In: Proceedings of the 2018 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), Delft, Netherlands, June 2018

On Learning from Human Expert Knowledge for Automated Scheduling

Neil Yorke-Smith
Delft University of Technology, Netherlands, and
American University of Beirut, Lebanon
n.yorke-smith@tudelft.nl

Abstract

Automated scheduling systems and decision support tools require at least four kinds of knowledge: 1) domain knowledge, 2) problem instance knowledge, 3) control knowledge, and 4) solving knowledge. This short paper draws attention to learning from human experts for these different kinds of knowledge, and advocates a complementarity of knowledge acquisition by automated techniques and by human knowledge engineers.

Introduction

Knowledge – computational knowledge – is the fulcrum of Artificial Intelligence. Whether hand-coded in a logical formalism, or extracted from data by a deep learning network, knowledge is the basis for computation. AI-based scheduling and planning is no different. Take a now-ubiquitous 'intelligent' personal assistant agent. One of the pain points helped by an assistant like Siri is scheduling meetings and managing your calendar (Berry et al. 2011). This assistance is based on knowledge of your calendar, to-do list, emails, location – and learned preferences.

If knowledge for computation is the fulcrum, then AI rests on its acquisition. As the KEPS workshop organizers put it, automated planning and scheduling systems "still need to be fed by carefully engineered domain and problem descriptions, and fine tuned for particular domains and problems." This position paper draws attention to learning from human experts as a way to accelerate the knowledge engineering process, which we take to comprise both elicitation and encoding.

We briefly discuss four of the kinds of knowledge required for automated scheduling:

1. **Domain knowledge**. What is the 'physics' of the problem domain?
2. **Problem knowledge**. What are the particulars of the problem instance, including its data and objectives?
3. **Control knowledge**. How does the system go about deciding how to solve the problem instance, and manage the solving process?
4. **Solving knowledge**. What solving approaches and heuristics can be used?

Information science has for decades distinguished between data, information, knowledge, and wisdom (Ackoff 1989). According to Ackoff's oft-quoted taxonomy, starting from the broad base layer of a pyramid and progressing to its narrow pinnacle layer[1], we have:

- Data: raw symbols ('know-nothing' (Zeleny 1987))
- Information: data that is processed to be useful; provides answers to 'who', 'what', 'where' and 'when' questions (know-what)
- Knowledge: application of data and information; answers 'how' questions (know-how)
- Wisdom: evaluated appreciation of 'why' (know-why)

Seen with this lens, the activity of 'knowledge engineering' – such as for an automated scheduling system – aims to apply raw data and processed data in order to support the answering of 'how' questions. For example, the calendaring assistant can (has the know-how to) arrange a meeting with Alice and Bob for next week. While not dwelling on nuances of terminology, we can see domain and problem knowledge as fitting closer to Ackoff's Information level, and control and solving knowledge as fitting closer to his Knowledge level.

We advocate a complementarity of knowledge acquisition by automated techniques and by human knowledge engineers, for the purpose of automated scheduling.

Learning for Domain and Problem Acquisition

The power of automated planning and scheduling systems comes from the combination of the model of the problem and the problem-solving techniques applied to that model. Two elements comprise the former: the model of the domain, and the model of the problem instance. The domain model tells us what is possible, while the problem instance models the questions we want to answer.

As surveyed by Vaquero et al. (2013), real-world problems require detailed knowledge elicitation, encoding and management. These authors' methodology, itSIMPLE, strives to use common notations such as UML in a process of moving from requirements analysis all the way to an input-ready model for solving algorithms.

This kind of methodology, which starts from a graphical representation used to represent statements from subject matter experts (SMEs), is found not only in AI planning and

[1] We follow a number of authors and join Ackoff's Understanding and Wisdom layers.

scheduling, but across other areas of AI such as organizational modelling and agent-based simulation (van Putten et al. 2008) and goal-oriented programming (Abushark et al. 2017).

In line with the rise of machine learning (ML), we can undertake automated acquisition of domain and problem models from data, as Celorrio et al. (2012) survey for AI planning. Since that survey, there has followed much more work on learning domains and problem instances.

Pushing further with learning from data, Lombardi, Milano, and Bartolini (2017) propose a strongly empirical approach to model learning for general combinatorial optimization problems, using ML to construct components of a model from data. The model encompasses both problem domain and instance. The data is obtained either from a relaxed version of the model (a form of boot-strapping), or if applicable and possible, from the modelled system itself. The SME could be involved in creating the initial approximate model; otherwise this approach is driven by data.

A key question is *representation*: how do we formulate the domain and problem models? There are various representations for AI planning, including standard languages such as PDDL, and ML approaches to acquire models into these representations are actively researched.

By contrast to planning, quite commonly scheduling problems have a fixed structure of domain and data, such as flow shop scheduling and other classical Operational Research (OR) scheduling problems. Learning into these representations is more straightforward, and it can suffice to learn from data – since the human expertise has already been put into defining the problem structure. For example, a knowledge engineer encodes the problem as a flow shop with a cyclic job structure, and obtains the data from instrumentation embedded in the manufacturing process. We note that a difficulty, however, given classical OR models is that the modeller can be tempted to coerce the actual problem at hand into one of the standard models, for the sake of convenience, tractability, or the assurance of familiarity.

A more general model for scheduling problems is based on Constraint Satisfaction Problem (CSP): see Salido et al. (2007) for a typical example. ML approaches to acquire (general) CSPs are also actively researched (O'Sullivan 2010; Beldiceanu and Simonis 2016; Bessiere et al. 2017). Bessiere et al. (2017) exemplify this line of work, in deriving CSP models – which like Lombardi, Milano, and Bartolini (2017)'s approach encompass both domain and instance – from a user; both passive and active elicitation are supported.

We advocate for a position that uses data-driven methods as much as possible, and hand-engineered methods in all other aspects. The advantage is to attempt to gain the best from both types of methods: automation and parsimony, and judgement and completeness. In some cases, the two can be used together to triangulate certain knowledge. In other cases, the knowledge acquired with ML can form the starting point for the knowledge engineer's refining of models. In still other cases, manual knowledge engineering can provide or structure data far enough so that ML can then be used.

Learning for Control and Problem Solving

Control knowledge decides what search and reasoning strategies to apply in a problem-solving process, and can adjust the strategies as solving proceeds. Problem solving knowledge comprises of the available strategies, in particular those suited to the domain or to the problem instance. Hence control knowledge is predicated on having problem solving knowledge available to it.

A potent recipe for control knowledge consists of portfolio solving approaches, in which control knowledge is acquired in the form of selection among solving algorithms for a problem instance. Portfolio approaches have proved successful in several subareas of AI, such as SAT (Xu et al. 2008) and automated planning (Gerevini, Saetti, and Vallati 2014). Beck and Freuder (2004) is one example of a portfolio approach specifically for a scheduling problem.

We identify problem solving knowledge for scheduling as 'heuristics'. The literature is substantial on learning how to solve a particular scheduling problem or class of problems (e.g., (Li, Pan, and Mao 2015)). Perhaps because scheduling problems tend to have structure – and at that often a variant of a standard structured problem class, as we have noted – it is easier for scheduling problems than for planning problem to hand-code the problem domain, and extract instance data from some book-keeping system or instrumentation.

Hence the focus of ML for scheduling is drawn to solving problem instances. In a now-classic paper, Gratch, Chien, and DeJong (1993) learn control knowledge for an aerospace scheduling problem; a whole literature on learning metaheuristics is now known. Shahrabi, Adibi, and Mahootchi (2017) is a recent example of learning control knowledge for scheduling, using reinforcement learning. Examples of heuristic learning are many (Russell et al. 2009; Braune and Doerner 2017).

In contrast to this kind of work, which focuses on learning from data, Alzugaray and Sanfeliu (2016) learn path planning heuristics from human expert problem solvers. The point here is that the experts may not be cognisant of their own strategies: they cannot articulate them fully.

Similarly, Berry et al. (2011) learn users' calendering preferences (heuristics) from user actions, with explicit elicitation being optional. These authors found that users' stated preferences often differ from their preferences exposed by their actions. Generalizing, Gombolay et al. (2016a; 2016b) coin the term 'apprenticeship scheduling' for apprenticeship learning techniques applied to scheduling problems. These authors learn heuristics from experts' actions, and in two domains show how an automated scheduling system performs at or exceeds a level that satifices for human approval.

We note that a difficulty of learning from a scheduling problem can arise when the data is gathered under a certain policy (objective, schedule) that we are now trying to optimize. This situation amounts to the classic exploration/exploitation dilemma in reinforcement learning.

We advocate for a position that uses data-driven methods inasmuch as data is available; using learning from experts, particularly to acquire expert knowledge that the SMEs cannot articulate; and using "fine tuned" interventions of knowl-

edge engineers in synergy with the ML techniques. Indeed, it has not passed un-noticed that the engineering of a ML model can be as much effort as manually engineering solving strategies and heuristics (Domingos 2012).

Outlook

We have considered four types of knowledge of automated scheduling, and drawn attention to learning from human experts for these different kinds of knowledge. We note that SME knowledge, and certainly human decision-making, may be imperfect; both (semi-)automated and manual knowledge engineering must recognize this. In advocating a complementarity of knowledge acquisition by automated techniques and by human knowledge engineers, we anticipate a growth in the KEPS sub-community and fruitful interactions with the ML community, including reinforcement learning. Let us take up this opportunity.

Acknowledgements Thanks to the KEPS workshop reviewers for their suggestions. The author also thanks J. Shah, M. Spaan, E. Walraven, M. de Weerdt and C. Witteveen.

References

Abushark, Y.; Miller, T.; Thangarajah, J.; Winikoff, M.; and Harland, J. 2017. Requirements specification via activity diagrams for agent-based systems. *Autonomous Agents and Multi-Agent Systems* 31(3):423–468.

Ackoff, R. 1989. From data to wisdom. *Journal of Applied Systems Analysis* 16:3–9.

Alzugaray, I., and Sanfeliu, A. 2016. Learning the hidden human knowledge of UAV pilots when navigating in a cluttered environment for improving path planning. In *Proc. of IROS'16*, 1589–1594.

Beck, J. C., and Freuder, E. C. 2004. Simple rules for low-knowledge algorithm selection. In *Proc. of CP-AI-OR'04*, 50–64.

Beldiceanu, N., and Simonis, H. 2016. ModelSeeker: Extracting global constraint models from positive examples. In *Data Mining and Constraint Programming – Foundations of a Cross-Disciplinary Approach*, volume 10101 of *Lecture Notes in Computer Science*. Springer. 77–95.

Berry, P. M.; Gervasio, M. T.; Peintner, B.; and Yorke-Smith, N. 2011. PTIME: Personalized assistance for calendaring. *ACM Transactions on Intelligent Systems and Technology* 2(4):40:1–40:22.

Bessiere, C.; Koriche, F.; Lazaar, N.; and O'Sullivan, B. 2017. Constraint acquisition. *Artifical Intelligence* 244:315–342.

Braune, R., and Doerner, K. F. 2017. Real-world flexible resource profile scheduling with multiple criteria: Learning scalarization functions for MIP and heuristic approaches. *Journal of the Operational Research Society* 68(8):952–972.

Celorrio, S. J.; de la Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *Knowledge Engineering Review* 27(4):433–467.

Domingos, P. M. 2012. A few useful things to know about machine learning. *Communications of the ACM* 55(10):78–87.

Gerevini, A.; Saetti, A.; and Vallati, M. 2014. Planning through automatic portfolio configuration: The PbP approach. *Journal of Artificial Intelligence Research* 50:639–696.

Gombolay, M. C.; Jensen, R.; Stigile, J.; Son, S.; and Shah, J. A. 2016a. Apprenticeship scheduling: Learning to schedule from human experts. In *Proc. of IJCAI'16*, 826–833.

Gombolay, M. C.; Yang, X. J.; Hayes, B.; Seo, N.; Liu, Z.; Wadhwania, S.; Yu, T.; Shah, N.; Golen, T.; and Shah, J. A. 2016b. Robotic assistance in coordination of patient care. In *Proc. of RSS'16*.

Gratch, J.; Chien, S. A.; and DeJong, G. 1993. Learning search control knowledge for deep space network scheduling. In *Proc. of ICML'93*, 135–142.

Li, J.; Pan, Q.; and Mao, K. 2015. A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems. *Engineering Applications of AI* 37:279–292.

Lombardi, M.; Milano, M.; and Bartolini, A. 2017. Empirical decision model learning. *Artifical Intelligence* 244:343–367.

O'Sullivan, B. 2010. Automated modelling and solving in constraint programming. In *Proc. of AAAI'10*, 1493–1497.

Russell, T.; Malik, A. M.; Chase, M.; and van Beek, P. 2009. Learning heuristics for the superblock instruction scheduling problem. *IEEE Transactions on Knowledge and Data Engineering* 21(10):1489–1502.

Salido, M. A.; Abril, M.; Barber, F.; Ingolotti, L. P.; Tormos, M. P.; and Lova, A. L. 2007. Domain-dependent distributed models for railway scheduling. *Knowledge-Based Systems* 20(2):186–194.

Shahrabi, J.; Adibi, M. A.; and Mahootchi, M. 2017. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Computers & Industrial Engineering* 110:75–82.

van Putten, B.; Dignum, V.; Sierhuis, M.; and Wolfe, S. R. 2008. OperA and Brahms: A symphony? In *Proc. of AAMAS'08 Workshop on Agent-Oriented Software Engineering (AOSE'08)*, 257–271.

Vaquero, T. S.; Silva, J. R.; Tonidandel, F.; and Beck, J. C. 2013. itSIMPLE: Towards an integrated design system for real planning applications. *Knowledge Engineering Review* 28(2):215–230.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research* 32:565–606.

Zeleny, M. 1987. Management support systems: Towards integrated knowledge management. *Human Systems Management* 7(1):59–70.