Design Space Exploration via Answer Set Programming Modulo Theories

Philipp Wanko

Institute of Computer Science, August-Bebel-Str. 89, 14482 Potsdam, Germany

Abstract. The design of embedded systems, that are ubiquitously used in mobile devices and cars, is becoming continuously more complex such that efficient system-level design methods are becoming crucial. My research aims at developing systems that help the designer express the complex design problem in a declarative way and explore the design space to obtain diver sets of solutions with desirable properties. To that end, we employ knowledge representation and reasoning capabilities of ASP in combination with background theories. As a result, for the first time, we proposed a sophisticated methodology that allows for the direct integration of multi-objective optimization of non-linear objectives into ASP. This includes unique results of diverse sub-problems covered in several publications which I will present in this work.

1 Introduction

With increasing demands for functionality, performance, and energy consumption in both industrial and private environments, the development of corresponding embedded processing systems (ECSs), that are for example used in mobile devices or cars, is becoming more and more intricate. Also, desired properties are conflicting and compromises have to be found from a vast number of options to decide the most viable design alternatives. Hence, effective Design Space Exploration (DSE; [20]) is imperative to create modern embedded systems with desirable properties; it aims at finding a representative set of optimal valid solutions to a design problem helping the designer to identify the best possible options.

The overall aim of my research until now was the study of exact DSE techniques for ECSs utilizing the knowledge representation and reasoning capabilities of ASP in combination with background theories (ASP modulo Theories, ASPmT). As a result, for the first time, we proposed a sophisticated methodology that allows for the direct integration of multi-objective optimization of non-linear objectives into ASP. This includes unique results of diverse sub-problems covered in several publications.

2 ASPmT-based System Synthesis

Our first objective was the development of an ASPmT-based approach to system synthesis including allocation, binding, routing, and scheduling while adhering to
area, energy, and timing constraints. Allocation determines the hardware components that are used from a given architecture, binding assigns applications to hardware components, routing determines the paths of messages between applications through the hardware architecture, and finally, scheduling determines the exact time points when applications are executed and messages sent. Area, energy, and timing constraints refer to the minimization of the number of hardware components, energy consumption, and duration of the execution of the applications, respectively. Our results, published in [15]. We holistically encode the system synthesis problem featuring meshed network-on-chip (NoC) hardware architecture and multi-hop communication, as well as conflict-free scheduling of periodically activated applications. The schedulability analysis is executed as two background theories, specifically by a propagator handling quantifier-free integer difference logic (QF-IDL) constraints and another checking periodic overlaps. The scheduling is tightly integrated and capable of partial solution checking by using our ASPmT framework [6]; it extends the ASP system clingo with a general interface to integrate application- and theory-specific reasoning into ASP. That is, arbitrary theories can now be directly included into ASP’s modeling language through common couple variables that are part of standard ASP rules and thus, can profit from the sophisticated propagation techniques offered by clingo.

We delegate aspects of the system synthesis problem to standard ASP, namely binding and routing as well as area and static energy requirements, since they can be effectively expressed and verified in pure ASP (cf. [1]) while we use QF-IDL for timing constraints. Thus, we are alleviating the need to express all possible start times directly in ASP, which would lead to a blowup in problem size, and assigning non-linear scheduling tasks to a more suitable paradigm. QF-IDL furthermore has the advantage of being solvable in polynomial time and the consistency checking yields minimal conflict clauses which are imperative for solution space pruning.

In fact, our contribution goes well beyond DSE: The system clingo[DL] [8] instantiates the ASPmT framework of clingo with QF-IDL, thus providing a generic combination of ASP with QF-IDL, featuring a hybrid modeling language along with hybrid multi-objective optimization. Moreover, our experimental studies showed that clingo[DL] performs very well compared to other state-of-the-art hybrid solvers, including other instantiations of clingo’s ASPmT framework, using linear programming and constraint programming for expressing difference constraints, as well as comparable state-of-the-art hybrid solver based on modern SMT-solvers.

3 ASPmT-based Design Space Exploration Framework

After studying ASP-based system synthesis, we extended our approach towards a holistic DSE framework including multi-objective optimization. The results have been presented in [16].
I will now outline alternative methods for computing Pareto-optimal solutions in our DSE framework. Ideally, after the design space is explored completely, DSE returns the true Pareto set containing all optimal solutions from which the designer can choose the favored design points. However, an exhaustive search in the design space is not viable for large problem instances, which is why only an approximate Pareto set is obtainable in reasonable time. Hence, we developed three different exploration strategies calculating the Pareto set in an anytime fashion. That is, whenever DSE is aborted prematurely, an approximate Pareto set is returned; its quality improves over time until eventually the true Pareto set is found. We evaluate all alternatives by two criteria: entropy [5] specifying how regular the design points are distributed across the objective space (diversity), and $\epsilon$-dominance [23] reflecting the distance between true Pareto and approximation set (convergence).

The first strategy is based on asprin [4], a general framework for optimization in ASP. In asprin, ASP is used to express objectives, and optimization follows a branch and bound scheme. That is, each new solution is required to be better than the previous one, until no better one is found. We extend this approach by allowing for propagators to implement objectives that depend on other background theories. This is needed, e.g., for optimizing latency since the starting times of tasks and communications are only known to the QF-IDL theory. Regardless of the origin of the preference, it is used to derive special atoms whenever the current solution improves a previous one. These atoms are in turn aggregated by an ASP rule to express Pareto preference. Consecutively, constraints are added to the problem definition ensuring an improvement of a solution, until a true Pareto-optimal design point is found. This approach first tries to prove Pareto optimality by improving convergence continuously. To enumerate Pareto optimal solutions, a constraint is added whenever an optimal solution is found, enforcing that newly found solutions are incomparable to the optimum. After that, the branch and bound process resumes until no more solutions are found.

The second approach follows the same optimization scheme but evaluates the objectives as well as the dominance checks within the background theory. The third approach explores the design space in a breadth-first fashion, i.e., if a new design point is found, it is not required to strictly dominate previously found solutions. All currently known best solutions are kept in an archive and dominated solutions are removed whenever a better solution is found. Similar to the second approach, evaluation of design points and enforcement of quality constraints are exclusively handled in the background theory.

Our experiments show that the third strategy finds the most diverse solutions [16]. Considering convergence, the first and third approach perform nearly the same. However, the former only returns one best known solution after the timeout while the latter offers an approximate Pareto set with multiple solutions. The experiments show that we are able to handle large problem instances with up to 170 tasks mapped to a hardware platform implemented as a meshed NoC as communication infrastructure.
Compared to existing meta-heuristic \cite{3}, hybrid \cite{9,22,12} and exact \cite{10} DSE techniques, our approach has several advantages. First, even for highly constrained problems, found solutions are guaranteed to be feasible. Second, design points are not explored multiple times resulting in a more efficient exploration. Third, reachability can be directly expressed in ASP. That is, the encoding for routing of communication messages is formulated naturally with recursive rules. Finally, since ASPmT allows us to conduct constraint and dominance checks on partial assignments, our approach prunes larger regions earlier during search.

As the number of non-dominated design points increases during search, archiving them becomes tedious due to the growing number of dominance checks. This is even more severe when dealing with partial assignments since each undergoes a dominance check until a complete solution is obtained. We address this in \cite{14} by investigating Quad-Trees as basic data structure of solution archives. Our experiments show that they help diminishing the number of comparison operations by an order of magnitude when considering more than three objective functions. In contrast to using Quad-Trees with complete solutions only, as done in \cite{11}, updating the archive of solutions does not have to be performed for each dominance check. The reason for this is that a partial solution may dominate the archive but deteriorates with additional decisions made. Thus, it cannot displace any solutions from the archive until all decisions are made. That is, the dominance check of partial solutions only signifies the necessity of assigning the remaining decisions. This allows for skipping the expensive update operation of the archive for partial assignments and executing it only in the final step.

4 Related Publications

To warrant the generality of our techniques, we elaborated upon Curriculum-Based Course Timetabling (CB-CTT) in \cite{2}, a problem closely related to schedulability analysis. While the hard constraints are fixed, the inclusion of soft constraints and their priority can be freely configured. We were able to draw from a well of existing real-world benchmarks stemming from the CB-CTT community, for which best known schedules have already been obtained by a plethora of different technologies. Besides improving or reproducing best known bounds for over half of the available instance set, our system solved difficult combinations of soft constraints for very large instances, like the benchmarks from the University of Erlangen, for the first time. We were also able to improve solutions for instances with unknown optima by using approximate optimization schemes. In detail, instead of aggregating all soft constraints, we ordered them lexicographically. While the optima of the lexicographic optimization might not provide the globally best bound, by separating the objective functions and optimizing difficult criteria last, we improved the best known bounds for large instances.

Often, a valid schedule is already in place but circumstances change. In such a case, it is desirable to change the previous schedule as little as possible (stability), while finding a schedule fulfilling the new constraints with high quality (optimality). This problem is called Minimal Perturbation Problem for Curriculum-Based
Course Timetabling. With our ASP techniques, we were able to implement a solving scheme based on lexicographic optimization that enumerates all Pareto optimal solutions regarding stability and optimality. The idea is to first obtain the extreme Pareto optimal solutions by calculating the lexicographic optima. Then, the bounds of these solutions are used to restrict the search space, cutting off dominated areas and lexicographic optimization resumes. The full Pareto set is found once no solution is found in the restricted search space.

To present small subsets of representative solutions, we introduce a framework for computing diverse (or similar) solutions to logic programs with preferences in [21]. The first contribution is the automation of various ASP solving schemes, namely max-min optimization, guess and check, querying and preferences over preferences. Using the solving schemes as building blocks, the framework provides three kinds of techniques: enumeration, replication and approximation. Each one tries to calculate $n$ most diverse/similar optimal solutions given a logic program with preferences and a distance measure. The latter gives the pair-wise distance between two solutions, e.g., the number of varying truth values in two solutions. A set of solutions is most diverse when the minimal pair-wise distance is maximal among all possible solution sets of cardinality $n$. Our experiments show that enumeration and replication are ineffective, since exponentially many optimal solutions might have to be enumerated, and treating series of already complex optimization problem leads to an explosion in complexity. Approximation was the most successful technique, and we witnessed a trade-off between diversification quality and run time depending on whether optimization or heuristics were used to identify the next optimal solution.

To evaluate our approaches, it is imperative to have a significant amount of test cases that allow for studying performance and scalability. We therefore investigated a methodology to systematically create system specification instances that results in a versatile and easily expandable ASP-based benchmark generator [13]. The generator produces synthetic system synthesis specifications composed of applications, heterogeneous hardware platforms, and mapping options. Due to its modular structure, rules for encoding the generation of applications or hardware platforms can be adjusted independently of each other, which allows us to utilize the framework for a wide range of application and hardware platform structures. By default, it generates applications with series-parallel communication patterns that allow for modeling a plethora of characteristics with one versatile ASP encoding. Depending on the dominance of series or parallel patterns, the application allows for more or less concurrency. Utilizing ASP, the generator produces variant instances with similar characteristics (e.g., number of tasks, connectivity, concurrency) while the shapes of the generated applications differ. To check produced instances w.r.t. constraints such as minimum latency, we combined the generator with the ASPmT-based system synthesis approach.

In order to improve DSE, we currently investigate over and under estimation techniques for quality constraints and objective functions. The rational behind such estimation techniques is a more rapid evaluation phase of a found design
point. Additionally, if an estimation is guaranteed to evaluate an objective as better compared to the exact value, e.g., under estimation for minimization problems, we use this information to skip expensive calculations without the loss of accuracy of the obtained Pareto optimality [17]. This is particularly true for partial assignments that have to be evaluated many times before the complete solution is obtained. That is, if the under estimation evaluation is already dominated by a design point in the archive, the exact evaluation would be worse and thus also be dominated. Hence, the expensive exact calculation can be skipped. Only if the under estimation indicates a good solution, the exact calculation has to be performed.

5 Future Work

My current goal is developing a theoretic framework based on the Logic of Here-and-There [7,18,19] capable of capturing the combination of arbitrary theories and sophisticated language constructs. This should enable, first, a foundation for seamless combination of multiple theories, e.g., QF-IDL with ILP, and second, advanced language constructs like aggregates involving entities related to the theories, e.g., an aggregate calculating the maximum over variables used in difference constraints. Eventually, this will be realized via clingo’s theory API and function as a culmination and combination of my doctoral studies.

References

1. Andres, B., Gebser, M., Glaß, M., Haubelt, C., Reimann, F., Schaub, T.: Symbolic system synthesis using answer set programming. In: Proc. of LPNMR. vol. 8148, pp. 79–91 (2013)
2. Banbara, M., Inoue, K., Kaufmann, B., Okimoto, T., Schaub, T., Soh, T., Tamura, N., Wanko, P.: teaspoon: Solving the curriculum-based course timetabling problems with answer set programming. Annals of Operations Research (2018)
3. Blickle, T., Teich, J., Thiele, L.: System-Level Synthesis Using Evolutionary Algorithms. In: Design Automation for Embedded Systems, pp. 23–62. Springer (1998)
4. Brewka, G., Delgrande, J., Romero, J., Schaub, T.: asprin: Customizing answer set preferences without a headache. In: Proc. of AAAI. pp. 1467–1474 (2015), http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9535
5. Farhang-Mehr, A., Azarm, S.: An information-theoretic entropy metric for assessing multi-objective optimization solution set quality. Journal of Mechanical Design 125(4), 655–663 (2003)
6. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: Proc. of ICLP. pp. 2:1–2:15 (2016)
7. Heyting, A.: Die formalen Regeln der intuitionistischen Logik. In: Sitzungsberichte der Preussischen Akademie der Wissenschaften, pp. 42–56. Deutsche Akademie der Wissenschaften zu Berlin (1930), reprint in Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik, Akademie-Verlag, 1986.
8. Janhunen, T., Kaminski, R., Ostrowski, M., Schaub, T., Schellhorn, S., Wanko, P.: Clingo goes linear constraints over reals and integers. Theory and Practice of Logic Programming 17(5-6), 872–888 (2017)
9. Lukasiewycz, M., Glaß, M., Haubelt, C., Teich, J.: SAT-Decoding in Evolutionary Algorithms for Discrete Constrained Optimization Problems. In: Proc. of CEC. pp. 935–942 (2007)
10. Lukasiewycz, M., Glaß, M., Haubelt, C., Teich, J.: Efficient Symbolic Multi-Objective Design Space Exploration. In: Proc. of ASPDAC. pp. 691–696 (2008)
11. Mostaghim, S., Teich, J.: Evolutionary Multiobjective Optimization, chap. Quad-trees: A Data Structure for Storing Pareto Sets in Multiobjective Evolutionary Algorithms with Elitism, pp. 81–104. Springer (2005). [https://doi.org/10.1007/1-84628-137-7_5]
12. Neubauer, K., Haubelt, C., Glaß, M.: Supporting composition in symbolic system synthesis. In: Proc. of SAMOS. pp. 132–139 (2016). [https://doi.org/10.1109/SAMOS.2016.7818340]
13. Neubauer, K., Haubelt, C., Wanko, P., Schaub, T.: Systematic test case instance generation for the assessment of system-level design space exploration approaches. In: Proc. of MBMV (2018).
14. Neubauer, K., Haubelt, C., Wanko, P., Schaub, T.: Utilizing quad-trees for efficient design space exploration with partial assignment evaluation. In: Proc. of ASPDAC. pp. 434–439 (2018)
15. Neubauer, K., Wanko, P., Schaub, T., Haubelt, C.: Enhancing symbolic system synthesis through ASPmT with partial assignment evaluation. In: Proc. of DATE. pp. 306–309 (2017)
16. Neubauer, K., Wanko, P., Schaub, T., Haubelt, C.: Exact multi-objective design space exploration using ASPmT. In: Proc. of DATE. pp. 257–260 (2018)
17. Neubauer, K., Wanko, P., Schaub, T., Haubelt, C.: On leveraging approximations for exact system-level design space exploration. In: Proc. of CODES+ISSS (2018), to appear
18. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Dix, J., Pereira, L., Przymusinski, T. (eds.) Proceedings of the Sixth International Workshop on Non-Monotonic Extensions of Logic Programming (NMELP’96). vol. 1216, pp. 57–70 (1997)
19. Pearce, D.: Equilibrium logic. Annals of Mathematics and Artificial Intelligence 47(1-2), 3–41 (2006)
20. Pimentel, A.: Exploring exploration: A tutorial introduction to embedded systems design space exploration. IEEE Design & Test 34(1), 77–90 (2017)
21. Romero, J., Schaub, T., Wanko, P.: Computing diverse optimal stable models. In: Proc. of ICLP. pp. 3:1–3:14 (2016)
22. Schlichter, T., Lukasiewycz, M., Haubelt, C., Teich, J.: Improving system level design space exploration by incorporating sat-solvers into multi-objective evolutionary algorithms. In: Proc. of ISVLSI (2006). [https://doi.org/10.1109/ISVLSI.2006.57]
23. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE TEC 7(2), 117–132 (2003)