Planning in the Browser

Nicolas Tran, Patrick Speicher, Robert Künnemann, Michael Backes
CISPA Helmholtz Center for Information Security, Saarland Informatics Campus, Saarbrücken, Germany
{patrick.speicher,robert.kuennemann,backes}@cispa.saarland

Álvaro Torralba
Aalborg University, Aalborg, Denmark
alto@cs.aau.dk

Jörg Hoffmann
Saarland University, Saarland Informatics Campus, Germany
hoffmann@cs.uni-saarland.de

Abstract

Traditionally, planning tools are designed as applications that users need to install, taking care of the specificities of their operating system, packages, etc. This often places a significant burden on users, especially due to the academic nature of the software. To remedy this situation, here we propose for planning to jump onto the growing trend of light-weight software use: We demonstrate the possibility to run planning tools directly in the browser. We used the emscripten framework to port Fast Downward into WebAssembly (WASM) code, which allows for in-browser execution at near-native speeds. This allows for in-browser PDDL editing and planner execution without the need for server-side resource management. Moreover, it opens up a host of new possibilities, ranging from the interactive presentation of planning-based analysis to new applications exploiting the availability of planning in browsers to perform user-centric analyses.

Introduction

'Move fast and break things often' — Facebook's developer motto until 2014 — summarizes the mentality of modern web development: web application development is an agile process with frequent specification changes. At the same time, there is a shift to push computation to the client side, i.e., the browser, motivated by both economic and operational reasons. User-side computation is scalable by design, averts security and availability issues that come with the allotment of server-side resources and saves cost (by externalizing them to the user). But there are also operational reasons: web applications can be 'closer' to the user, their personal habits and preferences. We can solve privacy-related challenges by simply not running the code in the cloud, while retaining the mobility of cloud-based code delivery.

Planning technology is a perfect match for 'move fast'-mentality: solvers are drop-in solutions; they are not necessarily performance-optimal, but fast. They can quickly adapt to frequent problem changes. Unfortunately, no competitive planner is available in the browser, where JavaScript is the lingua franca for active content. Moreover, traditional planners are typically heavily optimized C or C++ code. Even if rewritten in JavaScript, their performance would suffer, e.g., due to JavaScript's dynamic type checks, its garbage collector or lack of ahead-of-time compiler optimization.

WebAssembly (WASM), which has become a W3C recommendation in 2019 (W3C), provides a solution to both problems. It defines a portable binary-code format for executable programs and was designed to run at near-native speed. With emscripten[1], there is a compiler from C to WASM that helps porting existing planners. Considering the superior performance of WASM over JavaScript, a generic planning algorithm in WASM may end up faster than a specialized algorithm implemented in JavaScript.

We used emscripten to port the Fast Downward Planning System (Helmert 2006), including Speicher et al.'s extensions to Stackelberg planning (2018a). We will skim over some challenges encountered in porting before, but focus on existing and potential applications for in-browser planner. Finally, we evaluate the performance in WASM versus a native execution and show that the overhead is affordable (50–100%).

Porting Challenges

Emscripten is a source-to-source compiler that runs as a back end to the LLVM compiler and produces a subset of JavaScript known as WASM. The challenges were threefold.

- Fast Downward consists of two modules, the translator which reads an input PDDL file and transforms it to an intermediate FDR representation, and the search component which solves the task. Porting the translator part, which is written in Python, to WASM is not straightforward. In the demo, we substitute the translator by a domain-specific encoding of our domain in FDR.
- Portability: Fast Downward contains OS-specific code for memory usage statistics and input handling. As the browser's execution environment does not provide them, they had to be replaced or removed.
- Transmitting input/output: Fast Downward employs C++-style input/output streams which are incompatible with the browser's execution environment. By default, emscripten writes the output as a file within a virtual file system. We modified Fast Downward to obtain the input

[1] https://emscripten.org/

file as a command line string, and added glue code that fetches the output file and displays its content.

Demonstration

Our demonstration consists of two parts, a domain-specific application that runs the planner in the browser, and a simple interface that allows the user to run the planner on any instance provided by the user.

Analyses with customizable parameters Stackelberg planning (Speicher et al. 2018a) computes optimal solutions in a game-like planning task between two players. Speicher et al. used this technique to compare various security technologies (e.g., IPsec, DNSSEC, SMTP-over-TLS, START-TLS, strong certificate validation in SMTP) to improve confidentiality in the email system. Roughly speaking, the first player deploys one or more of these technologies on a set of hosts in the current infrastructure. The second player represents a surveillance attacker and spies on as many connections as possible. The outcome is a front of plans that are Pareto optimal w.r.t. the cost of deploying the countermeasures, and the users affected by the attack.

A major obstacle in communicating these results is the agreement on cost estimates, which vary wildly between countries and companies operating the infrastructure. Yet a result is only meaningful to a user if they consider these parameters accurate. We provide a 'parametric' version of our results as a web application that allows for customizing these parameters.[2] Users can now refine our estimates according to their expertise and, possibly, insider knowledge, without having to share it with other partners. They can also choose among several attacker models and groups of users to be protected, providing highly individualized results, if desired.

Planning in the browser We offer a simple interface, where users can provide a task in FDR representation and run Fast Downward to solve it. This allows to evaluate the performance of the in-browser planner before implementing a domain-specific user interface. The user can choose between an optimal (A*with $h^{\text{LM-cut}}$ (Helmert and Domshlak 2009)) and an agile (LAMA (Richter, Westphal, and Helmert 2011)) configuration.

Evaluation

We measured the performance impact of running Fast Downward inside the browser environment. We run A^* search with the LM-cut (Helmert and Domshlak 2009) heuristic on three standard IPC domains, namely Logistics98, Rovers, and Satellite. We ran the experiments on an Intel Core i7-7820HQ, 2.90 GHz with Chrome v85 and Firefox v80.0. For each task, we set a 5-minute time limit and a 4-GB memory limit. All tasks solved with the native implementation also finished in the browsers, and the memory limit was never hit.

[2] `https://project.cispa.io/fd-in-browser/`

	logistics98	rovers	satellite
coverage (N/C/F)	6/6/6 of 35	7/7/7 of 40	7/7/7 of 36
overh. C	83.2 ± 13.4	58.0 ± 6.38	72.2 ± 15.3
overh. F	93.1 ± 11.2	72.2 ± 5.11	87.0 ± 20.6

Table 1: Performance evaluation for Firefox (F) and Chrome (C). We ignored all tasks that take less than 0.3s on all platforms. On the remaining 3 tasks per domain, we computed overhead as the arithmetic mean of the native runtime (N) divided by the runtime in the respective browser minus 1, here presented in %.

Across all the tasks finishing in less than 5 minutes, but more than 0.3s, there is a performance loss of around 50% to 100% (see Table 1). This number is in line to the previous findings reporting an average slowdown of 45% (Firefox) to 55% (Chrome), with peak slowdowns of $2.08\times$ (Firefox) and $2.5\times$ (Chrome) (Jangda et al. 2019). Our results are slightly better, possibly because Fast Downward does not need to perform slow IO operations after reading in the input file.

Discussion and Potential Applications

This demo showcases how planners can be run in the browser to improve scalability and privacy compared to computation in the cloud, and avoid having to compile and install the planner, compared to native execution. Thus we foresee many potential applications.

Ready-to-go planning IDE An in-browser planner provides the possibility to edit and solve PDDL input directly in the browser. This facilitates the quick and easy use of planning technology. Existing in-browser editors like `https://editor.planning.domains` and WEB PLANNER (Magnaguagno et al. 2017) provide this functionality, but rely on a backend server for problem solving. Besides the aforementioned issues in terms of scalability, availability and privacy of the input data, the network latency adds to the perceived responsiveness, which is important in small domains, as used in teaching. Integrating our WASM module would immediately mitigate all these issues: results would display as soon as they are computed and users could exploit the full power of their machine.

Data privacy plugins Data privacy research suggests the sanitisation of data sent to social network to minimize the risk of exposure while retaining the utility of the information. Examples are the perturbation of location information (Luceri et al. 2020), the generalisation of hashtags in Tweets (Zhang et al. 2018) or the redaction of images (Orekondy, Fritz, and Schiele 2018)). Planning algorithms can be used to optimize these steps if the sanitisation can be described in terms of discrete operations (Kulynych et al. 2018). Sanitisation would typically be implemented in a browser plug-ins, as these can interact with the social network website while also being under the user's control.

References

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 162–169. AAAI Press.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Jangda, A.; Powers, B.; Berger, E. D.; and Guha, A. 2019. Not so fast: Analyzing the performance of webassembly vs. native code. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 107–120. Renton, WA: USENIX Association.

Kulynych, B.; Hayes, J.; Samarin, N.; and Troncoso, C. 2018. Evading classifiers in discrete domains with provable optimality guarantees. *arXiv preprint arXiv:1810.10939*.

Luceri, L.; Andreoletti, D.; Tornatore, M.; Braun, T.; and Giordano, S. 2020. Measurement and control of geolocation privacy on twitter. *Online social networks and media* 17:100078.

Magnaguagno, M. C.; Pereira, R. F.; Móre, M. D.; and Meneguzzi, F. 2017. Web planner: A tool to develop classical planning domains and visualize heuristic state-space search. In *Proceedings of the Workshop on User Interfaces and Scheduling and Planning, UISP*, 32–38.

Orekondy, T.; Fritz, M.; and Schiele, B. 2018. Connecting pixels to privacy and utility: Automatic redaction of private information in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8466–8475.

Richter, S.; Westphal, M.; and Helmert, M. 2011. LAMA 2008 and 2011 (planner abstract). In *IPC 2011 planner abstracts*, 50–54.

Speicher, P.; Steinmetz, M.; Backes, M.; Hoffmann, J.; and Künnemann, R. 2018a. Stackelberg planning: Towards effective leader-follower state space search. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, 6286–6293. AAAI Press.

Speicher, P.; Steinmetz, M.; Künnemann, R.; Simeonovski, M.; Pellegrino, G.; Hoffmann, J.; and Backes, M. 2018b. Formally reasoning about the cost and efficacy of securing the email infrastructure. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P'18)*, 77–91.

W3C. 2019. Webassembly core specification.

Zhang, Y.; Humbert, M.; Rahman, T.; Li, C.-T.; Pang, J.; and Backes, M. 2018. Tagvisor: A privacy advisor for sharing hashtags. In *Proceedings of the 2018 World Wide Web Conference*, 287–296.