# How to Make Scheduling Research Relevant

**David Montana**
BBN Technologies
10 Moulton Street, Cambridge, MA 02138
dmontana@bbn.com

## Abstract

Scheduling research has been an active field for the last forty or so years. Over the decades, many techniques have been developed for application to a variety of different problems, and we have learned a lot about how to perform scheduling. There have been some large successes in the application of automated scheduling to real-world problems. However, the reality remains that most scheduling tasks that could potentially benefit from automated scheduling are performed manually, and if not, then by an algorithm that produces highly suboptimal schedules. The question is how to focus future scheduling research so as to better address the needs of the real world and hasten the adoption of automated, optimized scheduling for a wider range of real-world problems. I will discuss some of the reasons for the gap between scheduling research and practice and some approaches to bridging this gap.

## 1 Analysis of Current Problems

Many organizations and individuals with scheduling problems would need scheduling technology to provide some or all of the following capabilities before it would make sense for them to utilize automated scheduling.

**The automated scheduler should solve the users' particular problem, not some simplified abstraction.** Each user has their own scheduling problem, with a unique set of constraints and criteria governing how the scheduling should be done. A scheduler that only solves a different problem with a different set of constraints and criteria is of no use to a user. An example of this is manufacturing. For years, researchers have been solving job-shop and flow-shop problems. However, the scheduling algorithms they devised are generally of no use to companies with actual problems in scheduling of manufacturing. The real-world problems are different, and often more complicated, than the research problems. Hence, with the exception of those larger companies that have the resources to develop custom solutions, most manufacturing is still scheduled manually.

**The automated scheduler should produce schedules that are good enough and execute fast enough.** Scheduling is computationally complex, with most scheduling problems being NP-complete. As the size of a scheduling problem grows, the time it takes to find the globally optimal solution increases very quickly. Usually, in order to find a schedule in an acceptable amount of time, a tradeoff has to be made in terms of accepting a less-than-optimal schedule. The parameters of this tradeoff are highly problem-dependent and user-dependent; some problems require schedules quickly even if they are highly suboptimal while others require nearly optimal schedules even if this takes a long time. There is a need for algorithms that not only provide better schedules faster but also that allow the user to make explicit tradeoffs between schedule quality and computation time.

**The total cost of the automated scheduling system should be low enough to justify the return on investment (ROI).** Before committing to adoption of automated scheduling for a particular problem, an organization (or individual) will do some type of cost-benefit analysis to make sure that the benefits justify the costs. While most researchers concentrate only on the scheduling algorithm, the development of an algorithm is just the beginning of the costs. The automated scheduler must be part of a full system that includes (i) displays that involve the user and (ii) interfaces to the data sources and sinks. In the case

when the organization can purchase or download a pre-existing software package, there are still the purchase price and the costs of customizing the system for their particular situation. After these initial costs, there are also the costs of training the operators as well as administering and maintaining the system. With most current technology, unless the organization has a lot of money riding on the quality of the schedules, it cannot justify the investment in automated scheduling. The total costs of an automated scheduling solution must be greatly reduced before smaller organizations and individuals will use one.

**There should be some easy way to provide confidence that an automated scheduler will solve the users' problem.** Most organizations are concerned about the risk that their investment in an automated scheduler will not pay off because the automated scheduler will not do its job correctly. Even when the expected ROI is sufficient to justify the project, many organizations will avoid adopting an automated scheduler because of this perceived risk. To a certain extent, success breeds confidence; being able to demonstrate an automated scheduler working on a similar problem allays fears. However, it is also very helpful to have a rapid prototyping capability, i.e. some way to quickly create a demonstration that is sufficiently similar to the real problem to inspire confidence (as well as providing a good way to learn about the problem).

**The automated scheduler should be easy to integrate with its data sources and sinks.** The scheduler needs to get the problem data from somewhere. While it is possible for the user to enter the data by hand or create a file with the data, in the real world the data usually comes from another application. This could be a large relational database, or a smaller application such as Excel or Access, or even another scheduler (e.g., in a supply chain). The scheduler also often needs to write its schedules to other applications, e.g. billing or payroll applications, that utilize the schedules. Developing these interfaces from scratch can be difficult and time-consuing, and can be a major driver of the total cost, so it is best to make this integration as easy as possible.

**The scheduling system should support the "accessories" needed to make it practical.** For most users, an algorithm that automatically creates schedules from scratch is only the beginning of what they need from an automated scheduling system. They also need dynamic rescheduling, i.e. an algorithm that monitors a schedule as it is executed and repairs it as necessary. Furthermore, they need to get humans involved in the scheduling process. They need displays that allow humans not only to visualize schedule but also to provide inputs to the scheduler on what to do, i.e. mixed-initiative scheduling. Many scheduling algorithms do not support dynamic rescheduling or mixed-initiative scheduling, and this can become a big issue when there is a need for these capabilities.

## 2 Potential Solutions from the Researchers

Here are some steps that the scheduling research community could take to address the problems from the last section.

**Scheduling researchers should solve real-world problems more often.** Yes, it is still very important to develop theory and general-purpose tools not specific to a particular problem. However, the work on theory and tools that is not targeted at, or at least based on, real-world problems often ends up being irrelevant. Therefore, it is important that scheduling researchers seek out people and organizations with real problems to solve. If the organization is big enough and the problem is important enough, they may pay you to help them solve it. Even if the funding comes from a third party, it is worth finding real problems.

[Researchers using evolutionary algorithms (and more generally metaheuristics) for scheduling have been good about maintaining a focus on real-world problems, partly because evolutionary algorithms are particularly well suited to the representational complexities of real-world scheduling (see below). One pointer to some applications is [Montana, 1998].]

**The scheduling research community should create and promote a better, expanded set of benchmarks.** There is a saying that "you get what you measure". Currently, the commonly used benchmarks for scheduling, such as the job-shop scheduling problem and traveling salesman problem, do not give a good indication of utility for most real-world problems. One reason is that the scheduling logic of the benchmarks is too simple (even when the problems are computationally complex). Real-world problems tend to have much more convoluted constraints (on what consitutes a legal schedule) and criteria (for what makes a good schedule). Second, the only variability in the different instances of the benchmarks is the numbers used. This is unlike the real world where there is variability not just in the numbers but also in the scheduling logic between problems that are os-

tensibly the same. Third, the benchmarks are "static" and reflect none of the dynamic updates and interaction with human operators that characterize many real-world problems.

Fixing the problem with benchmarks will take a concerted effort. The first step is defining these benchmarks. There are some more realistic benchmarks available to the public, but a greater variety would be good. The second step is getting these benchmarks accepted as standard. This is particularly difficult for a few reasons. For one, there is the catch-22 that people want to use traditional benchmarks because that is the best way to gain acceptance with a general audience, but new benchmarks will not become accepted unless people use them. A second obstacle to acceptance is that some scheduling techniques, particularly operations research techniques, would not be able to handle the new benchmarks, and this is a disincentive to practitioners of these techiques to accept new benchmarks. However, for the sake of advancing the field, it is worth the effort to push for new benchmarks.

[An example of a sequence of benchmarks that captures the complexity of real-world scheduling are those available on the web at [Fox and Ringer, 1995]. They are based on data related to the scheduling of a project for manufacturing an airplane and include issues such as dynamic rescheduling. These benchmarks never caught on, and are not included in standard collections such as the OR Library [Beasley, 1990], partly because most scheduling algorithms cannot handle them.]

**Scheduling researchers should focus more on metaheuristics such as evolutionary algorithms and tabu search and less on small improvements to older techniques.** There are some distinct advantages that evolutionary algorithms and closely related techniques have over other approaches. First, they are easy to apply to almost any scheduling problem, including those with odd constraints and criteria that may derail other algorithms. Second, compared to other general-purpose optimization techniques, including standard operations research techniques, evolutionary algorithms are fast at finding good (albeit often suboptimal) solutions, particularly as the problem size increases. Third, evolutionary algorithms allow an explicit tradeoff between the search time and the quality of the solution. Fourth, evolutionary algorithms, with their population-based approach, allow for easy and effective large-scale parallelization. Fifth, evolutionary algorithms are well suited to the problems of fast schedule repair and dynamic rescheduling.

**Scheduling researchers should focus more on issues of reconfigurability.** A reconfigurable scheduler is one that can be configured for a wide range of different scheduling problems without modifying the underlying software, with the problem-specific scheduling logic specified essentially as configuration data. Reconfigurability addresses many of the issues discussed above. There is a greatly reduced cost of development because there is no coding involved. Since the code is reused, there is also a reduced cost of maintenance and training. The scheduler can be configured specifically to solve the user's problem. Plus, dynamic rescheduling and mixed-initiative scheduling come for free if they are part of the basic functionality of the reconfigurable scheduler.

Outside the evolutionary (and more generally the metaheuristic) community, reconfigurability has been considered important for a long time. The constraint programming community has always considered highly desirable the separation of the problem specification and the solver. (ILOG has a sophisticated reconfigurable scheduling product based on OPL [Van Hentenryck, 1999].) The mathematical programming community has AMPL [Fourer *et al.*, 1993] and its associated solvers that also provide this separation. However, these have both been limited: constraint programming because of the lack of true optimization and mathematical programming because of the inability to handle algorithmic constraints in addition to algebraic constraints. Our Vishnu reconfigurable scheduler is the first to combine reconfigurability with evolutionary algorithms [Montana, 2001b].

[Further discussion of the benefits of reconfigurability is contained in [Montana, 2001a].]

**The scheduling research community should create and adopt a set of standards for data formats and interfaces and should accumulate a library of open-source components that adhere to these standards.** A taste of the power that standardization brings can be seen with AMPL. Users can specify the constraints and optimization criterion for a problem in a high-level language and have a choice of different solvers, with the option of picking the one that works best for their particular problem. Developers can release their coded algorithms, knowing that users can potentially employ them without the need to learn about new representation formats or to reexpress their problem in these new formats.

In full scheduling systems, there are a variety of different components besides the automated scheduler. Displays show Gantt charts and data as well as accepting user inputs. There are data interfaces that

read from and write to not just file formats but also from applications such as Excel, Access, and Oracle. Furthermore, there can be a centralized database controlling access to the data and schedules. It is very difficult for one researcher or one research institution to develop a full scheduling system. However, if the scheduling community were to define a standardized architecture and standardized interfaces, researchers could instead develop components, knowing that they can easily be plugged into a larger system. Hence, researchers would have a direct path to get their software to users instead of just publishing articles and hoping that somebody else implements their ideas. This allows individual researchers to compete on an equal footing with a company such as ILOG that has a full, integrated product line and the resources to support it. This component-based approach is particularly powerful when combined with an open-source distribution model (as discussed in [Montana, 2001a]). This allows users to experiment freely with different components and to tailor the components to their own ends.

[An early attempt at such standardization was the Scheduling Applications Interface Language (SAIL) [ Hull, 1995], but it never caught on. As part of Vishnu, we have recently developed an XML-based representation for scheduling problems and solutions that could serve as the beginning of such a standardized representation [Montana, 2001b]. The DARPA Agent Markup Language (DAML) [Hendler and McGuinness, 2000] could be an even better language for a standard representation than XML (of which it is an extension) because of its ability to represent semantic content.]

**The scheduling research community should create and adopt standard protocols for communication and coordination between automated schedulers.** There are times when automated schedulers need to communicate with each other, negotiating deals and coordinating schedules. For example, the automated schedulers for a supplier and a consumer might need to negotiate about what the supplier is delivering and when, since it effects both schedules. Another example is that two schedulers that share resources might need to coordinate their use of these resources. Humans are good at this negotiation, but machines traditionally are not. Defining standard protocols for how automated schedulers communicate with each other could lay the groundwork for automated negotation of schedules. This is likely to be important in the near future, as this is an enabling technology for automated supply chains and many automated business-to-business (B2B) transactions.

[For our work on multiagent scheduling [Montana *et al.*, 2000], we have used the Cougaar multiagent architecture (available at www.cougaar.org) for communication between and coordination of the scheduling agents. DAML provides an alternative for defining ontologies and protocols for interagent communication.]

## References


[Beasley, 1990] J. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[Fourer *et al.*, 1993]
R. Fourer, D. Gay, and B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Belmont, CA, 1993.

[Fox and Ringer, 1995] B. Fox and M. Ringer. Planning and scheduling benchmarks, 1995. http://www.neosoft.com/ benchmrx/.

[Hendler and McGuinness, 2000] J. Hendler and D. McGuinness. The DARPA agent markup language. *IEEE Intelligent Systems*, 15(6):67–73, 2000.

[Hull, 1995] L. Hull. SAIL reference manual, 1995.

[Montana *et al.*, 2000] D. Montana, J. Herrero, G. Vidaver, and G. Bidwell. A multiagent society for military transporation scheduling. *Journal of Scheduling*, 3(4):225–246, 2000.

[Montana, 1998] D. Montana. Introduction to the special issue: Evolutionary algorithms for scheduling. *Evolutionary Computation*, 6(1):v–ix, 1998.

[Montana, 2001a] D. Montana. Optimized scheduling for the masses. In *Genetic and Evolutionary Computation Conference Workshop Program*, pages 132–136, 2001.

[Montana, 2001b] D. Montana. A reconfigurable optimizing scheduler. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1159–1166, 2001.

[Van Hentenryck, 1999] P. Van Hentenryck. *The OPL Optimization Programming Language*. MIT Press, Cambridge, MA, 1999.