The Evolution of Scheduling Applications and Tools

Mark Boddy

■ *The available tools and support for building planning and scheduling systems and applications have been steadily improving for decades. At the same time, the scope, scale, and complexity of the problems to be addressed have been increasing. In this column, I discuss several different scheduling applications developed over the past 25 years and then describe the tools and techniques used in addressing these problems, showing how improved tools simplified (and in some cases enabled) the solution of problems of increasing difficulty.*

According to folk definitions of planning and scheduling commonly used in the AI community, *planning* is deciding what to do, while *scheduling* is deciding when and how to do it. Neither of these terms are fundamental categories. *Scheduling applications* are simply those that can be addressed using scheduling tools and techniques. Using a few examples drawn from personal experience spanning more than two decades, in this column I provide one perspective on how those tools and techniques have evolved, as well as the resulting effect on the scope and scale of applications that can be addressed.

Between 1993 and 1995, a group of us at Honeywell implemented a constraint-based scheduler for the Airplane Information Management System (AIMS) that was developed for the Boeing 777. The initial AIMS scheduling problem encompassed 29,000 discrete activities, subject to 97,000 complex metric constraints specified by AIMS applications developers. Generating feasible schedules was an essential requirement for operating the 777, potentially threatening a Boeing investment of almost 10 billion dollars. The scale and complexity of this problem were unprecedented, and there were very few applicable tools or standards. Input requirements were provided as text, with a semantics negotiated and maintained through frequent discussion. As this was one of the earliest schedulers based on the simple temporal problem (STP), we implemented methods for incremental updates and bounds computation, as well as integrating the STP model with a large set of discrete decision variables. The solver used a locally implemented adaptation of Ginsberg's recently published Dynamic Backtracker, a systematic search algorithm combining stack reordering (not just conflict-directed backjumping) with what would later be called clause learning, as well as several customized constraint propagators (Boddy and Goldman 1994). Notable in this development effort was the extended process of negotiation with the AIMS developers as they sought to preserve functionality, repeatedly providing sets of requirements that we demonstrated to be unsatisfi-

able, using tools implemented specifically for that purpose.

Subsequently, tooling support for building planning and scheduling systems became more prevalent. For example, ilog Solver and Scheduler provided constraint modeling and solving capabilities, including specialized implementations of global constraints such as all-different. These tools used an extension of the Prolog goal stack, rendering them of limited utility where explicit control of the search process was required. Advances in understanding the relationship between propositional reasoning and integer programming widened the set of solvers available, while improvements in tools like CPLEX and a range of constraint-satisfaction problem (CSP) solvers including but not limited to iLog tools such as OPL further reduced the amount of implementation needed for a new application. Scheduling-specific ontologies and specialized constraints, custom control over constraint propagation, and support for backjumping during search made these systems increasingly useful. By 2010, these improvements in integration and scale enabled us (now at Adventium) to implement a system modeling processing and communication demands in large networks (10,000 to 1,000,000 nodes). Newly developed tools and standards played a key part: the domain and problem instances were represented in the Architecture Analysis and Design Language (AADL), from which we extracted a set of constraints in the high-level constraint language MiniZinc, translated from there into a linear program, which solved in single-digit minutes, using stock hardware (Michalowski, Boddy, and Carpenter 2010).

Then in 2013, very nearly 20 years after the AIMS scheduler, we constructed a prototype scheduler for a modern avionics system for a large commercial jetliner. The problem was larger, more complex, and significantly more diverse than that addressed by the AIMS scheduler. Instead of a single integrated network with external sensor interfaces, there were multiple networks with gateways between them. Instead of one communication protocol (and thus one scheduling model) there were several. Instead of functions preassigned to processors, that assignment was part of the problem. Multiple, gatewayed networks with different communication protocols were a particular issue, requiring enforcement of timing guarantees across multiple asynchronous boundaries. Fortunately, the available tools were much improved as well, to the point where the application could be assembled largely by integrating existing tools. Instead of hand-rolled domain models, we used AADL. Instead of manual integration of discrete and continuous parts of the problem as had previously been required, we used satisfiability modulo theories (SMTs). Instead of hand-implemented search control, we used an off-the-shelf solver. The largest remaining implementation task was to translate from the AADL model to a formulation suitable for the SMT solver.[1] This tool remains in a prototype form, but the models and integration methods have been used for other applications.

Most recently, we have integrated a scheduling engine into design tools for integrated, heterogeneous embedded software systems. These tools have been evaluated by system developers, and are now being provided as Government Furnished Equipment for the preliminary design phase of a major aircraft development program.[2] In our view, this kind of integration is where the main challenge of the future lies. Public infrastructure is increasingly distributed and integrated, while embedded systems grow increasingly complex and interdependent. Individual components of these large systems may have very different dynamics and may be provided by vendors desiring to protect proprietary information. Addressing these problems requires integrating diverse tools, sharing limited information, with a rigorous semantic mapping among them.

Over the past 20 years, there has been significant progress in the tools available to support all aspects of defining and implementing constraint-based scheduling applications. This process is ongoing; examples of current research technologies with promise for real applications include various refinements of Monte-Carlo tree search, and the synthesis of specific algorithms for problem instances, as for example in ongoing work by Doug Smith at Kestrel. At the same time, the increasing complexity and integration of computing systems continues to provide more, larger, and more complex problems to solve. This class of applications should provide a fruitful source of new modeling and solution challenges for years to come.

Notes

1. See nari.arc.nasa.gov/sites/default/files/Boddy SPICA PhaseI FinalReport r2.pdf

2. www.adventiumlabs.com/our-work/products-services/model-based-engineering-mbe-tools

References

Boddy, M., and Goldman, R. 1994. Empirical results on scheduling and dynamic backtracking. Paper presented at the Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space, October 18–20, Pasadena, California USA.

Michalowski, M.; Boddy, M.; and Carpenter, T. 2010. Coordinated Management of Large-Scale Networks Using Constraint Satisfaction. Paper presented at the 2010 AAAI Workshop on Intelligent Security (Security and Artificial Intelligence), 12 July, Atlanta Georgia, USA.

Mark Boddy is co-owner and chief scientist at Adventium Labs, a small, for-profit research lab located in Minneapolis, MN. His areas of research include planning and scheduling, automated reasoning, and constraint-based reasoning. Over the past 30 years, most of his work has been on adapting and extending techniques from a broad variety of research areas for application to a range of problems, primarily in planning and scheduling domains. This work has drawn from and in some cases contributed to research in constraint satisfaction, heuristic search, mathematical optimization, classical planning and extensions thereto, temporal reasoning, multiagent cooperative negotiation, resource-bounded reasoning, and a number of other areas. He coauthored the papers that coined the widely used terms *anytime algorithm, performance profile,* and *conformant planning.*