Enabling Easier Programming of Machine Learning Algorithms on Robots with oneAPI Toolkits

Denisa Constantinescu, Angeles Navarro, Rafael Asenjo
Computer Architecture Dpt., University of Málaga

Juan-Antonio Fernández-Madrigal, Ana Cruz-Martín
System Engineering and Automation Dpt., University of Málaga

***Abstract*—This work shows that it is feasible to solve large-scale decision-making problems for robot navigation in real-time onboard low-power heterogeneous CPU+iGPU platforms. We can achieve both performance and productivity by carefully selecting the scheduling strategy and programming model. In particular, we remark that the oneAPI programming model creates new opportunities to improve productivity, performance, and efficiency in low-power systems. Our experimental results show that the implementations based on the oneAPI programming model are up to 5× easier to program than those based on OpenCL while incurring only 3 to 8% overhead for low-power systems.**

■ **WHEN** you think of oneAPI use cases that leverage the performance of heterogeneous computing, you might first envision powerful workstations and multi-megawatt supercomputers loaded with big CPUs, GPUs, and FPGAs training complex neural networks. However, our team is working on the opposite end of the spectre. This work explores solving problems in systems that use compute-intensive reinforcement learning (RL) algorithms optimized for battery-powered heterogeneous computing devices.

We have focused for many years on productively exploiting heterogeneous chips leveraging Intel® Threading Building Blocks (TBB) as the orchestrating framework and developing heterogeneous scheduling strategies [7], [8]. So, the announcement of oneAPI was immediately received in our group as an enticing opportunity to raise the level of abstraction in our implementations of heterogeneous schedulers. We see oneAPI as a solid endorsement to SYCL and modern C++ as the basis for the 'homogeneous programming of heterogeneous platforms' idea.

Heterogeneous computing with multiple types of processors can benefit large-scale and supercomputers to embedded systems. However, with smaller systems (as mobile robots), compute-intensive, automated decision-making algorithms need to be efficient, and schedulers need to be aware of energy consumption as they assign tasks to different processor architectures to optimize throughput. These aspects require new approaches to optimizing solutions for low-power systems, which we explore in this work.

Many automated planning and decision-making algorithms rely on Markov Decision Processes (MDPs) [1] and Partially Observable MDPs (POMDPs) [2]. MDPs and POMDPs describe how an intelligent agent with a defined goal learns to make better decisions by doing, even without knowing the map of its environment. POMDP and MDP agents can cope with uncertainty, such as not knowing what lies ahead or whether their actions will be beneficial. The literature shows that solving real-world problems for this kind of agent is not considered for low-power platforms [3] and computing an optimal solution for medium to large-

sized problems is intractable [4]. But the medium-large scale problems include most of the practical applications in autonomous robots, deep-space navigation, search and rescue, inspection and repair, toxic-waste cleanup, and much more.

Consequently, we focus our efforts on optimizing decision-making and planning algorithms for mobile robots (illustrated as the red boxes in **Figure 1**). This niche would highly benefit from a solution for runtime and energy-efficient implementation.

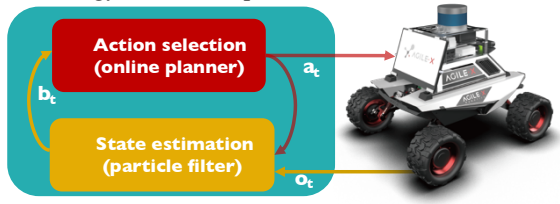

Figure 1. Decision making under uncertainty to plan for the navigation of a mobile robot modelled as a POMDP agent.

The primary source of uncertainty in a POMDP is modeled by maintaining an information state called belief, $b \in B$, representing a probability distribution over the state space S. In practice, a particle filter (the yellow block in **Figure 1**) is used to implement a recursive Bayesian filter to estimate the underlying state s_t of a dynamical system like this: $b_t = b(s_t) \propto p(o_t|s_t) \int p(s_t|s_{t-1}, a_{t-1}) b(s_{t-1}) ds_{t-1}$ [5]. Here, o_t is a sensor measurement and a_t is the action or control command; $p(o_t|s_t)$ models the observations (from sensors) while $p(s_t|s_{t-1}, a_{t-1})$ describes the system's dynamics (e.g., state transition due to actuators).

At any time t, an MDP agent is in one of the possible states, s_t, which is known, while a POMDP agent only knows its belief state estimate b_t. Given either the current state or the belief state, the agent must decide for and execute the action a_t, given a finite set of possible actions. The chosen action is known as the *policy*. Taking action a_t results in an immediate reward $r_t = r(s_t, a_t) \in \mathbb{R}$, and in a transition to state s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$. After each transition, the agent observes o_t, and the process repeats until the agent reaches its goal.

We aim to enable easy and feasible ways to implement decision making on low-power mobile platforms and focus on online planning under uncertainty for practical applications, such as autonomous driving and service robotics, that must run on SoC mobile platforms. These applications often have real-time execution constraints and run on battery-powered platforms.

The main challenge is to keep the runtime and energy performance in check while allowing the users (programmers) to code solvers for decision-making problems. Our proposed solution uses low-power aware heterogeneous computing strategies, sparse data structures to fit real-world size decision-making problems on SoCs (System on Chip) with scarce memory and computing resources, and oneAPI with DPC++ programming [6].

DEVELOPING THE SOLUTION—NAVIGATING NEW SPACES FOR MDP AGENTS

Our team has created new methods, memory-efficient data structures, and low-power aware heterogeneous computing schedulers [3], [7] to enable an intelligent agent to act autonomously in environments where the effects of its actions are not deterministic (**Figure 2**). For example, a rover taking samples from the surface of Mars may not know if a sample is worth taking or if the direction of travel will lead to worthy specimens. Or, a drone looking for survivors trapped after a natural disaster may not know if it is following a path that will eventually lead to a person.

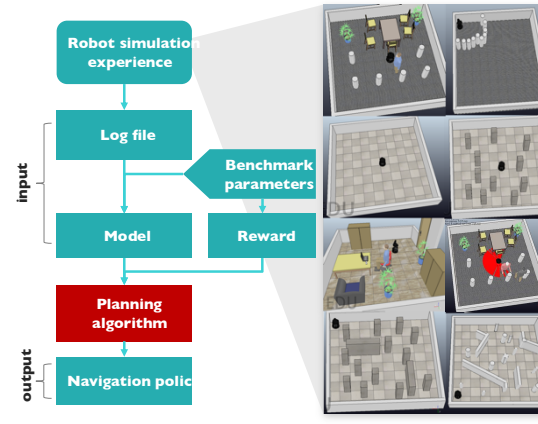

Figure 2. Robots must learn to navigate in uncertain environments, such as those above, using RL methods. We illustrate the configuration of different navigation scenarios in V-REP simulator (the black object is the robot) and how we use the simulated experience to plan navigation policies in new scenarios.

We start our journey with the Value Iteration (VI) algorithm, commonly used in MDPs and a core kernel in many RL methods, optimizing its data structures for memory use and access. Then, we explore ways to improve the performance of this planning and decision-making algorithm. In the flow diagram from **Figure 2**, the VI algorithm is represented by the red block (planning algorithm).

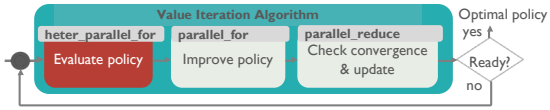

Figure 3. Planning flow with a VI algorithm.

We explore different parallel implementation strategies on low-power SoCs to improve the VI runtime and energy use. We have initially implemented multicore parallelism, using OpenMP and TBB [8], and then included GPU accelerators programmed with OpenCL. Finally, we have added heterogeneous scheduling to balance and optimize the computing resource utilization to minimize runtime and energy consumption on the most resource-consuming kernel of VI (Evaluate policy, in red), as shown in **Figure 3**.
We approach the heterogeneous scheduler code development using three different programming models: OpenCL (OCL), oneAPI with SYCL-style buffers (BUFF) written in DPC++, and oneAPI with unified shared memory (USM) written in DPC++.

EXPERIMENTAL SETUP

For the implementation, testing, and evaluation, we use Intel® DevCloud and two local mobile CPU+iGPU SoCs:

- Kaby Lake - A 1.6 GHz Intel® quad-core CPU i5-8250U, featuring a UHD 620 integrated GPU @300 MHz, 8GB of DDR4 and a TDP of 10 to 15 Watts. OS: Ubuntu 18.04.
- Tiger Lake - A 2.8 GHz Intel® quad-core CPU i7-1165G7, featuring an Iris Xe integrated GPU @1.3 GHz, 16GB of LPDDR4x and a TDP of 12 to 28 Watts. OS: Ubuntu 20.04.

Our parallel implementations use OpenCL, TBB, the oneAPI programming model, the Intel® oneAPI Base Toolkit, and Intel® oneDPL. We have installed the oneAPI DPC++ 2021.1 Compiler and the corresponding Intel® NEO Graphics Drivers on each platform. We measure energy use with PCM library [9].

We have chosen Intel low-to-medium-power processors as testbeds in our experiments because they are energy-efficient and powerful enough to run at least some AI benchmarks onboard a mobile robot. Besides, the quality and ease of use of the Intel profiling, debugging, and supporting tools now included in the Intel® oneAPI toolkits—Intel® VTune™ Profiler, Intel® Advisor and its Flow Graph Analyzer feature, Intel® Inspector—add to the "productivity" factor that we consider key to democratizing parallel and heterogeneous programming.

EVALUATING THREE PROGRAMMING MODELS

The oneAPI programming model creates new opportunities to improve performance and efficiency in low-power systems. We implement three heterogeneous schedulers for orchestrating CPU+GPU execution and evaluate them for low-power use cases.

We use the Cyclomatic complexity (CC) and Programming Effort (PE) metrics to measure how easy (or difficult) it is to program a code [10] and show the results in **Figure 4**. Higher values for CC and PE mean it is more complicated for a programmer to code the algorithm.

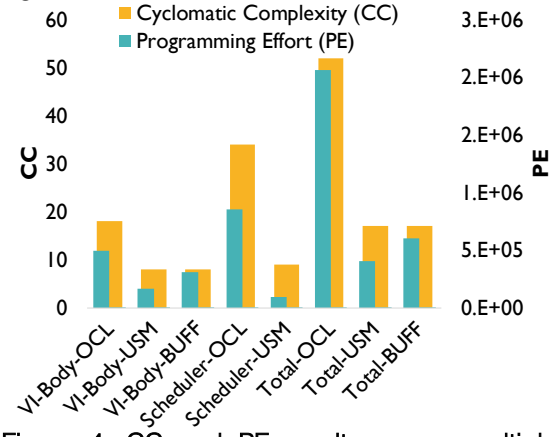

Figure 4. CC and PE results across multiple implementations of heterogeneous schedulers using OpenCL (OCL) and oneAPI (BUFF, USM).

We have found that because DPC++ code is more compact and efficient to program than OpenCL, it is as much as five times easier to program than OpenCL. With a careful scheduling strategy, it only adds three to eight per cent overhead.

Figure 5 shows results with MDP problems using three heterogeneous schedulers for CPU+GPU execution: HO uses a static partitioning strategy, HD a dynamic partitioning strategy, and HL an adaptive strategy. Each scheduler has an OpenCL (OCL) and oneAPI (ONE) version. TBB and OCL are CPU-only and GPU-only implementations, respectively, and the rest are CPU+GPU optimizations.

To configure the best CPU+iGPU workload partition for HO-OCL, we apply brute force offline exploration and tune HO with problem-specific know-how and optimizations. As a result, HO-OCL has no overhead, and the performance improvement is highest, but more painful and time-consuming to get it right. However, applications using the oneAPI Implementation of HL are extremely easy to code in comparison. We pay for productivity with some performance loss compared to HO-OCL due to the abstraction overhead. HD scheduler uses a strategy in

between HO and HL in terms of efficiency and ease of programming.

Figure 5. Speedup and energy improvement results across multiple implementations of heterogeneous schedulers using OpenCL (OCL) and oneAPI (ONE) on the Kaby Lake platform.

From the three scheduling strategies evaluated, static scheduling (HO) performs best in performance and energy efficiency, but it requires exhaustive offline searching. Adaptive scheduling provides good results with no previous training (HL), when using the USM approach to code the kernels and scheduler for large problem sizes.

ENHANCING ONLINE PLANNING FOR POMDP

Next, we apply these lessons learned from optimizing MDP planning with VI [12] to more complex decision-making procedures for POMDP agents.

We enhance a state-of-the-art POMDP online planner, DESPOT-α [11], by adding an efficient experience memory based on Bloom filters to it. This data structure is used to recall policies from experience with similarity search. In our preliminary evaluation, we compare the planning time of the baseline planner (baseline_p), DESPOT-α, and our proposal (recall_p).

Figure 6 summarizes our preliminary results for two known benchmarks in the literature of robot navigation with POMDPs, Tag and RockSample [2]. The recall_p (yellow lines) is sequential and brings little to no improvement in the planning time compared to baseline_p (blue lines). Both use the maximum time available to search for the navigation action (one second) and obtain indistinguishably "good" policies in most experiments.

Then we compare the results to other versions of the recall_p that implement the similarity search method with oneDPL. The recall_p_cpu implementation offloads the kernel to the multicore (green lines) and reacall_p_gpu to the integrated GPU (red lines). For the multicore implementation with oneDPL, we see a 2.5 to 5 × reduction in the planning time, enabling real-time performance for the evaluated benchmarks.

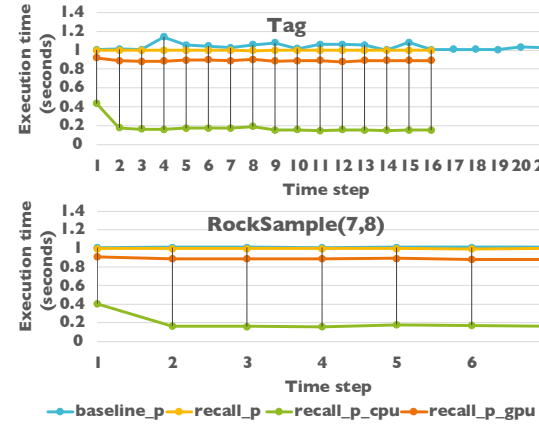

Figure 6. POMDP planning time evaluation for baseline_p, recall_p, recall_p_cpu, and recall_p_gpu on Tiger Lake platform. On the X-axis, we use as a timescale the planning timestep. Y-axis shows the planning time.

By carefully setting the experience memory parameters, recall_p may converge in fewer time steps and produce a superior policy. For example, we have the Tag benchmark evaluation in **Figure 6**, where the robot tags its target in 16 moves (time steps or actions) when using our experience memory data structure. In comparison, baseline_p requires 21 moves, given the same scenario and initial conditions.

SUMMARY OF RESULTS

During the first development phase, we have evaluated three different programming models, including oneAPI using the DPC++ programming language for planning sequences of actions for mobile robot navigation. When the scheduling strategy is carefully selected, we have found DPC++ to be five times easier to program while incurring only three to eight per cent of overhead.

In the second part, we take the lessons learned from optimizing Value Iteration for low-power execution and apply them to POMDPs—a more complex autonomous decision-making framework that accounts for all sources of uncertainty in the agent interaction with the environment. We propose a new method for online planning under uncertainty for POMDPs, Recall-Planner, that outperforms the state-of-the-art online planners for a set of benchmarks.

A novelty of our solution is that it allows us to plan for large problems on low-power SoCs, and we actively seek to achieve energy efficiency and not just make code run fast. Also, we have created a set of robot

navigation benchmarks for testing decision-making for basic navigation to a goal. The source code is available on bitbucket [13].

CONCLUSION

Memory and power are scarce for low-power embedded systems and mobile robotics computing. Despite this, applications often need close to real-time performance while balancing power use. In this work, we have presented some solutions for decision making in robotics using SoCs in low-power systems. We conclude that implementations based on oneAPI are more readable, shorter, and easier to debug. We think that a process or method that is compact, easy to understand and reproduce is far more valuable than a method that works slightly better at the cost of added complexity. In the short term, we hope this work will benefit developers who care about runtime performance and energy efficiency, as our ultimate goal is to make it feasible to develop intelligent agents and autonomous decision-making applications on mobile robots.

FUTURE DIRECTIONS

To date, we have covered the MDP category of decision-making problems and are working on POMDPs. Once we achieve this milestone, we will implement benchmarks on an AgileX Scout mobile robot. The resulting software will be made public on bitbucket.

ACKNOWLEDGMENT

This work was supported in part by grants from TIN2016-80920-R, PID2019-105396RB-I00, UMA18-FEDERJA-108, UMA18-FEDERJA-113, and Intel.

■ REFERENCES

1. Bellman R., "A Markovian Decision Process", Journal of Mathematics and Mechanics. vol. 6, no. 5, pp. 679–684, 1957. (journal)
2. Pineau J., Gordon G., Thrun S., "Point-based value iteration: An anytime algorithm for POMDPs", IJCAI, pp. 1025–1032, Aug. 2003. (conference proceedings)
3. Constantinescu D.-A., Navarro A. et al., "Performance evaluation of decision making under uncertainty for low power heterogeneous platforms." J. Parallel Distrib. Comput., vol. 137, pp. 119-133, 2020, doi:10.1016/j.jpdc.2019.11.009. (journal)
4. Nicholas R., Gordon G., Thrun S., "Finding approximate POMDP solutions through belief compression." J. Artif. Intell., vol. 23, pp. 1-40, 2005. (journal)
5. Schön T. B. "Estimation of nonlinear dynamic systems: Theory and applications", 2006. (Thesis or dissertation)
6. Reinders J., et al. "Data parallel C++: mastering DPC++ for programming of heterogeneous systems using C++ and SYCL", Springer Nature, 2021. (book)
7. Corbera F., Rodríguez A., Asenjo R., et al., "Reducing overheads of dynamic scheduling on heterogeneous chips", arXiv preprint arXiv:1501.03336, 2015. (PrePrint)
8. Voss M., Asenjo R., Reinders, J., "Pro TBB: C++ parallel programming with threading building blocks", New York: Apress., 2019. (book)
9. Processor Counter Monitor – PCM (Github). [Online]. Available: https://github.com/opcm/pcm (URL)
10. Dios A. J., Asenjo R., Navarro A., Corbera F., Zapata E. L, "High-level template for the task-based parallel wavefront pattern", 18th International Conference on High Performance Computing, IEEE, pp. 1-10, 2011. (conference proceedings)
11. Garg N.P., Hsu D., Lee W.S., "DESPOT-α: Online POMDP planning with large state and observation spaces", Robot.: Sci. Syst., 2019. (conference proceedings)
12. Constantinescu D.-A., Navarro A., Corbera F., et al., "Efficiency and productivity for decision making on low-power heterogeneous CPU+GPU SoCs." J. Supercomput., vol. 77, no. 1, pp. 44-65, 2021, doi:10.1007/s11227-020-03257-3. (journal)
13. Sourcecode (Bitbucket). [Online]. Available: https://bitbucket.org/corbera/vi-mdp/src/oneAPI/ (URL)

Authors Biographies

Denisa Constantinescu is a PhD candidate in Mechatronics at the University of Malaga and an oneAPI Innovator. She has been PI of two research projects and has collaborated on five other projects. Ms. Constaninescu has co-authored 2 book chapters, and 2 journal papers and 12 conference communications. She received the "2021 SCIE-ZONTA Award in Informatics" from The Spanish Computing Scientific Society (SCIE) and the "Intel Innovator Award" for the project "Efficiency and Productivity for Decision-making on Mobile SoCs" in 2020. Her research interests include heterogeneous programming models, optimization, decision making, and mobile robotics. Contact her at dencon@uma.es.

Rafael Asenjo is Professor of Computer Architecture at the University of Malaga. He obtained a PhD in Telecommunication Engineering in 1997. His research interests include programming models, parallel programming, heterogeneous computing, parallelization of irregular codes and energy consumption. He has participated in 19 research projects and two research contracts, published in 33 international journals indexed in the JCR, 11 contributions to IEEE and ACM "Core A" conferences, 4 Keynotes, 9 book chapters and 40 international conferences. He served as General Chair for ACM PPoPP'16 and as an Organization Committee member and a Program Committee member for several HPC related conferences (PPoPP, SC, PACT, IPDPS, HPCA, EuroPar, and SBAC-PAD). Along with Michael Voss and James Reinders, he co-authored the latest book on Threading Building Blocks (Pro TBB). He is a oneAPI Innovator, SYCL Advisory Panel member and ACM member. Contact him at asenjo@uma.es.

Angeles Navarro is full professor at the Department of Computer Architecture of the University of Malaga, Spain, since 2019. She received a PhD in Computer Science from the University of Malaga in 2000. She has been PI of several regional projects and has collaborated with national and European projects. Dr. Navarro has co-authored more than 90 papers, and 2 patents. She has served as a program committee member for several IEEE/ACM High-Performance Computing related conferences, as PPoPP, SC, ICS, PACT, IPDPS, ICPP, ISPA. She is the co-leader of the Parallel Programming Models and Compilers group at the University of Malaga.

Her research interests are in programming models for heterogeneous systems, analytical modelling, compiler, and runtime optimizations. Contact her at angeles@ac.uma.es.

Juan-Antonio Fernández-Madrigal is full professor at the System Engineering and Automation Dpt. of the University of Málaga. He received a PhD in Computer Science in 2000. He has worked on local, regional, national and UE robotics research projects since 1996, has 40 journal publications, 3 monographies, more than 70 conference communications and 6 patents. He has an H-index of 31 according to Google Scholar and has supervised a number of PhD theses on mobile robotics and cognitive robotics. His interests include mobile robotics, decision making, bayesian inference and cognitive robotics. Contact him at jafernandez@uma.es.

Ana Cruz-Martín is an associate professsor with the System Engineering and Automation Dpt. of the University of Málaga. She received the PhD in Computer Science in 2004. She has worked on local, regional, national and UE robotics research projects, which have led to several journal and conferences publications. Her research lines mainly involve educational robotics and machine learning techniques applied to mobile robotics. Contact her at acm@uma.es

Upcoming Event: Chapter Open House