of the DPAC profile is to establish a nationally leading and internationally renowned research centre that facilitates close cooperation between academia and industry to achieve a significant increase in research and available knowhow on advanced dependable platforms for embedded systems. Embedded computer systems are nowadays incorporated in many kinds of products including many mission-critical applications such as trains, autonomous utility vehicles, aviation, smart grid power management etc. These systems offer advanced functionality and serve an important role for the competitiveness of companies and the future national and global infrastructure. The scientific and technical results of DPAC will support future innovation by providing dependable platforms that can be used to efficiently realize dependable, reliable and safe electronically controlled products.

Contents

ABB ALTEN ERICSSON Senseair AsahiKASEI UNIBAP VOLVO VOLVO GROUP

BOMBARDIER SAAB ENEA Arcticus Systems

MÄLARDALEN UNIVERSITY SWEDEN Knowledge Foundation

Welcome to the DPAC Newsletter for Spring 2021

Dear DPAC'ers and friends, we once again decided to offer a newsletter instead of hosting a summit. Work at MDH has continued, and we have several new results and publications to share with you. Some of our results are presented in this issue, in the form of short papers focusing on key scientific results. For more details on research results, contact the project leaders (listed below) or the authors of the short papers directly.

We look forward to intensifying cooperation with our partner companies during 2021, primarily within the three subprojects P1, P2, and P3, and in the demonstrator. We also hope to be able to host a regular summit during 2021, or alternatively a number of mini-summits online.

DPAC will continue as planned, with researchers and industrial partners working together towards realizing dependable platforms for autonomous systems and control, and as soon as times allow, we will continue with organizing summits and other joint events.

P1: Predictability and dependability in parallel architectures:

This area addresses challenges in designing predictability and dependability in parallel architectures for electronic platforms, and challenges in developing dependable and predictable software platforms that execute on such electronic platforms.
Contact: Saad Mubeen,
saad.mubeen@mdh.se

P2: Autonomous systems and control:

This area addresses challenges with respect to achieving dependability in autonomous systems. We specifically target control-intensive systems that should operate in a dependable, reliable, and safe way without a human operator providing detailed control.
Contact: Mikael Ekström,
mikael.ekstrom@mdh.se

P3: Design methodologies:

This area addresses challenges associated with design methodologies used for developing dependable platforms and systems. It includes methodologies to capture and validate correct sets of requirements and design assurance, and it focuses on ways to simplify the complex dependable embedded system models and make analysis more tractable.
Contact: Håkan Forsberg,
hakan.forsberg@mdh.se

Demonstrator:

This area addresses the design and implementation of a predictable autonomous wheel loader, which acts as the unifying use case in DPAC. The wheel loader is based on applying selected results of the research conducted in P1, P2, and P3, with respect to dependable architectures, autonomous intelligent algorithms, and design methodologies.
Contact: Cristina Seceleanu,
cristina.seceleanu@mdh.se

Selected DPAC publications during 2020:

1. **A software Implemented Comprehensive Soft Error Detection Method for Embedded Systems** (Sep 2020) Seyyed Amir Asghari, Mohammadreza Binesh Marvasti, Masoud Daneshtalab ***Elsevier Journal of Microprocessors and Microsystems*** (MICPRO)
2. **A Systematic Literature Study on Definition and Modeling of Service-Level Agreements for Cloud Services in IoT** (Aug 2020) Svetlana Girs, Séverine Sentilles, Sara Abbaspour Asadollah, Mohammad Ashjaei, Saad Mubeen ***IEEE Access*** (ACCESS'20)
3. **Improving Motion Safety and Efficiency of Intelligent Autonomous Swarm of Drones** (Aug 2020) Amin Majd, Mohammad Loni, Golnaz Sahebi, Masoud Daneshtalab ***Drones*** (Drones)
4. **The Genetic Algorithm Census Transform: Evaluation of Census Windows of Different Size and Level of Sparseness through Hardware In-The-Loop Training** (Jul 2020) Carl Ahlberg, Miguel Leon Ortiz, Fredrik Ekstrand, Mikael Ekström
5. **Open Access Journal of Real-Time Image Processing** (J RT Image Proc)
6. **Verification of Cyberphysical Systems** (Jul 2020) Marjan Sirjani, Edward Lee, Ehsan Khamespanah ***Mathematics*** (Mathematics)
7. **Modelling Multi-Criticality Vehicular Software Systems: Evolution of an Industrial Component Model** (Jun 2020) Alessio Bucaioni, Saad Mubeen, Federico Ciccozzi, Antonio Cicchetti, Mikael Sjödin ***International Journal on Software and Systems Modeling*** (SoSyM'20)
8. **VeriVANca Framework: Verification of VANETs by Property-Based Message Passing of Actors in Rebeca with Inheritance** (Jun 2020) Farnaz Yousefi, Ehsan Khamespanah, Mohammed Gharib, Marjan Sirjani, Ali Movaghar ***International Journal on Software Tools for Technology Transfer*** (STTT)
9. **A Review on Deep Learning Methods for ECG Arrhythmia Classification** (Jun 2020) Zahra Ebrahimi, Mohammad Loni, Masoud Daneshtalab, Arash Ghareh Baghi ***Expert Systems with Applications*** (ESWA)
10. **Enabling Radiation Tolerant Heterogeneous GPU-Based Onboard Data Processing in Space** (Jun 2020) Fredrik Bruhn, Nandinbaatar Tsog, Fabian Kunkel, Oskar Flordal, Ian Troxel ***CEAS Space Journal*** (CEAS20)
11. **NOM: Network-On-Memory for Inter-Bank Data Transfer in Highly-Banked Memories** (May 2020) Seyyed Hossein Seyyedaghaei Rezaei, Mehdi Modarressi, Rachata Ausavarungnirun, Mohammad Sadrosadati, Onur Mutlu, Masoud Daneshtalab ***IEEE Computer Architecture Letters*** (CAL)
12. **Timing Predictability and Security in Safety-Critical Industrial Cyber-Physical Systems: A Position Paper** (Apr 2020) Saad Mubeen, Elena Lisova, Aneta Vulgarakis Feljan ***Applied Sciences--Special Issue "Emerging Paradigms and Architectures for Industry 4.0 Applications"*** (ApplSci'20)
13. **DeepMaker: A Multi-Objective Optimization Framework for Deep Neural Networks in Embedded Systems** (Jan 2020) Mohammad Loni, Sima Sinaei, Ali Zoljodi, Masoud Daneshtalab, Mikael Sjödin ***Elsevier Journal of Microprocessors and Microsystems*** (MICPRO)

A Trade-Off between Computing Power and Energy Consumption of On-Board Data Processing in GPU Accelerated In-Orbit Space Systems

Nandinbaatar Tsog*, Saad Mubeen*, Mikael Sjödin*, Fredrik Bruhn*†
*Mälardalen University, Västerås, Sweden
{nandinbaatar.tsog, saad.mubeen, mikael.sjodin, fredrik.bruhn}@mdh.se
†Unibap AB (publ), Uppsala, Sweden
f@unibap.com

***Abstract*—Intelligent on-board data processing encounters higher energy consumption, although it improves the performance capability of in-orbit space systems such as deep-space exploration. In this paper, we consider a graphics processing unit (GPU) accelerated in-orbit space systems for on-board data processing. According to our prior works, there exist radiation-tolerant GPU, and the computing capability of systems is improved by the balanced execution of parallel segment either on GPU parallel or on central processing unit (CPU) sequential manners. Hence, we conduct experimental observations of energy consumption and computing power using this balanced execution method in our GPU accelerated in-orbit space systems. The results show that the proper use of GPU increases computing power with 10-140 times and consumes between 8-130 times less energy. Furthermore, the entire task system consumes 10-65% less energy compared to the traditional use of processing units.**

I. Introduction

In the space community, technological advances make it possible to work on a new challenge for on-orbit activities [1] including in-orbit servicing and in-situ experiments. On-board data processing is one of the prior on-orbit activities that improves the performance capability of in-orbit space systems. We consider that the advanced on-board data processing solves the current communication limitation which is low-speed connections between satellites and ground stations with limited access time intervals. Furthermore, the rapid development of technology makes advanced on-board data processing possible for small scale space systems using heterogeneous processing units that meet the requirements of size and weight limitations [2].

The interest of using heterogeneous computing in real-time and low-end embedded systems is increasing along with advanced on-board processing such as machine learning and computer vision algorithms. However, heterogeneous processing units require more energy in total than micro-controllers or a single-core processor [3]. In addition, the use of GPUs in the context of space is not well studied yet, due to the prior concern that GPUs are not suitable for the radiation-hardened environments. Therefore, in this paper, we consider a trade-off between computing power and energy consumption, focusing on the entire task set with different scenarios in GPU accelerated systems.

This paper is a curated paper for the DPAC newsletter Fall 2020, and the original of the paper is accepted in the Transactions of JSASS, Aerospace Technology Japan.

Our contribution in this paper is to conduct observations of energy consumption in GPU accelerated real-time systems while using the mapping method for the balanced use of heterogeneous processors. These observations provide us the fundamental understanding to perform the dynamic allocation of tasks to the heterogeneous processors under limited energy budget.

II. System Model

In order to study a trade-off between computing power and energy consumption, we consider a task set Γ, which consists of n independent periodic tasks $\{\tau_1, .., \tau_n\}$ that their parallel segments can be executed either on GPU parallel or on CPU sequential manner [4]. Each task τ_i has a period T_i, deadline D_i, and worst case execution time (WCET) C_i. The response time (RT) R_i of task τ_i is measured by experimental observations in this paper. The system energy consumption E_{system} can be described and calculated by the following two ways focusing on: i) the energy consumption of peripherals such as E_{CPU} on CPU, E_{GPU} on GPU, and E_{other} on others, ii) the execution stages of system such as $E_{preparation}$ and $E_{execution}$ for the data preparation and for execution parts, respectively. Hence, $E_{system} = E_{CPU} + E_{GPU} + E_{others}$ and $E_{system} = E_{preparation} + E_{execution}$.

III. Experimental studies

A. Experimental design

1) Algorithms: Two on-board algorithms, namely Anisotropic Diffusion [5] and Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH) [6], are considered in this paper. We have ported the 2D Anisotropic Diffusion code from MATLAB to C++ Accelerated Massive Parallelism (C++ AMP), OpenCL and OpenMP in order to execute the application on Heterogeneous System Architecure (HSA) compliant platforms. LULESH suits well in order to evaluate the parallelism and is provided for MPI, OpenMP, OpenCL and C++ AMP.

2) Testbeds: Two test machines, A10 and R&R, are used for this experiment. A10 is a low-end platform which employs A10 series accelerated processing unit (APU), normally termed as ”integrated GPU”. R&R is a high-end custom made desktop computer and consists of AMD Ryzen 7 CPU and AMD Radeon R9 nano discrete GPU. Both A10 and R&R are used for experimental observations 1 and 2, while only A10 is used for observation 3.

3) Experimental observations: The following three observations are evaluated in this paper: i) compiler vs computing power, ii) energy consumption vs programming manner, and iii) energy consumption vs execution manner. The observations are described as follows. For the observation 1, a test application is compiled by three different versions (GCC5.4.0, GCC6.2.1, and GCC 7.1.0) of GCC compiler together with 2 different options, ”non-optimised” with ”-O0” flag and ”optimised” with ”-O3”. In the observation 2, a test application is implemented in three programming manners as using HSA for parallelization on GPU, normal sequential execution on CPU, using OpenMP for parallelization on CPU. This is about how parallelization differs by using different techniques such as HSA and OpenMP. The observation 3 considers experiments of a task set with two independent tasks (τ_1 and τ_2) in which the tasks are allocated to the different processing units. For example, a parallel segment of τ_1 is allocated to CPU in one case, while the segment is allocated to GPU in another case.

B. Evaluation and Results

1) Observation 1: On both A10 and R&R, the HSA calculation (HSACalc) shows between 122 to 152 times faster than the calculation which uses 1 core CPU with non-optimised version. This ratio improves in the case of 1 core CPU with optimised version, however, HSACalc shows still between 11 to 22 times faster computation time than 1 core CPU. Moreover, A10 HSACalc shows the best computation time among others, even better or similar R&R HSACalc. In general, a high end discrete GPU has significantly more compute cores than an integrated GPU. However, the relative performance of a discrete GPU over an integrated GPU is workload dependent. The more the data that need to be densely processed, the higher the desired frame-rate, and the more processing steps they will undergo, the more likely it is that a discrete GPU will outperform an integrated GPU. However, as argued in this paper, they will both typically significantly outperform even a very powerful CPU.

2) Observation 2: We confirm that the execution part of HSACalc uses between 15 to 132 times less energy consumption compared to the execution part of CPUCalc. Similar to the comparison of energy consumption regarding the execution parts of HSACalc and OMPCalc is between 8 to 55 times difference, which means HSACalc consumes that much less energy. In other words, adapting an HSA compliant GPU uses between 8 to 132 times less energy compared to the CPU cores.

3) Observation 3: We consider an execution manner of concurrent execution of two tasks, hence, the response time (RT) will be the key in these measurements. In case that both parallel segments of τ_1 and τ_2 are allocated on GPU, we can see the system consumes less energy (at least 10% and up to 65%) compared to other allocations, although the RT of τ_2 gets almost two times longer than the stand alone version. Further, the RT of τ_2 in this case is shorter than the stand alone version (i.e., compared to WCET) of τ_2 when the parallel segment is allocated to CPU sequentially. This means that the proper use of GPU shows better results of both computing power and energy efficiency.

IV. Conclusion

In this paper, we have focused on the energy consumption and computing power of GPU accelerated in-orbit space systems. Further, both programming manner (how to program parallelization) and executing manner (where to allocate a parallel segment) are considered in the experiments. The experimental study shows that the execution part of HSA compliant GPU computes the calculation between 10 to 140 times faster and consumes between 8 to 130 times less energy, compared to the execution part of CPU-based (including single- and multi-core processors) calculations. The use of GPU is supported even when we consider all the parallel segments in systems as allocation of the workload to GPU is most energy efficient compared to the other allocations. Therefore, we conclude that GPU can be a high potential candidate in the on-board data processing of the small satellites.

For future work, we would like to continue developing a system with real-time GPU scheduler, which can dynamically allocate the tasks under limited power budget.

Acknowledgment

The work presented in this paper is supported by the Swedish Knowledge Foundation (KKS) through the research profile DPAC. We would like to express our sincere gratitude to Dr. Moris Behnam and Dr. Harris Gasparakis for sharing their great knowledge in real-time embedded systems and computer vision and HSA related areas.

References

[1] Master, T.: Consortium for Execution of Rendezvous and Servicing Operations (CONFERS), DARPA, https://www.darpa.mil/program/consortium-for-execution -of-rendezvous-and-servicing-operations, (accessed Apr 22, 2019).

[2] Bruhn, F., Brunberg, K., Hines, J., Asplund, L., and Norgren., M.: Introducing radiation tolerant heterogeneous computers for small satellites, 2015 IEEE Aerospace Conference, pp. 1–10, IEEE, 2015.

[3] Tsog, N., Behnam, M., Sjödin, M., and Bruhn, F.: Intelligent data processing using in-orbit advanced algorithms on heterogeneous system architecture, 2018 IEEE Aerospace Conference, pp 1–8, IEEE, March 2018.

[4] Tsog, N., Becker, M., Bruhn, F., Behnam, Sjödin, M.: Static Allocation of Parallel Tasks to Improve Schedulability in CPU-GPU Heterogeneous Real-Time Systems, IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, pp. 4516-4522, 2019.

[5] Perona, P. and Malik, J.: Scale-space and edge detection using anisotropic diffusion, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**, (1990), pp. 629–639.

[6] Lawrence Livermore National Laboratory: LULESH, https://computation.llnl.gov/projects/co-design/lulesh, (accessed April 21, 2019).

Run-Time Cache-Partition Controller for Multi-Core Systems

Jakob Danielsson[1], Marcus Jägemar[1,2], Moris Behnam[1], Tiberiu Seceleanu[1], Mikael Sjödin[1]
[1] Mälardalen University, Västerås, Sweden
[2] Ericsson AB, Kista, Sweden

***Abstract*—In this paper, we discuss the results of our Last-level cache partition controller called LLC-PC that uses the Palloc page coloring framework. LLC-PC generates initial cache partitioning directives for any given application through the Palloc interface and continiously re-sizes the cache partitions depending on how the application responded to the cache-partition size change. We have evaluated LLC-PC using 3 different applications, including the SIFT image processing algorithm which is commonly used for feature detection in vision systems. We show that LLC-PC is able to decrease the amount of cache size allocated to applications while maintaining their performance allowing more cache space to be allocated for other applications.**

I. Introduction

Multi-core systems are infamous for performance variations, which can become problematic in time-sensitive systems [3]. These variations often occur due to inter-core resource sharing, such as shared caches, shared memory bus, Translation Lookaside Buffers (TLB), shared DRAM-banks, and others. Sharing resources means that an application (e.g. app_0), executing on one core, does not have exclusive ownership of that particular resource, but instead shares the resource with another application, (e.g. app_1), executing on an adjacent core. Such scenario can lead to shared resource contention where app_0 unexpectedly stalls, since app_1 has access to the resource.

In recent years, several studies have proposed methods aiming to mitigate LLC contention through isolation. Some examples are cache partitioning which partition the LLC so that accesses from one application do not affect the performance of another [4]. An additional technique is cache locking [5], that forces applications to use only certain cache lines. Another example is cache scheduling [2] that schedules applications to minimize conflicts in the cache memory. Isolating the cache memory can however be a costly process in terms of lost memory space, since it is hard to determine how much cache memory should be allocated to a process. In this paper, we argue that there is a point of saturation, which is the point from where the performance of an application does not improve when given additional cache-partition space.

In this paper, we present the Last Level Cache-Partitioning Controller (LLC-PC), which finds the cache-partition saturation points. Our tool continuously reads the instructions retired event from the Performance Monitoring Unit (PMU) to estimate the application's performance and is, therefore, connected to an application ad-hoc. This paper focuses on the LLC, the methodology is, however, general and the PMU supports a broad set of events [8], and our method can be applied to other shared resources and partitioning techniques - to be investigated in the future.

II. Cache Partitioning - a Brief Introduction

LLC contention occurs when multiple applications compete for the same cache lines and can drastically degrade the execution time. Page-coloring, a.k.a. cache coloring [6] or cache partitioning, is a way of disqualifying applications from using certain cache lines. LLC partitioning in Linux can be done by replacing the standard Buddy allocator [7], forcing applications to take a subset of the total number of cache lines. Forming LLC partitions is often done by assigning colors to an application. The colors are then used to control where data requests from the physical memory should be put in the cache, see Fig. 1.

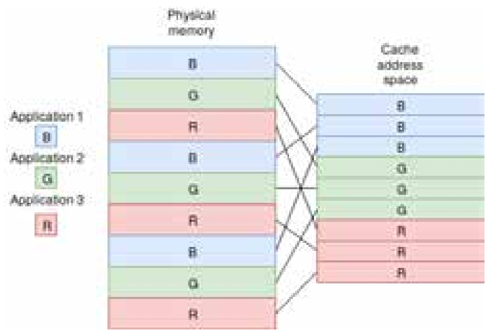


Fig. 1. Cache coloring

The figure shows three applications which split the cache memory equally. The applications are assigned three different colors in the physical memory which are then used to map memory rows to cache line locations, the colors are referenced using the set-associative bits of the LLC. We have used the combined DRAM-bank partitioning and LLC coloring tool called Palloc [7] to create LLC partitions. Palloc is a kernel module which runs partitions at the granularity of a page and replaces the regular Linux Buddy allocator with a colored page approach.
[1]

III. LLC-PC

We limit this paper to focus on applications that are bound to the Last Level Cache (LLC). LLC boundess means that the

[1]This paper is a curated paper for the DPAC newsletter Fall 2020, and the original of the paper is accepted in the Proceedings of IECON 2019 [1]

performance of the application is tightly correlated with the amount amount of cache miss, i.e., if the cache misses significantly increases, the performance will significantly decrease and vice versa. We present our partition controller logic in Fig. 2.

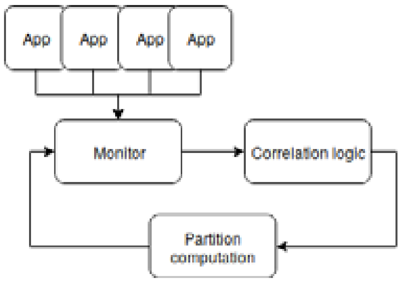


Fig. 2. LLC-PC

The cache controller is a correlation based control loop which regulates the cache partition size according to the correlation between a performance metric and the increase in cache size for a specific application. The controller will continuously increase the cache size for as long as the correlation between the increase in amount of cache partition size and the performance metric is high. Once the correlation starts to decline and reaches a certain threshold, an LLC saturation point has been found and the controller will stop assigning additional cache partitions to the specific application.

IV. Evaluation

We evaluate LLC-PC versus a static partition based solution which uses a LLC partition size of 16 and 32. We depict the execution flow of a 756x756 matrix multiplication in Fig. 3 and a 8MB SIFT execution in Fig. 4 using LLC-PC. The left-hand side y-axis of the graphs plots the median instructions retired (i.e., performance) per 50 milliseconds of the application using LLC-PC (blue squares), 16 statically assigned cache partitions (orange cross) and 32 statically assigned cache partitions (yellow plus). The right hand-side axis show the correlation over time using LLC-PC. A higher value on the left-hand side axis means more instructions executed per 50 milliseconds and is, therefore, better than a low value. The x-axis shows the number of partitions used, where a lower value is preferred since more cache partitions can be given to other applications.

Fig. 3 shows a full LLC-PC run of a 756x756 matrix multiplication, where the system saturates at 16 partitions, with comparable performance to that of the static partitions. For this particular matrix multiplication size, the static partition size was equal to the correlated size. Statically increasing the LLC sizes to 32 does not improve the matrix multiplication performance significantly. Furthermore, Fig. 4 show SIFT operating within the LLC-PC, with a final assignment of 13 LLC partitions.

V. Summary

We have implemented a correlation based LLC partition controller, called LLC-PC, which can be used to find LLC partition sizes for workloads with unknown cache usage. We evaluate LLC-PC using two LLC heavy loads, a Matrix multiplication, and a SIFT feature detection algorithm. The results show that LLC-PC can be used for this set of workloads to reduce the amount of cache size given to an algorithm compared to a static 32 cache LLC partition assignment, and also in most cases a 16 LLC partition assignment - while still maintaining similar performance.

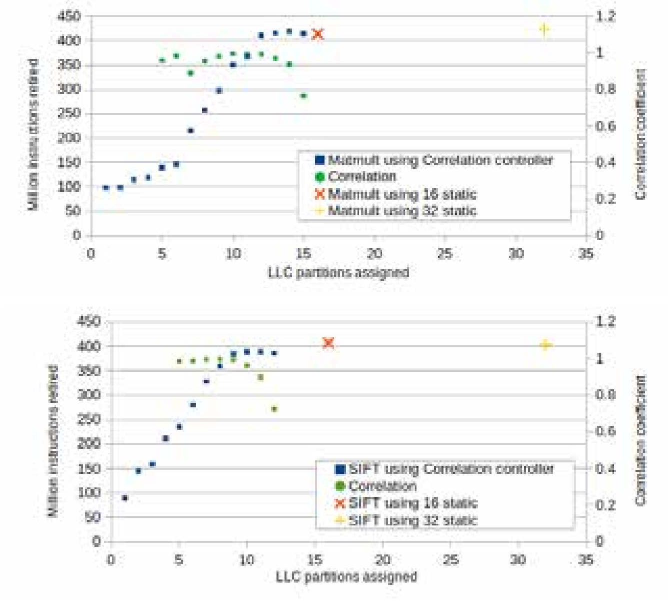


Fig. 4. Comparison of 8MB SIFT executions

References

[1] Jakob Danielsson, Marcus Jägemar, Moris Behnam, Tiberiu Seceleanu, and Mikael Sjödin. Run-time cache-partition controller for multi-core systems. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1.

[2] Nan Guan, Martin Stigge, Wang Yi, and Ge Yu. Cache-aware scheduling and analysis for multicores. In *Proceedings of the seventh ACM international conference on Embedded software*, pages 245–254, 2009.

[3] Abdelhafid Mazouz, Denis Barthou, et al. Study of variations of native program execution times on multi-core architectures. In *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, pages 919–924. IEEE, 2010.

[4] G Edward Suh, Larry Rudolph, and Srinivas Devadas. Dynamic partitioning of shared cache memory. *The Journal of Supercomputing*, 28(1):7–26, 2004.

[5] Xavier Vera, Björn Lisper, and Jingling Xue. Data cache locking for higher program predictability. In *ACM SIGMETRICS Performance Evaluation Review*, volume 31, pages 272–282, 2003.

[6] Ying Ye, Richard West, Zhuoqun Cheng, and Ye Li. Coloris: a dynamic cache partitioning system using page coloring. In *Parallel Architecture and Compilation Techniques (PACT), 2014 23rd International Conference on*, pages 381–392. IEEE.

[7] Heechul Yun, Renato Mancuso, Zheng-Pei Wu, and Rodolfo Pellizzoni. Palloc: Dram bank-aware memory allocator for performance isolation on multicore platforms. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2014 IEEE 20th*, pages 155–166. IEEE, 2014.

[8] Gerd Zellweger, Denny Lin, and Timothy Roscoe. So many performance events , so little time. *APSys '16*, 2016.

Planning and Supervising Autonomous Underwater Vehicles through the Mission Management Tool

Afshin Ameri E[1]., Baran Cürüklü, Branko Miloradović & Mikael Ektröm
School of Innovation, Design and Engineering, Mälardalen University, Västerås Sweden
{afshin.ameri, baran.curuklu, branko.miloradevic, mikael.ekstrom}@mdh.se

***Abstract*—Complex underwater missions involving heterogeneous groups of Autonomous Underwater Vehicles (AUVs) and other types of vehicles require a number of steps: defining and planning a mission, orchestration during the mission execution, recovery of the vehicles post-mission, and finally analysis of data collected during the mission. In this work the Mission Management Tool (MMT), a software solution offering the above-mentioned services is proposed. The MMT hides the complex system consisting of software solutions, hardware, and vehicles from the user, and allows intuitive interaction with the vehicles involved in a mission. The tool can adapt to a wide spectrum of missions assuming different types of robotic systems and mission objectives. As demonstrated in the real-world tests the MMT is able to support the mission operators.**

I. Introduction

The seas and the oceans cover 70% of the planet. These habitats are host to large industries and economical activities such as gas and oil, transportation, shipping, marine food production, tourism, and other service-oriented businesses and secondary sectors [1]. In many of these industries, operations that are carried out at the sea may potentially affect the marine habitats and the coastlines [2]. In some cases, the consequences of these operations can be critical, e.g. when the outcome is an oil spill, a leak of wastewater, or a problem associated with an underwater construction, which needs detailed inspection or even replacement of a construction part. During such critical events, seamless collaboration within a distributed team of humans and robotic agents is critical. The latter consists of Autonomous Underwater Vehicles (AUVs) and/or Remotely Operated Underwater Vehicles (ROVs). The Mission Management Tool (MMT) is a human multi-robot interaction software aiming at extending the abilities of current robotic solutions for underwater missions.

The main capabilities of the MMT are: (i) defining and planning a mission; (ii) monitoring a mission; (iii) supervision of a mission with the objective of affecting the mission's progress if required; and finally; (iv) post-mission data analysis. Similar to other GUI-based mission operation software, the MMT can support the operators, and hence hypothetically improve the overall performance of a mission, reduce the need for human presence, i.e. divers, at the mission site, which results in less risk-taking, and improved working conditions.

The MMT has been developed and demonstrated as part of the Horizon 2020/ECSEL JU European projects SWARMs (2015-18) and AFarCloud (2018-21), the latter aims at autonomous solutions in precision agriculture through drones, autonomous tractors and other types of ground vehicles.

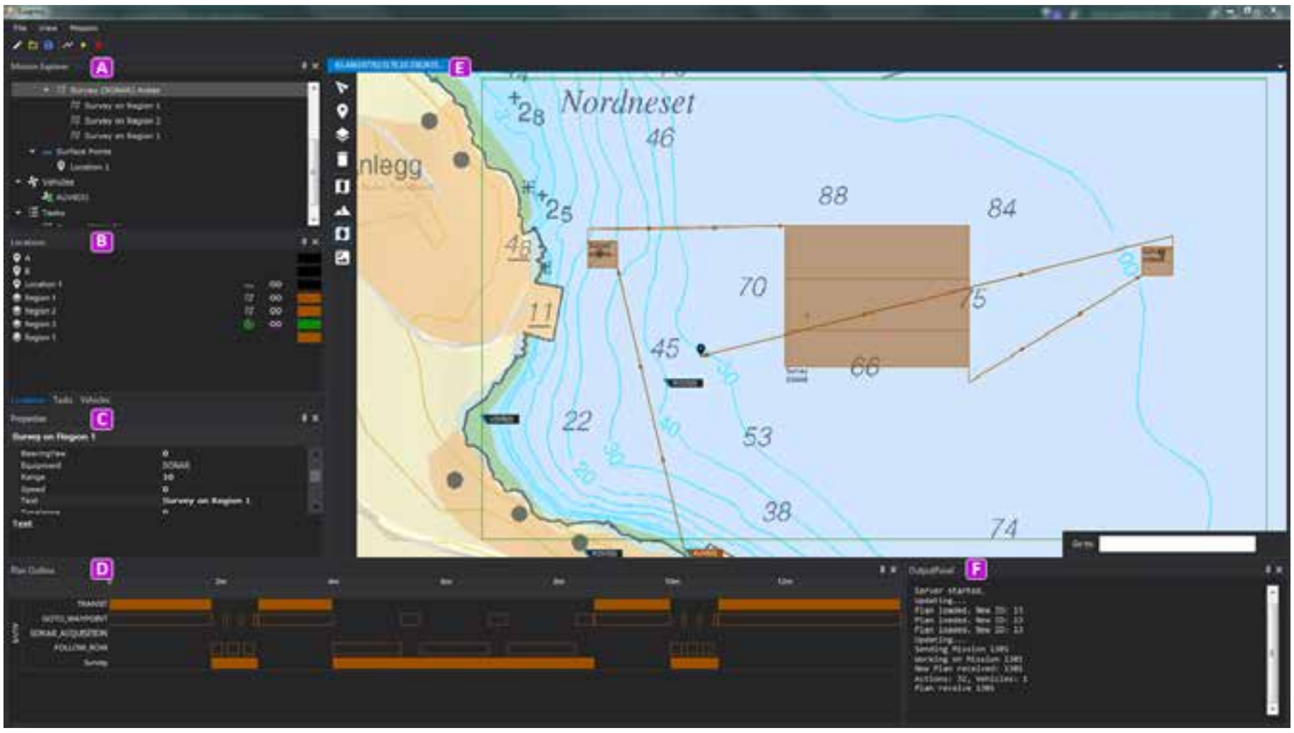
From the perspective of the operator, the MMT's core services are defining, planning, monitoring, controlling, and analyzing missions:

- **Defining a mission:** An area for the mission site is marked in the map (Fig. 1.E). Typically, during the mission, one or more AUV activities will be performed in the mission area. These tasks are directly based on vehicle capabilities and available sensors.
- **Planning a mission:** In the MMT the operator can define mission plans on a high-level, which later can be interpreted by the vehicles' onboard planners. In the MMT the result of the high-level planning can be presented on the map and as a Gantt chart in the Plan Outline Panel (Fig. 1.D).
- **Monitoring a mission:** Monitoring is done using two main means, i.e. the Gantt chart at the bottom window of the MMT Graphical User Interface (GUI), and the large window in the middle showing the map. The former's objective is to show the progress of the mission, whereas the latter provides spatial information such as the location of the vehicles.
- **Supervising a mission:** One critical case is when the mission needs to be halted or terminated due to an unexpected event. Also, during the normal progress of the mission there may be a need for interaction between the operator and any of the vehicles.
- **Analyzing a mission:** From the moment a mission starts the middleware (not described here) starts to collect data for analysis during and post-mission.

II. Experimental Results

The overall purpose of the experiments was to demonstrate that the proposed MMT-based solution can support the operators regarding preparing, planning, execution, and monitoring a mission, followed by recovering the deployed AUVs. The experiments took place in the waters of the Trondheim fjord, Norway (Coordinates: Lat. 63.4430, Lng. 10.3651).

[1] This paper is a curated paper for the DPAC newsletter Fall 2020. See [3] for the original.



```
Server started.
Updating...
Plan loaded. New ID: 11
Plan loaded. New ID: 13
Plan loaded. New ID: 13
Updating...
Sending Mission 1301
Working on Mission 1301
New Plan received: 1301
Actions: 32, Vehicles: 1
Plan receive 1301
```

Figure 1. The Mission management tool overview showing the chosen mission plan.

The MMT was evaluated through a set of demonstrations, which were detection, inspection, and tracking of plumes of wastewater, at different depths. The plumes are identified by change in water salinity. They were simulated during the mission, due to environmental considerations. In the experiment three different depths were surveyed, i.e. 30, 35, and 40 meters. The AUVs send status vectors including location (latitude, longitude, and depth), battery level, task status, and sensor readings to the middleware. This allowed the MMT to fetch this data and present it to the operator. Four vehicles were involved in this mission. In order to plan the mission, the operator executes the High-Level Planners (HLP). The planning process involved defining the mission area and the type of the task to be performed (in this case scanning an area using salinity sensors). The High-Level planner then chooses the vehicles based on their capabilities and assigns different tasks to each vehicle.

The sub windows of the MMT are marked in Fig. 1: (A) Mission Explorer: The central place for defining the relationship between different mission components (locations, vehicles, tasks), (B) Asset Panel: Is a combination of three panels groups together. They are locations, vehicles and tasks. This is where the operator can see and interact will all mission related assets (locations/vehicles/tasks), (C) Properties Panel: Lists all the properties for the selected item, (D) Plan Outline: Shows the plan as a Gantt chart, where the timeline of tasks of easy AUV is shown. (E) Map, (F) Console: Text messages from the system are displayed here.

The mission was performed on 3 different occasions. During one of the runs, one of the AUVs sent a propulsion error message and aborted its mission and surfaced. The error and the surface extraction point were received at MMT and helped the extraction team to recover the AUV. It was later discovered that the reason for the error was algae which were stuck in the propeller. Upon receiving the error message on MMT, the operator had the option to cancel the mission, re-plan it, or continue with the mission with one less AUV. The operator in this case chose to continue the mission and no re-planning was performed.

III. Conclusion

The experiments have shown that MMT is capable of preparing, planning, executing, and supervising multi-robot missions. The planning algorithms integrated into MMT also made it easy for the operator to define their set of goals while leaving the actual planning to the planning algorithms. The map and the plan outline kept the operator aware of the vehicles' and mission status by visualizing the information in different forms. The AUVs detected the plum, which was the main goal of the mission. The real-world experiments have shown that a complex system consisting of several heterogeneous robotic agents can be orchestrated by the MMT, which can be adapted to different allocation domains.

References

[1] Fernández-Macho, J., González, P., & Virto, J. (2016). An index to assess maritime importance in the European Atlantic economy. *Marine Policy*, *64*, 72-81.

[2] Branch, A., & Stopford, M. (2013). *Maritime economics*. Routledge.

[3] Ameri A., Cürüklü, B., Miloradović, B., & Ektröm , M. (2020) Planning and Supervising Autonomous Underwater Vehicles through the Mission Management Tool. Global OCEANS 2020 OCEANS, 05 Oct 2020, US Gulf Coast, United States.

A Dependable Multi-Path Planning with Congestion and Obstacle Avoidance for Multiple Autonomous Robots

Lan Anh Trinh, Mikael Ekström and Baran Cürüklü

***Abstract*— This paper presents a novel path planning for multiple autonomous robots working in a complex workspace along with the presence of humans and other unpredictable moving objects. Each robot has different options, i.e. global paths, to choose between to reach its defined goal. All the planned paths are shared among the robots, and combined with obstacle and congestion avoidance constraints to form as an optimisation problem. The solution for the problem provides an optimal and safe moving plan for all robots. The effectiveness of the proposed approach is demonstrated by experimental results.**

Keywords*— multiple path planning; multiple robot; congestion avoidance; collision avoidance

I. Overview

As of yet, traffic rules have been applied in transportation systems to control movements of people and vehicles to ensure safety and efficiency. One of the crucial aims is to prevent accidents by avoiding collisions among traffic participants. A path planning algorithm of an autonomous robot is developed by learning more or less from such systems. However, as moving behaviours of the traffic participants are independent and not controlled in an optimised manner, the congestion could happen with heavy traffics in a small area. It is different with a group of working robots when nowadays they are well equipped with communication infrastructure, which can be utilised to develop a better dependable, i.e. safe, reliable, and effective, navigation system. This work has proposed a new path planning algorithm by combining obstacle avoidance and congestion control in an optimisation framework using sharing information among robots. Each robot establishes multiple paths from a start to a goal. The constraints of not routing many robots into a potential conflict areas are applied for congestion control. This is feasible as every robot has alternative options (paths) to travel. The collision avoidance is defined using velocity obstacles (VOs). The whole system is implemented in a well-known platform, robot operating system (ROS) [1], in a centralised manner. A center node collects information from all robots, finds optimal path by solving an optimisation problem, and shares the moving plans back to every robot.

II. Proposed Approach

It is assumed that there are n robotic agents in the working space, denoted by $\mathcal{A} = \{i | i \in 1, 2, ..., n\}$. A set of moving obstacles, detected by the agent i, is then denoted $\mathcal{O}_j = \{j | j \in 1, 2, ..., n_i\}$. The footprint of a robot i is modelled as a closed disk with the radius r_i.

This paper is a curated paper for the DPAC Newsletter 2020 and is derived from the original paper [3] published in IEEE ICCAR 2020.

To check for the collisions among robots and moving obstacles in a local range, the concept of velocity obstacle is utilised. Given two robots with position $\mathbf{a}_A$ and $\mathbf{a}_B$, a set of relative reference velocities $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$ leading to the collision within time τ is given by,

$$\mathcal{VO}^{\tau}_{AB} = \{\mathbf{v}_{AB} | \forall t \in [0, \tau], \|\mathbf{a}_A - \mathbf{a}_B + t\mathbf{v}_{AB}\| \le r_A + r_B\}. \tag{1}$$

From the current position $\mathbf{a}_i(t)$ of the robot i, there are available p_i paths to its goal. Let $\mathcal{V}_i = \{k | k \in 1, 2, ..., p_i\}$ be a set of the available paths for robot i, $\mathcal{P}_i = \{\bar{\mathbf{v}}_i^1, \bar{\mathbf{v}}_i^2, ..., \bar{\mathbf{v}}_i^{p_i}\}$ be a set of preferred velocities on each path.

Let $\mathbf{z}_i = [z_i^1, z_i^2, ..., z_i^{p_i}]^T$ be the binary vector to select the path, $z_i^k \in \{0, 1\}$. The optimisation cost function $C_i(\mathbf{v}_i, \mathbf{z}_i)$ is defined as follows:

$$\begin{aligned} C_i(\mathbf{v}_i, \mathbf{z}_i) &= \|\mathbf{v}_i - \sum_{k=1}^{p_i} \bar{w}_i^k z_i^k \bar{\mathbf{v}}_i^k\|^2 + \sum_{k=1}^{p_i} z_i^k \bar{s}_i^k \\ \text{s. t.} \quad & \sum_{k=1}^{p_i} z_i^k = 1 \end{aligned} \tag{2}$$

where $\bar{w}_i^k$ and $\bar{s}_i^k$ are the weights and the travelled lengths of the path.

The joint optimisation function for all robots to find their optimal paths and velocities is expressed by,

$$\begin{aligned} C(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n) &= \sum_{i=1}^{n} C_i(\mathbf{v}_i, \mathbf{z}_i) \\ \text{s. t.} \quad \sum_{k=1}^{p_i} z_i^k &= 1, \forall i \in [1, n]. \end{aligned} \tag{3}$$

To find collision-free paths as well as to avoid dynamic obstacles, a set of constraints are defined as below:

- Multi-path conflict-free constraints

Assume that two robots A and B are configured with multiple paths to goals (Fig. 1).

$$\begin{aligned} \mathcal{CF} = \{z_i^k + z_j^l \le 1 | \forall i, j \in \mathcal{A}, k \in \mathcal{V}_i, l \in \mathcal{V}_j, \\ \text{if there is a potential conflict when robot } i \\ \text{chooses the path } k \text{, and } j \text{ chooses } l\}. \end{aligned} \tag{4}$$

- Moving obstacle avoidance constraints

According to the definition of VO, the collision between robot A and another robot or a moving obstacle B is avoided if $\mathbf{v}_A - \mathbf{v}_B \notin \mathcal{VO}^{\tau}_{AB}$.

Finally, the overall optimisation problem is formulated, in which the optimal control velocities and selected global paths $[\mathbf{v}^*_{1:n}, \mathbf{z}^*_{1:n}] = [\mathbf{v}^*_1, \mathbf{v}^*_2, ..., \mathbf{v}^*_n, \mathbf{z}^*_1, \mathbf{z}^*_2, ..., \mathbf{z}^*_n]$ are

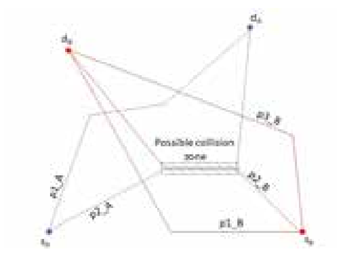


Fig. 1. An example of creating multiple paths for two moving robots with a possible collision zone.

estimated in a joint manner,

$$
\begin{aligned}
[\mathbf{v}^*_{1:n}, \mathbf{z}^*_{1:n}] &= \underset{[\mathbf{v}_{1:n}, \mathbf{z}_{1:n}]}{\arg\min}\, C(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n) \\
\text{s. t.} \quad & \sum_{k=1}^{p_i} z_i^k = 1, \forall i \in [1, n] \\
& \mathcal{CF} \qquad \text{(Constraint 1)} \\
& \mathbf{v}_i, \mathbf{v}_j \notin \mathcal{VO}^{\tau}_{ij}, \forall i, j \in \mathcal{A} \qquad \text{(Constraint 2)} \\
& \mathbf{v}_i \notin \mathcal{VO}^{\tau}_{io_j}, \forall i \in \mathcal{A}, j \in \mathcal{O}_i .
\end{aligned} \tag{5}
$$

III. EXPERIMENTS

Two different scenarios are used to evaluate the proposed approach.

A. *Two Robots Crossing Narrow Corridor Scenario*

Two robots are located at two different sides of a corridor (Fig. 2). Two different paths are established for each robot, the red and green ones. The yellow paths are the actual moving trajectories of the two robots. Congestion avoidance helps preventing dead-locks of two moving robots.

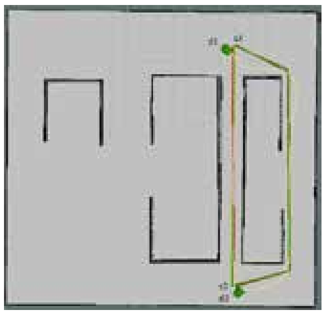


Fig. 2. Moving trajectories of two robots.

B. *Scenario with Robots/Humans Together*

Similarly, three robots are located randomly in the working space. Each robot also has two alternative paths to reach its goal. In this scenario, two human actors are added with predefined moving trajectories (Fig. 3). The effectiveness of the algorithm is shown in the Fig. 4.

For quantitative evaluation, Table I shows a comparison between the proposed algorithm and twos others, dynamic window approach (DWA) and DWA+VOs [2]. Note that a collision happens if the minimum distance is less than 0.5 meters.

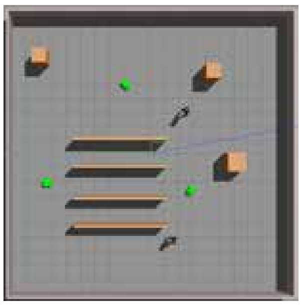


Fig. 3. The simulated working space with three robots and humans.

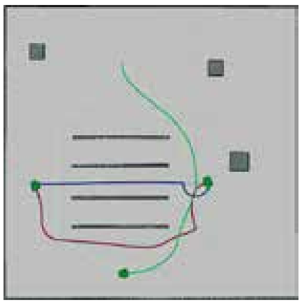


Fig. 4. Moving trajectories of robots co-existing with humans. The thin trajectories are the two planned path while the bold ones are the actual moving trajectories of robots.

IV. CONCLUSIONS

This paper presents a novel multiple path planning approach where the simulation results have shown that the proposed method is effective and safe to avoid collisions and dead-lock situations caused by congestion of multiple robots. In the future, a decentralised manner will be investigated to mitigate the dependency of robots to one central node, which consequently improves the fault tolerance of the whole system. Also, an extensive evaluation with real robots will be planned.

ACKNOWLEDGMENT

The research leading to the presented results has been undertaken within the research profile DPAC - Dependable Platform for Autonomous Systems and Control, funded by the Swedish Knowledge Foundation.

REFERENCES

[1] https://www.ros.org/.
[2] D. Claes and K. Tuyls, Multi robot collision avoidance in a shared workspace, Autonomous Robots, Springer, 2018.
[3] L.A. Trinh, M. Ekström, and B. Çürüklü, Multi-path planning for autonomous navigation of multiple robots in a shared workspace with humans, IEEE ICCAR, 2020.

TABLE I
PERFORMANCE OF THE PROPOSED APPROACH WITH OTHER WORKS.

	Distance (m)	Collisions	Dead-locks
Multi-path planning	0.56	0	0
DWA	0.21	3	5
DWA + VOs [2]	0.57	0	3

Verifiable and Scalable Mission-Plan Synthesis for Autonomous Agents

Rong Gu, Eduard Enoiu, Cristina Seceleanu, and Kristina Lundqvist
Mälardalen University, Västerås, Sweden
Email: (first.last)@mdh.se

***Abstract*—Synthesizing mission plans for multiple autonomous agents, including path planning and task scheduling, is often complex. In this paper, we propose a novel approach called MCRL that integrates model checking and reinforcement learning to solve this problem. Our approach employs timed automata and timed computation tree logic to describe the autonomous agents' behavior and requirements, and a reinforcement learning algorithm, namely Q-learning, to train the agent model, in order to generate mission plans that satisfy the requirements. Our method provides a means to synthesize mission plans for multi-agent systems whose complexity exceeds the scalability boundaries of exhaustive model checking, but also to analyze and verify synthesized mission plans to ensure given requirements. We evaluate the proposed method on various scenarios involving autonomous agents, as well as present comparisons with two similar approaches, TAMAA and UPPAAL STRATEGO. The evaluation shows that MCRL performs better for a number of agents greater than three.**

I. Introduction

Autonomous agents are systems that usually move and operate in a possibly unpredictable environment, can sense and act on it, over time, while pursuing their goals. Figure 1 depicts an example of an autonomous quarry, where different kinds of agents are deployed. Autonomous wheel loaders

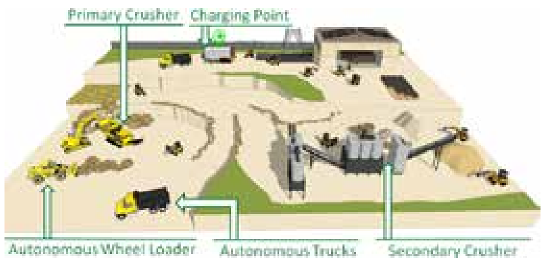


Fig. 1. An example of autonomous quarry

(AWL) working near stone piles dig stones and load them into autonomous trucks (AT). The latter carry and unload the stones into the primary crushers, where the stones are crushed into fractions, before AWL load the crushed stones into the AT. Finally, AT transfer and unload the stones into the secondary crusher. In case the battery level of an AWL becomes low, the former must go to the charging point timely. During this process, the AWL and AT sometimes work independently, sometimes cooperate, and eventually accomplish the entire mission, i.e., transferring all the stones to the secondary crusher. Hence, the AWL and AT constitute a multi-agent system (MAS), and in order to realize their functions, this system needs *mission planning*, including path planning and task scheduling. As path-planning algorithms focus just on calculating collision-free paths towards the destination, they do not cover requirements concerning logical and temporal constraints, e.g., delivering goods in a given order, and within a certain time limit. In addition, when considering a group of agents that need to collaborate with each other and usually work alongside humans, the job of synthesizing correctness-guaranteed mission plans becomes crucial and more difficult.

In order to solve this problem and verify the synthesized mission plans, we propose a novel approach that employs formal methods, advanced path-planning and collision-avoidance algorithms, as well as reinforcement learning. In our previous work [1], we have proposed an approach based on Timed Automata (TA) and Timed Computation Tree Logic (TCTL) to formally capture the agents' behavior and requirements, respectively, and synthesize mission plans for autonomous agents, by model checking. Our approach is successfully implemented in a tool called TAMAA, and shown to be applicable to solving the mission-planning problem of industrial autonomous agents. However, TAMAA alone does not scale for a large number of agents, as the state space of the model explodes when the number of agents grows.

In this paper[1], we propose a novel method called **MCRL** that combines model checking with reinforcement learning to restrict the state space, in order to synthesize mission plans for large numbers of agents. Our method is based on UPPAAL [2] and leverages the model of autonomous agents generated by TAMAA. Instead of exhaustively exploring the states of the model, MCRL uses random simulations and Q-learning to train the model in order to synthesize mission plans. Evaluation of the methods are conducted on an industrial use case: an autonomous quarry. Details are in the following section.

II. A Walk Through the Methods and Experimental Results

TA-Based Model for Mission-Plan Synthesis. The model of an agent has three component: (i) a TA of agent movement, (ii) a TA of task execution, and (iii) a TA for monitoring the variation of important signals, e.g., fuel level, and warning the agent when a signal passes its threshold. These TA can be generated automatically by our tool named TAMAA, once users configure the environment and agents [1]. After the model is

[1]This paper is a curated paper for the DPAC newsletter Fall 2020, and the original of the paper is published in FMICS 2020. Best-paper award.

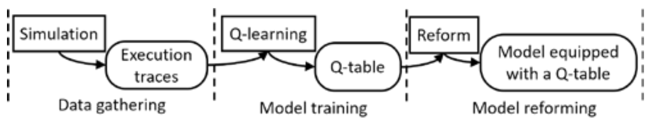


Fig. 2. The process of creating a model using a Q-table

generated, they are verified in UPPAAL against the formalized requirement (TCTL queries). However, the verification is based on exhaustive model checking, which means that the entire state space is built and stored during the process. The number of states of the model grows exponentially with the number of agents, hence the computation time and memory consumption increase dramatically.

A Novel Method of Model Checking + Reinforcement Learning (MCRL). Figure 2 depicts the process of MCRL. First, in the *data-gathering* phase, we obtain the execution traces of the model by the simulation function in UPPAAL. We assign rewards to the state-action pairs of the execution traces that satisfy the desired properties, and penalties to the ones containing deadlocks. The traces that neither hold the properties nor contain deadlocks are ignored and not used in the next phase.

In the *model-training* phase, we adopt Q-learning to process the traces and populate a Q-table, which is then used to form a new model whose state space is restricted. Q-learning is an algorithm that simulates the agents or agent models, and during the simulation, it computes the rewards the agents get from the environment and accumulates the rewards in Q-tables. Therefore, the next time when an agent comes to the same situation, it makes wiser choices by looking up the Q-table and choosing the action with the highest reward. The equation used in Q-learning guarantees that the accumulated rewards consider the current and future value of the action. Practically, the *model-training* phase is implemented as a Java program, where the execution traces (i.e., text files), are parsed and used as the input of the Q-learning algorithm. Finally, the Q-tables are produced by the Java program and injected manually into the original model of agents.

In the *model-reforming* phase, a new TA model, which we call *conductor*, is designed for each of the agents; it looks up the agent's Q-table and sends controlling commands. Since there is no centralized control in the environment, each agent model is equipped with one conductor, respectively. The conductor contains the Q-tables of all agents in order to decide which one has the priority to act, when multiple agents intend to perform some concurrent action.

Experimental Results. In our work [3], we show the verification results of the synthesized mission plans. The requirements are categorized into four types, and four types of TCTL queries are designed correspondingly, which are shown below. In these queries, $\texttt{te}_n$ and $\texttt{move}_n$ are the task execution TA and movement TA of agent n, respectively. Variable `tasks` is a two-dimensional integer array of agents' task execution status, e.g., finished, or unfinished, `event` is a two-dimensional Boolean array of events' status, and `x` is a clock variable.

- *Milestone Matching*. Query (1) checks that agent n is always at milestone P_i, when it is executing task T_i.

$$A\Box\ (\texttt{te}_n.\texttt{T}_i\ \texttt{imply}\ \texttt{move}_n.\texttt{P}_i) \tag{1}$$

- *Task Sequencing*. Query (2) checks if agent n's precedent task is always finished, when agent n is executing task T_i.

$$A\Box\ (\texttt{te}_n.\texttt{T}_i\ \texttt{imply tasks[n][i-1]==FIN}) \tag{2}$$

- *Timing*. Query (3) checks if agent n can always finish all its tasks within TL time units, where `i` is the index of a task, `M` is the number of tasks, and `TL` is an integer of time limit.

$$A\Box\,(\forall i \in \texttt{[0,M-1]}\ \ \texttt{(tasks[n][i]==FIN imply x}\leq\texttt{TL))} \tag{3}$$

- *Event Reaction*. Query (4) checks if agent i can always reach milestone P_k within EL time units if event j occurs.

$$\texttt{event[i][j] -->}\ (\texttt{move}_n.\texttt{P}_k\ \&\&\ \texttt{x} \leq \texttt{EL}) \tag{4}$$

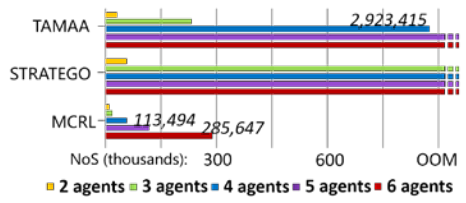


Fig. 3. The number of explored states

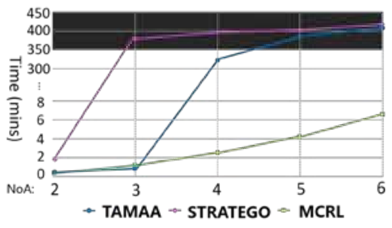


Fig. 4. The computation time of mission plan synthesis

To show the outstanding performance of MCRL, we compare MCRL with TAMAA and UPPAAL STRATEGO. The result is depicted in Figures 3 and 4, which show that the number of explored states and the computation time of MCRL increases linearly as the number of agents grows, whereas the other two methods raise exponentially.

III. Conclusion

To synthesize mission plans for large numbers of autonomous agents and verify them formally, we propose a method named MCRL. The method combines model checking with reinforcement learning. We compare the new method with TAMAA and UPPAAL STRATEGO. The experimental results show that MCRL's performance is better than the other two tools when the number of agents is more than 5.

References

[1] R. Gu, E. P. Enoiu, and C. Seceleanu, "Tamaa: Uppaal-based mission planning for autonomous agents," in *SAC 2020*.

[2] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," in *Formal methods for the design of real-time systems*. Springer, 2004.

[3] R. Gu, E. Enoiu, C. Seceleanu, and K. Lundqvist, "Verifiable and scalable mission-plan synthesis for autonomous agents," in *FMICS 2020*. Springer.

An Actor-based Approach for Security Analysis of Cyber-Physical Systems

Fereidoun Moradi*, Sara Abbaspour Asadollah*, Ali Sedaghatbaf*,
Aida Čaušević*, Marjan Sirjani*, and Carolyn Talcott†
* IDT, Mälardalen University, Västerås, Sweden and †SRI International, Menlo Park, CA, USA
{fereidoun.moradi, sara.abbaspour, ali.sedaghatbaf, aida.causevic, marjan.sirjani}@mdh.se
clt@csl.sri.com

***Abstract*—In this work, we present an actor-based approach for security analysis of Cyber-Physical Systems at the design phase. We use Timed Rebeca, an actor-based modeling language, to model the behavior of components and potential attacks, and verify the security properties using Rebeca model checking tool. We employ STRIDE model as a reference for classifying the attacks. To demonstrate the applicability of our approach, we use a Secure Water Treatment (SWaT) system as a case study. We analyze the architecture of the SWaT system using three different attack schemes in which various parts of the system network and physical devices are compromised. In the end, we identify single and combined attack scenarios that violate security properties.**

I. Introduction

Rebeca is an actor-based modeling language with formal semantics that makes it suitable for model checking purposes. Timed Rebeca [2] is an extension of Rebeca where computation time and network delay can be modeled. Timed Rebeca is supported by a model checking tool suite Afra [3] and can be used for verifying Cyber-Physical Systems (CPS). STRIDE [4] is a model for identifying different types of threats that a system may experience and the corresponding security objective which might be violated.

II. Methodology

The proposed method for CPS security analysis includes the following steps: (1) the Rebeca model of the CPS is developed from the system design specifications, (2) the potential attack scenarios against the system are modeled, (3) the security properties are defined in terms of assertions or temporal logic, and (4) Afra is used to identify the events trace that leads to a security failure. The above steps are elaborated in the following subsections.

A. Building the Rebeca model of the Cyber-Physical System

We consider each CPS component and physical processes as an actor. We realise four types of actors in our Rebeca model, controllers, sensors, actuators and physical processes. The Rebeca model of a CPS includes reactive classes describing the behavior of each actor.

This paper is a curated paper for the DPAC newsletter Fall 2020, and the original of the paper is accepted in FMICS20, Vienna, Austria [1].

B. Attack Modeling

According to the malicious behaviour on communication channels and components three cases are considered as follows: ***Attack on Communication***: attacker targets the communication channel between two components through injecting malicious messages, ***Attack on Components***: attacker manipulates the internal behavior of one or more components e.g. through malicious code injection, and ***Combined Attack***: one or more attackers perform a coordinated attack to launch malicious behaviour on both the communication channels and the components.

C. Model Checking and Security Analysis

The most important security objectives are *confidentiality*, *integrity* and *availability*. We use Afra tool [3] to automatically verify each of the specified security properties. The security objectives will be the basis for defining the security properties to be verified. Afra supports Linear Temporal Logic (LTL), Timed Computation Tree Logic (TCTL) and assertions for property specification. If Afra detects that a property is not satisfied by the Rebeca model, it provides the modeler with a counter-example detailing the sequence of events that would lead to a security violation. The sequence of events determines a successful attack.

III. The SWaT Case Study and Evaluation

We discuss an experimental study on the SWaT testbed [5]. We present the SWaT actor model and its security objectives. Then, we provide details on the security analysis results. The SWaT testbed is a scaled-down version of an industrial water treatment system. This testbed is used for several research and training purposes in the iTrust research center [5].

A. SWaT Actor Model

The actor model of the SWaT system is depicted in Figure 1. In this model, each shape represents an actor which corresponds to a component in the SWaT abstract architecture. Each arrow models a message passed between two components. In the model, the messages that may be the targets of attackers are distinguished from the secure ones. The red points with numbers from one to six indicate the possible compromised channels where the attackers may inject messages. The compromised channels are due to the lack of strong authentication and tamper-resistant mechanisms.

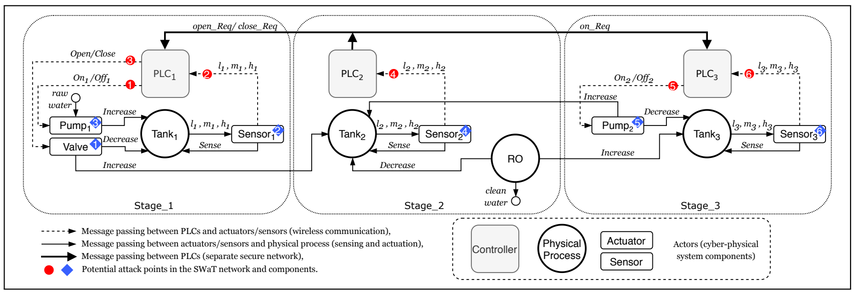

open_Req/ close_Req
on_Req
Open/Close
On1 /Off1
On2 /Off2
PLC1
PLC2
PLC3
l1 , m1 , h1
l2 , m2 , h2
l3, m3 , h3
raw water
Pump1
Valve
Pump2
Increase
Decrease
Tank1
Tank2
Tank3
Sensor1
Sensor2
Sensor3
Sense
RO
clean water
Stage_1
Stage_2
Stage_3
Message passing between PLCs and actuators/sensors (wireless communication),
Message passing between actuators/sensors and physical process (sensing and actuation),
Message passing between PLCs (separate secure network),
Potential attack points in the SWaT network and components.
Controller
Physical Process
Actuator
Sensor
Actors (cyber-physical system components)

Figure 1: SWaT actor model.

B. Security objectives and Threats

We assume that malicious attackers have the ability of injecting arbitrary packets into the communication channels between PLCs and sensors/actuators, and also they are able to alter the functionality of sensors/actuators.

In this experiment we focus on the *integrity* of SWaT system following the STRIDE model. In fact, we use model checking to detect the undesirable events that might happen while attackers tamper the channels (e.g., by injecting packets) and compromise sensors/actuators by altering their functionality (e.g., physical attack).

C. The Rebeca Model of the SWaT System

We provide a brief explanation of the Rebeca model developed for the SWaT system. The complete model is available in [6]. Listing 1 shows an abstract view of the SWaT Rebeca model. The main block includes the declarations of all actors defined in the SWaT actor model (see Figure 1) together with an attacker actor. In each declaration, the first parameter list includes the known actors, those which the declared actor communicates with. For example, the known actors of PLC_1 are $Pump_1$, Valve and $Sensor_1$. The second parameter list includes the parameters to be passed to the constructor of the actor.

```
1   reactiveclass PLC1(5){...}
2   reactiveclass PLC2(5){...}
3   reactiveclass PLC3(5){...}
4   reactiveclass Tank1(10){...}
5   ...
6   reactiveclass Attacker(3){
7       knownrebecs{...}
8       Attacker(int chl, int maliciousMsg, int attackTime){
9           if (chl == 1)
10              {self.channelPlc1P1(maliciousMsg, attackTime);
11          } else if (chl == 2)
12              {self.channelPlc1S(maliciousMsg, attackTime);
13          } else {...}}
14      msgsrv channelPlc1P1(int msg, int attackTime){
15          if(msg == 1)
16              {pump1.on() after(attackTime);
17          } else if(msg == 0)
18              {pump1.off() after(attackTime);}}
19      msgsrv channelPlc1S(int msg, int attackTime){
20          plc1.processSensorData(msg) after(attackTime);}
21      ... //message servers
22  }
23  main{
24      PLC1 plc1(pump1,valve,sensor1):();
25      PLC2 plc2(plc1,plc3,sensor2):();
26      PLC3 plc3(pump2,tank3,sensor3):();
27      Tank1 tank1(sensor1):();
28      ...
29      Attacker attacker(plc1,plc2,plc3,pump1,pump2,valve):
30                          (chl,malMsg,attackTime);
31  }
```

Listing 1: An abstract version of the SWaT Rebeca model.

D. Attack Models in Rebeca

The constructor of *Attacker* (see line 8 in Listing 1) class has three arguments representing the target channel, malicious message content, and attack time. Based on the value passed to the target channel argument, the message server responsible for sending malicious messages to the corresponding channel is invoked by the constructor. Message content is another numeric argument whose value indicates either the water level in a tank, an *on/off* command for Pump, or an *open/close* command for Valve. Finally, the third argument represents the time during the system operation that the malicious message is sent to a channel.

In addition, we model compromised actors using two parameters that are passed to all the actors that can be compromised. The first parameter sets the status of the actor, and the second parameter sets the time of the attack.

E. Model Checking and Security Analysis Results

In total we modeled 105 communication attacks and 84 attacks on components, and also the combination of these attacks (resulting in 8820 attack scenarios). Each attack scenario takes approximately twenty seconds to be verified by model checking, thus the total verification time for all attack scenarios (attacks on communication and components) is around one hour. Verification of each combined attack scenario takes around thirty seconds to complete, and the total verification time for all possible combinations is 72 hours. Totally, out of all above possible attack scenarios 29 cases successfully violate the system security.

References

[1] F. Moradi, S. A. Asadollah, A. Sedaghatbaf, A. Čaušević, M. Sirjani, and C. Talcott, "An actor-based approach for security analysis of cyber-physical systems," in *International Conference on Formal Methods for Industrial Critical Systems*, pp. 130–147, Springer, 2020.
[2] M. Sirjani and E. Khamespanah, "On time actors," in *Theory and Practice of Formal Methods*, pp. 373–392, Springer, 2016.
[3] "Afra: an integrated environment for modeling and verifying rebeca family designs." https://rebeca-lang.org/alltools/Afra, 2019.
[4] A. Shostack, *Threat modeling: Designing for security*. Wiley, 2014.
[5] A. P. Mathur and N. O. Tippenhauer, "Swat: a water treatment testbed for research and training on ics security," in *Cyber-physical Systems for Smart Water Networks (CySWater)*, pp. 31–36, IEEE, 2016.
[6] "Rebeca homepage." http://rebeca-lang.org/allprojects/CRYSTAL, 2020.

All DPAC publications can be found here: **http://www.es.mdh.se/dpac/**

About MDH

MDH is one of Sweden's largest HEIs, with 16 000 students reading courses and programmes in Business, Health, Engineering and Education. At MDH, research is conducted within all areas of education to address the challenges of society, and of this the research in future energy and embedded systems is internationally outstanding. MDH's close cooperation with the private and public sectors enables us to help people feel better and the earth to last longer. MDH is located on both sides of Lake Mälaren, with campuses in Eskilstuna and Västerås.

MÄLARDALEN UNIVERSITY
SWEDEN

About KKS

The Knowledge Foundation funds research and competence development at Sweden's university colleges and new universities with the purpose of strengthening Sweden's competitiveness. We provide funding when activities are conducted in collaboration between academic staff and business sector partners. The aim is to build internationally competitive, integrated research and education environments. Our mission is to strengthen Sweden's competitiveness, and we know that collaborative projects between academia and industry create great benefits for both parties. The Foundation was established in 1994 with a founding capital of 3.6 billion SEK, and has now invested some 9.3 billion SEK in over 2 500 projects.

Knowledge Foundation

Contact

DPAC

Profile leader: Kristina Lundqvist
kristina.lundqvist@mdh.se
+46(0)73 960 74 40
www.es.mdh.se/dpac

MÄLARDALEN UNIVERSITY
SWEDEN

Address: P.O. Box 883, SE-721 23 Västerås. Sweden
Address: P.O. Box 325, SE-631 05 Eskilstuna. Sweden
E-mail: info@mdh.se Web: www.mdh.se