Design Issues for Software Tools for Optimization

Andrew Davenport
IBM T.J.Watson Research Center, Yorktown Heights, New York 10598
davenport@us.ibm.com

Abstract

We discuss issues relating to the deployment of complex technologies from the fields of artificial intelligence and operations research. We argue that for such technologies to gain wider acceptance in the user community, more emphasis must be placed on designing intuitive interfaces at the right level of abstraction for non-experts to use.

Researchers in the fields of artificial intelligence and operations research have been very successful at developing technologies for solving combinatorial optimization problems as they arise in range of industries, from operations planning and scheduling (Okano *et al.* 2004; Kalagnanam *et al.* 2000), to product design, configuration and procurement (Hohner *et al.* 2003). Success stories from applying such innovative technologies are presented frequently at conferences such as IAAI, INFORMS, and the INFORMS Edelman Award series. However, most of these success stories describe projects involving active researchers with deep expertise in their particular area. One of the major hurdles limiting the wider adoption of these technologies is the lack of software tools embodying these technologies that can be easily understood and deployed by non-expert developers. This is a problem for two reasons: firstly the number of experts in a particular area that can work on projects is limited, and such experts often prefer to work on innovative projects; and secondly, software developed with the help of experts often needs to be maintained by non-experts after the end of the project. Concern about maintenance issues can often limit the adoption of innovative but unproven technologies by organisations.

In our experience, we have found that talented software developers (who do not have a research background) can learn to have a very good understanding of abstractions of problems that are often encountered when solving combinatorial optimization problems, if the abstraction is at the right level. For example, we have found that although many people have difficulties understanding and using the modelling languages provided by integer programming and constraint programming packages, they are able to easily grasp the underlying concepts of higher level primitives in optimization such as network flow problems, travelling salesman problems and matching problems. Such users are often able to understand how solving a particular problem, such as an inventory matching problem (Kalagnanam *et al.* 2000), can be achieved by modelling it as a weighted bipartite matching problem, even though they do not understand the underlying algorithms for solving these problems. However an integer programming formulation of the same problem, even though it may capture more the problem's objectives and constraints, will not be understood by many developers and will be harder to maintain by them. We are not claiming that the bipartite matching problem provides the best modelling abstraction for this particular problem, but our experience is that is does provide a better modelling abstraction than integer programming. This may lead to the situation where an inferior but easy to understand model is preferred to a difficult to understand model that captures all of a problem's constraints and objectives and provides a better solution.

There is a vast amount of algorithmic research from the fields of artificial intelligence, operations research and computer science that can be applied to problems arising in industry. It should not be necessary to understand how these algorithms work in order to deploy them. However frequently it is: for instance some variation of the travelling salesman problem (TSP) often forms the core of many real world optimization problems ((Okano, Morioka, & Yoda 2002)), such as the TSP with time windows, the prize collecting TSP and the multiple TSP. However, as far as we are aware, there are no commercial tools available providing specialised and efficient implementations of algorithms for solving TSP's. This limits the adoption of research in this area to people who understand the algorithms and how to implement them, despite their wide applicability.

As a result of our experiences on applied optimization projects, we believe that research is needed into how to provide the right abstractions and modelling layers so that they can be deployed by experienced software developers, who are not experts in any of these research fields. One example of where providing the right abstraction layer for a technology whose underlying algorithms are complicated and understood by few is the field of databases. Another success in an area more closely related to optimization is LEDA, a Library of Efficient Data-types and Algorithms (Mehlhorn, Näher, & Uhrig 1997). LEDA provides a C++ library of commonly used data structures, including graph data struc-

Copyright © 2004, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

tures and visualization tools, as well as efficient implementations of a wide range of polynomial time graph based algorithms, such as for solving shortest path problems, matching problems and network flow problems. LEDA can be used easily by software developers to solve such problems without understanding how the underlying algorithms work.

Many optimization problems are NP-hard however, and require expertise in modelling and implementation. Standard problem models, such as TSP, knapsack problems and job shop scheduling, may not be sufficient to capture all the constraints in a problem. General purpose solvers, such as those based on integer and constraint programming, do not provide modelling languages geared towards the intuitions of the non-expert software developer. Some promising initial research in constraint programming has identified the use of design patterns as a way of specifying modelling expertise for constraint programs (Walsh 2003). We believe that further research into appropriate level of modelling abstractions is necessary for optimization technology to reach its full potential.

References

Hohner, G.; Reid, G.; J., R.; Ng, E.; Davenport, A.; Kalagnanam, J.; Lee, H. S.; and An, C. 2003. Combinatorial and quantity-discount procurement auctions benefit mars, incorporated and its suppliers. *Interfaces* 33(1).

Kalagnanam, J.; Dawande, M.; Trumbo, M.; and Lee, H. S. 2000. The surplus inventory matching problem in the process industry. *Operations Research* 48(4).

Mehlhorn, K.; Näher, S.; and Uhrig, C. 1997. The LEDA platform for combinatorial and geometric computing. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming (ICALP'97)*, 7–16. Springer-Verlag, LNCS 1256.

Okano, H.; Davenport, A.; Trumbo, M.; Reddy, C.; Yoda, K.; and Amano, M. 2004. Finishing line scheduling in the steel industry. Technical report, IBM Research.

Okano, H.; Morioka, T.; and Yoda, K. 2002. A heuristic solution for the continuous galvanizing line scheduling problem in a steel mill. Technical Report RT0478, IBM Research Report.

Walsh, T. 2003. Constraint patterns. In *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming (CP2003)*.