Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Opportunities and Challenges for Constraint Programming*

Barry O'Sullivan
Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
b.osullivan@cs.ucc.ie

Abstract

Constraint programming has become an important technology for solving hard combinatorial problems in a diverse range of application domains. It has its roots in artificial intelligence, mathematical programming, operations research, and programming languages. This paper gives a perspective on where constraint programming is today, and discusses a number of opportunities and challenges that could provide focus for the research community into the future.

Introduction

Constraint programming (CP) is a technology for solving combinatorial optimisation problems (Rossi, Beek, and Walsh 2006). A major generic challenge that faces CP is *scalability*. This is, traditionally, because the problems that it is usually applied to are computationally intractable (NP-Hard). While CP has been successfully applied in domains such as scheduling, timetabling, planning, inventory management and configuration, many instances of these problems are extremely challenging for traditional CP methods due to their hardness.

However, an emerging dimension of scale relates to problem size, and the volume of data available that is relevant to solving a particular instance, e.g. extremely large domain sizes, or very large extensionally defined constraints of high arity. In 2009 information on the web was doubling every 18 months. It is now believed that this occurs in less than 12 months. This exponential growth in data, often referred to as the "big data" challenge, presents us with major opportunities. For example, McKinsey Global Institute estimates that European government administrations could benefit from over €250 billion in operational efficiencies by properly exploiting "big data".

Another major challenge in real-world application domains is *uncertainty* (Verfaillie and Jussien 2005). For example, in scheduling, the duration of a task might be uncertain, while in inventory management there might be uncertainty related to customer demand. Surprising, even predicting the future does not imply we can make better decisions. The interactions between the choices that face us are usually interlinked in complex ways. Being able to react appropriately to risk is more important than knowing about the risk or even modelling it. The traditional "get data model implement"-cycle is no longer sufficient in most domains. We often need to deal with large amounts of rapidly changing data whereby adaptation becomes key. The study of managing complex sources of data upon which we must make complex, risky, economic, or environmentally important, decisions provides a compelling context for constraint programming.

The next section will present a brief overview of some recent advances in constraint programming. The subsequent sections will highlight some important emerging application domains which will motivate a set of technical challenges for constraint programming that offer rich opportunities for both societal and economic impact.

A Brief State-of-the-Art Review of CP

Advances in constraint programming have been implemented in a variety of software toolkits, languages and systems. For example, OPL (Hentenryck et al. 1999), Minizinc (Nethercote et al. 2007) and NUMBERJACK (Hebrard, O'Mahony, and O'Sullivan 2010) provide rich modelling functionalities in a high-level language especially designed for specifying combinatorial optimisation problems. Other systems, such as CHIP (Dincbas et al. 1988), Prolog III (Colmerauer 1990), Comet (Van Hentenryck and Michel 2005), Gecode[1], IBM ILOG CP, and CPInside (Feldman, Freuder, and Little 2009), provide general programming capabilities. Many systems also exist for combinatorial optimisation using mixed-integer programming, e.g. IBM ILOG CPLEX and SCIP (Achterberg et al. 2008).

Significant progress has been made in the development of techniques that assist in the *acquisition* of models of problems through the use of machine learning techniques (Bessiere et al. 2005; Lallouet and Legtchenko 2005; Wilson, Grimes, and Freuder 2007; Mizoguchi and Ohwada 1992; O'Connell, O'Sullivan, and Freuder 2002; Coletta et al. 2003; Bessiere et al. 2004; Vu and O'Sullivan 2008). Advances in automated *reformulation* ensure that the resulting

*This paper was invited as a "Challenge" paper to the AAAI'12 Sub-Area Spotlights track.

[1] http://www.gecode.org/

CSP can be solved as efficiently as possible. Recently a large number of papers have appeared that study the reformulation of specifications of CSPs (Frisch et al. 2005b; 2005a; 2007; Gent, Miguel, and Rendl 2007; Charnley, Colton, and Miguel 2006; Colton and Miguel 2001; Bessière, Coletta, and Petit 2007).

In constraint programming the concept of a "constraint" is key. A major focus is on the development of ad-hoc constraints that provide constraint programmers with an expressive vocabulary for modelling problems (Régin 1994).

It is recognised within the CP community that different solvers are better at solving different problem instances, even within the same problem class (Gomes and Selman 2001; Xu et al. 2007; Leyton-Brown, Nudelman, and Shoham 2002; Streeter, Golovin, and Smith 2007; Sayag, Fine, and Mansour 2006). Several systems have been proposed that use machine learning to select from a portfolio of solvers the one that is best suited to a given problem instance, e.g. SATZILLA (Xu et al. 2007) and AQME (Pulina and Tacchella 2007). CPHYDRA (O'Mahony et al. 2008).

Challenge Domains

A number of important application domains are emerging due to the growth in large-scale and complex computing infrastructure, a desire to improve business efficiency, the need for improved quality in public servies, and smart and sustainable environments. Four examples are briefly discussed below.

Data Centre Optimisation. The *EU Stand-by Initiative* recently published a Code of Conduct for Energy Efficiency in Data Centres. In 2007 western European data centres consumed 56 Tera-Watt Hours (TWh) of power, which is expected to almost double to 104 TWh per year by 2020. Such a rise in energy consumption is likely to present problems for EU energy and environmental policies. To minimise carbon emissions and the strain on energy infrastructure it is important that the energy efficiency of data centres is maximised.

It is possible to minimise the energy needs of a data centre through energy-aware workload consolidation (Srikantaiah, Kansal, and Zhao 2008), which is a challenging optimisation problem. Speed and latency factors motivate geographically distributing data centres (Greenberg et al. 2009). The energy cost per unit of computation can vary significantly between two different locations due to regional demand differences, transmission inefficiencies, and generation diversity (Qureshi et al. 2009). Solving such problems is challenging because of their scale, the level of uncertainly involved in predicting energy price and load, as well as the requirement for fast response/reconfiguration times.

Innovative Enterprise and Public Service Delivery. There is a ubiquitous demand for personalised and optimised public service delivery. For example, in a hospital setting data is collected on its patients in a patient management system (PMS). Based on this data, operational schedules for the hospital are created. In an ideal hospital, no beds would be unoccupied and patients would not have to be on a waiting list for a long period. However, due to the dynamic nature of the hospital, planning is difficult. By analysing the data available at the hospital, we can discover weaknesses in its planning policies; this in turn can lead to recommendations for improved planning, which can be implemented in improved planning systems. Such intelligence can be used to support personalised and transparent service delivery to the patient.

Human Mobility and Smart Cities. This is a challenge for most large cities, with problems ranging from public transport planning to controlling dynamic traffic management systems (Giannotti et al. 2010). A typical goal is to satisfy the mobility requirements of a population by, for example, designing, planning and timetabling transportation networks. Traces of human mobility can be observed through a variety of sensing technologies, such as mobile phone networks, GPS traces, web-based calendars and social networks, etc. (Giannotti et al. 2010). Data mining can be used to extract, from individual movement and sensor data, patterns that characterise emerging requirements that transportation planning should meet. In turn, such requirements are affected by current schedules, which can be generated using constraint programming technology.

Natural Resource Management. Sustainable development focuses on ensuring that current actions do not improperly impact upon future generations. Computational sustainability is an interdisciplinary field that aims to apply techniques from computer science, information science, operations research, applied mathematics, and statistics for balancing environmental, economic, and societal needs for sustainable development.[2] Many optimisation-based challenges arise in this domain such as the sustainable management of forestry, fisheries, and agriculture.

Technical Challenges

In this section a set of specific technical challenges are presented, motivated by the complexities of decision making in data-rich domains like those highlighted above. There is the general challenge of integrating CP with other technical disciplines to provide a holistic solution to specific classes of problems, or to address the requirements of particular application domains. Therefore, CP must integrate with a variety of other technical domains in order to meet these challenges such as: machine learning; data mining; game theory; simulation; knowledge compilation; visualization; control theory; engineering; medicine and health; bioscience; and mathematics. Domain-specific integrations must also emerge in areas such as: life sciences, sustainability, energy efficiency, the web, social sciences, and finance.

A number of scientific advances are required in order to address the specific kinds of challenges raised in the context of extremely large and dynamic data sources. In contrast to the standard setting in which constraint programming and operations research is applied, in the domains described above the data upon which the specific problem instance

[2] http://www.cis.cornell.edu/ics/

is defined is subject to constant change, e.g. real-time energy prices, connections in a social network, which in turn changes the nature of the objective function of the problem. As a consequence the features of the optimisation algorithm must also evolve. Below we present the specific challenges in constraint programming that are motivated by such domain characteristics.

Optimisation of Evolving Problems. In the presence of highly dynamic data, problem models and the associated objective functions evolve. This is closely related to Open Constraint Optimisation (Faltings and Macho-Gonzalez 2003; 2005; Maher 2009), but that framework makes assumptions about the order in which options are discovered, which can be exploited algorithmically. Problem models, and their associated solvers, will need to be parameterised within a time window within which the model is considered valid. This is closely related to the use of learning distributions from historical data in stochastic optimisation (Bent and Hentenryck 2005), which attempts to study the effect of relaxing the assumption that a distribution of future requests is available at "optimisation time". Other approaches to this problem relate to filtering historical data to reflect online observations (Ikonomovska et al. 2009).

Online Optimisation and Optimal Stopping Theory. Recently there has been a focus on solving combinatorial problems in an online setting (Hentenryck and Bent 2006), e.g. reservation systems, allocating jobs dynamically in a data centre, etc. Optimal stopping theory is concerned with the problem of choosing a time to take a given action based on sequentially observed random variables in order to maximise an expected payoff or to minimise an expected cost (Bruss 2000; Zheng, Ge, and Zhang 2009). The classic problem in this domain is the 'Secretary Problem' whereby one must choose the best candidate from amongst n for a job immediately having conducted an interview, without the facility of recalling previous candidates or observing future ones. Such kinds of online problem occur in many domains such as energy-cost minimisation and the pricing of financial options. A challenging research direction lies in the development of online stochastic optimisation methods that can exploit the guarantees that optimal stopping theory can offer, or use optimal stopping theory to decide when specific actions should be taken.

Automated Large-Neighbourhood Search. In our experience of solving extremely large real-world problems, large-neighbourhood search (Perron, Shaw, and Furnon 2004) has proven to be an excellent technique. Large-neighbourhood search (LNS) is a meta-heuristic approach whereby an incomplete local search algorithm identifies parts of a solution to be improved by a systematic algorithm. In order to successfully apply LNS to a new problem one needs to understand how different definitions of neighbourhood affect the objective function. This is extremely complex in problem settings that are subject to frequent and substantial change. An interesting research challenge is to develop LNS methods that use machine learning techniques to assist in defining useful neighbourhoods as problem scale rapidly increases. Existing approaches to adaptive LNS, e.g. (Laporte, Musmanno, and Vocaturo 2010) use weights during search to try to identify useful subsets of variables to define the best neighbourhood. Machine learning techniques, especially feature selection methods, have not yet been applied to this problem. Thus, motivating an integration of machine learning and constraint programming.

Exploiting Problem Structure. Knowledge compilation and complexity theory can help us exploit the structure of real world problems. Compiling parts of the problem offline can also lead to scalable online reasoning (Darwiche and Marquis 2002). Parameterised complexity deals with computational intractability by identifying problems whose exponential worst-case behaviour is only polynomially dependent on the size of the problem n, but exponentially dependent on a parameter k, independent of n (Neidermeier 2006). Many global constraints are fixed parameter countable. These results can be used in conjunction with compilation methods to assist in designing very scalable LNS methods, search heuristics and problem reformulation methods that can improve the scalability of optimisation.

Online Distributed Stochastic Optimisation. Consider a road network, which is an example of a shared resource. Improving the quality of some routes over others can alter the behaviour of drivers. An interesting challenge is to develop optimisation techniques that reason about the behaviour that they incentivise so as to maximise the efficiency with which these resources are utilised, avoiding phenomena such as Braess' Paradox (Roughgarden 2006; Valiant and Roughgarden 2006). Braess' Paradox arises in settings, such as network design, where as a consequence of adding capacity to a system self-interested agents modify their behaviour in a fashion that reduces overall system performance. An integration with algorithmic game theory (Nisan et al. 2007) is particularly relevant here.

Conclusion

In this paper a number of general challenges for constraint programming were presented, motivated by the emergence of application domains characterised by the availability of large and complex data sources. Such domains provide strong motivation for the study of novel hybrid methods in constraint programming, and novel integrations with other fields, some within the more general AI family.

Acknowledgements

This work was supported by Science Foundation Ireland Grant 10/IN.1/3032.

References

Achterberg, T.; Berthold, T.; Koch, T.; and Wolter, K. 2008. Constraint integer programming: A new approach to integrate CP and MIP. In Perron, L., and Trick, M. A., eds., *CPAIOR*, volume 5015 of *Lecture Notes in Computer Science*, 6–20. Springer.

Bent, R., and Hentenryck, P. V. 2005. Online stochastic optimization without distributions. In Biundo, S.; Myers, K. L.; and Rajan, K., eds., *ICAPS*, 171–180. AAAI.

Bessiere, C.; Coletta, R.; Freuder, E. C.; and O'Sullivan, B. 2004. Leveraging the learning power of examples in automated constraint acquisition. In *Proceedings of CP 2004*, LNCS 3258, 123–137.

Bessiere, C.; Coletta, R.; Koriche, F.; and O'Sullivan, B. 2005. A SAT-based version space algorithm for acquiring constraint satisfaction problems. In *Proceedings of ECML 2005*, 23–34.

Bessière, C.; Coletta, R.; and Petit, T. 2007. Learning implied global constraints. In *IJCAI*, 44–49.

Bessiere, C., ed. 2007. *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*. Springer.

Bruss, F. T. 2000. Sum the odds to one and stop. *Annals of Probability* 28:13841391.

Charnley, J.; Colton, S.; and Miguel, I. 2006. Automatic generation of implied constraints. In Brewka, G.; Coradeschi, S.; Perini, A.; and Traverso, P., eds., *ECAI*, volume 141, 73–77. IOS Press.

Coletta, R.; Bessiere, C.; O'Sullivan, B.; Freuder, E. C.; O'Connell, S.; and Quinqueton, J. 2003. Constraint acquisition as semi-automatic modeling. In *Proceedings of AI-2003*, 111–124. Cambridge, UK: Springer.

Colmerauer, A. 1990. An introduction to Prolog III. *Commun. ACM* 33(7):69–90.

Colton, S., and Miguel, I. 2001. Constraint generation via automated theory formation. In Walsh, T., ed., *CP*, volume 2239 of *Lecture Notes in Computer Science*, 575–579. Springer.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *J. Artif. Intell. Res. (JAIR)* 17:229–264.

Dincbas, M.; van Hentenryck, P.; Simonis, H.; and Aggoun, A. 1988. The Constraint Logic Programming Language CHIP. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, 693–702.

Faltings, B., and Macho-Gonzalez, S. 2003. Open constraint optimization. In Rossi, F., ed., *CP*, volume 2833 of *Lecture Notes in Computer Science*, 303–317. Springer.

Faltings, B., and Macho-Gonzalez, S. 2005. Open constraint programming. *Artif. Intell.* 161(1-2):181–208.

Feldman, J.; Freuder, E. C.; and Little, J. 2009. CP-INSIDE: Embedding constraint-based decision engines in business applications. In van Hoeve, W. J., and Hooker, J. N., eds., *CPAIOR*, volume 5547 of *Lecture Notes in Computer Science*, 323–324. Springer.

Frisch, A. M.; Hnich, B.; Miguel, I.; Smith, B. M.; and Walsh, T. 2005a. Transforming and refining abstract constraint specifications. In Zucker, J.-D., and Saitta, L., eds., *SARA*, volume 3607 of *Lecture Notes in Computer Science*, 76–91. Springer.

Frisch, A. M.; Jefferson, C.; Hernández, B. M.; and Miguel, I. 2005b. The rules of constraint modelling. In Kaelbling, L. P., and Saffiotti, A., eds., *IJCAI*, 109–116.

Frisch, A. M.; Grum, M.; Jefferson, C.; Hernández, B. M.; and Miguel, I. 2007. The design of essence: A constraint language for specifying combinatorial problems. In *IJCAI*, 80–87.

Gent, I. P.; Miguel, I.; and Rendl, A. 2007. Tailoring solver-independent constraint models: A case study with Essence' and Minion. In Miguel, I., and Ruml, W., eds., *SARA*, volume 4612 of *Lecture Notes in Computer Science*, 184–199. Springer.

Giannotti, F.; Nanni, M.; Pedreschi, D.; Pinelli, F.; Renso, C.; Rinzivillo, S.; and Trasarti, R. 2010. Mobility data mining: discovering movement patterns from trajectory data. In Geers, D. G., and Timpf, S., eds., *Computational Transportation Science*, 7–10. ACM.

Gomes, C. P., and Selman, B. 2001. Algorithm portfolios. *Artif. Intell.* 126(1-2):43–62.

Greenberg, A. G.; Hamilton, J. R.; Maltz, D. A.; and Patel, P. 2009. The cost of a cloud: research problems in data center networks. *Computer Communication Review* 39(1):68–73.

Hebrard, E.; O'Mahony, E.; and O'Sullivan, B. 2010. Constraint programming and combinatorial optimisation in numberjack. In Lodi, A.; Milano, M.; and Toth, P., eds., *CPAIOR*, volume 6140 of *Lecture Notes in Computer Science*, 181–185. Springer.

Hentenryck, P. V., and Bent, R. 2006. *Online Stochastic Combinatorial Optimization*. MIT Press.

Hentenryck, P. V.; Michel, L.; Perron, L.; and Régin, J.-C. 1999. Constraint programming in OPL. In Nadathur, G., ed., *PPDP*, volume 1702 of *Lecture Notes in Computer Science*, 98–116. Springer.

Ikonomovska, E.; Gama, J.; Sebastião, R.; and Gjorgjevik, D. 2009. Regression trees from data streams with drift detection. In Gama, J.; Costa, V. S.; Jorge, A. M.; and Brazdil, P., eds., *Discovery Science*, volume 5808 of *Lecture Notes in Computer Science*, 121–135. Springer.

Lallouet, A., and Legtchenko, A. 2005. Consistency for partially defined constraints. In *Proceedings of ICTAI 2005*, 118–125.

Laporte, G.; Musmanno, R.; and Vocaturo, F. 2010. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science* 44(1):125–135.

Leyton-Brown, K.; Nudelman, E.; and Shoham, Y. 2002. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In Hentenryck, P. V., ed., *CP*, volume 2470 of *Lecture Notes in Computer Science*, 556–572. Springer.

Maher, M. J. 2009. Soggy constraints: Soft open global constraints. In Gent, I. P., ed., *CP*, volume 5732 of *Lecture Notes in Computer Science*, 584–591. Springer.

Mizoguchi, F., and Ohwada, H. 1992. Constraint-directed generalization for learning spatial relations. In *Proceedings of the International Workshop on Inductive Logic Programming*, volume ICOT TM-1182.

Neidermeier, R. 2006. *Invitation to Fixed Parameter Algorithms (Oxford Lecture Series in Mathematics and Its Applications)*. Oxford University Press, USA.

Nethercote, N.; Stuckey, P. J.; Becket, R.; Brand, S.; Duck, G. J.; and Tack, G. 2007. Minizinc: Towards a standard CP modelling language. In Bessiere (2007), 529–543.

Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press.

O'Connell, S.; O'Sullivan, B.; and Freuder, E. 2002. Strategies for interactive constraint acquisition. In *Proceedings of the CP-2002 Workshop on User-Interaction in Constraint Satisfaction*, 62–76.

O'Mahony, E.; Hebrard, E.; Holland, A.; Nugent, C.; and O'Sullivan, B. 2008. Using case-based reasoning in an algorithm portfolio for constraint solving. In *Proceedings of Artificial Intelligence and Cognitive Science (AICS 2008)*.

Perron, L.; Shaw, P.; and Furnon, V. 2004. Propagation guided large neighborhood search. In Wallace, M., ed., *CP*, volume 3258 of *Lecture Notes in Computer Science*, 468–481. Springer.

Pulina, L., and Tacchella, A. 2007. A multi-engine solver for quantified boolean formulas. In Bessiere (2007), 574–589.

Qureshi, A.; Weber, R.; Balakrishnan, H.; Guttag, J. V.; and Maggs, B. V. 2009. Cutting the electric bill for internet-scale systems. In *SIGCOMM*, 123–134.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in CSPs. In *AAAI*, 362–367.

Rossi, F.; Beek, P. v.; and Walsh, T. 2006. *Handbook of Constraint Programming*. Foundations of Artificial Intelligence. New York, NY, USA: Elsevier.

Roughgarden, T. 2006. On the severity of Braess's paradox: Designing networks for selfish users is hard. *J. Comput. Syst. Sci.* 72(5):922–953.

Sayag, T.; Fine, S.; and Mansour, Y. 2006. Combining multiple heuristics. In Durand, B., and Thomas, W., eds., *STACS*, volume 3884 of *Lecture Notes in Computer Science*, 242–253. Springer.

Srikantaiah, S.; Kansal, A.; and Zhao, F. 2008. Energy aware consolidation for cloud computing. In *Proceedings of HotPower*.

Streeter, M. J.; Golovin, D.; and Smith, S. F. 2007. Combining multiple heuristics online. In *AAAI*, 1197–1203.

Valiant, G., and Roughgarden, T. 2006. Braess's paradox in large random graphs. In Feigenbaum, J.; Chuang, J. C.-I.; and Pennock, D. M., eds., *ACM Conference on Electronic Commerce*, 296–305. ACM.

Van Hentenryck, P., and Michel, L. 2005. *Constraint-Based Local Search*. The MIT Press.

Verfaillie, G., and Jussien, N. 2005. Constraint solving in uncertain and dynamic environments: A survey. *Constraints* 10(3):253–281.

Vu, X.-H., and O'Sullivan, B. 2008. A unifying framework for generalized constraint acquisition. *International Journal on Artificial Intelligence Tools* 17(5):803–833.

Wilson, N.; Grimes, D.; and Freuder, E. C. 2007. A cost-based model and algorithms for interleaving solving and elicitation of CSPs. In Bessiere (2007), 666–680.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2007. : The design and analysis of an algorithm portfolio for SAT. In Bessiere (2007), 712–727.

Zheng, D.; Ge, W.; and Zhang, J. 2009. Distributed opportunistic scheduling for ad hoc networks with random access: An optimal stopping approach. *IEEE Transactions on Information Theory* 55(1):205–222.