Constraint Programming Letters 1 (2007) 15-20 Submitted 5/2007; Published 11/2007

The Science of Constraints*

Carla Gomes GOMES@CS.CORNELL.EDU
Bart Selman SELMAN@CS.CORNELL.EDU
Department of Computer Science
Cornell University
5133 Upson Hall, Ithaca USA

Editor: Pascal Van Hentenryck

1. The Engineering of CP

Constraint Programming (CP) has moved into the real-world arena. Sophisticated constraint-based reasoning techniques have been developed bridging across different communities, such as artificial intelligence, operations research, algorithms, systems, and theoretical computer science. Examples include a vast repertoire of so-called global constraints, using e.g., network flows, dynamic programming, and automata theory, hybrid approaches integrating ideas from, e.g., mathematical programming, binary decision diagrams, and a rich set of modeling languages for global search and local search. CP techniques and methodology have become a worthy approach for solving combinatorial discrete problems, and CP is recognized as such by other communities. Over the last decade, the CP community has grown significantly, which is reflected in the number of its researchers, including well-regarded researchers from other communities, as well as the number and quality of CP journals and conferences.

An important factor in the recent growth of the field has been the pursuit of an "engineering perspective" with emphasis on tool building and practical applications. In particular, the development of practical constraint-based solvers, both in academia and in industry, has made it possible for the community to attain critical mass, fostering additional research in search, inference, and modeling techniques, and at the same time, making it possible for the application of constraint-based methods to solve real-world problems, demonstrating their competitiveness.

From an engineering perspective, CP can be viewed as a rich set of tools for constraint satisfaction and constraint optimization. This technology has become a worthwhile complement to more traditional optimization methods developed in OR, such as linear and integer programming. In fact, the value of CP as a complementary technology has been recognized by the OR community. Presumably the interactions between OR will continue to flourish.

*. The views expressed in this article are clearly shaped by our work in the area and are not meant to provide a comprehensive overview of the field. We point the reader to the Handbook of Constraint Programming (Rossi et al., 2006) for an overview of the current state of the art of CP and for an extensive bibliography. This research has been supported by the Air Force Office of Scientific Research (AFOSR), the Intelligent Information Systems Institute (IISI), Cornell University, and the National Scientific Foundation (NSF).

We will argue below this is a good time to consider an alternative to the engineering perspective of CP based on a more science-driven perspective. However, before we do so we will list several concrete technical challenges for the field. Note that our challenges and our views expressed here are clearly shaped by our own work in the area.

Beyond discrete domains

Constraint satisfaction involving mainly discrete domains has been the predominant paradigm explored in CP. A key open challenge is the development of an effective approach for constraint satisfaction and optimization that combines both discrete and continuous variables. The best route to combining discrete and continuous quantities appears to be hybrid methods that integrate a variety of algorithmic strategies, such as linear and non-linear programming, satisfiability, global and local search, and probabilistic reasoning. However, in order to provide an effective integration of these methods, it will be necessary to develop ways of exchanging information, such as no-good style explanations, between the various subsolvers.

Effective constraint learning as developed in the satisfiability community

Nogood learning has been studied for a long time, starting with truth maintenance systems, and further developed within the constraint programming community. Nevertheless, nogood learning has not yet been effectively exploited in CP. This is in contrast to the situation in the satisfiability (SAT) community. In order to make nogood learning a practically useful technique, SAT researchers have extended nogood learning (referred to as "clause learning" in the SAT community) with caching, conflict analysis, and restarts. Such techniques are one of the key factors in the rapid advances in SAT solvers. It seems likely that such clause learning methods should be amenable to application in CP solvers.

Exploiting randomization, restarts, and algorithm portfolios for complete or exact solvers

Local search methods exploit randomization to diversify their search. Recently, randomization has also been proven to be effective for backtrack style, complete or exact search algorithms. In particular, state-of-the-art complete SAT solvers exploit randomization and restarts to obtain more robust and consistent performance over a wide range of problem domains. The most recent SAT solvers also employ sophisticated adaptive restart schedules to further optimize performance. Another strategy to diversify search is to combine different solution strategies in so-called algorithm portfolios. We see the beginning of the use of these techniques in CP. For example, Ilog's most recent CP Optimizer includes randomization and restarts for complete search. However, there is much room for further use of these techniques in CP.

Beyond satisfaction and optimization

In order to reach a new range of applications for CP, we will have to go beyond pure satisfaction and optimization, by considering, e.g., quantified constraint reasoning, model counting, probabilistic constraint reasoning, reasoning with preferences, user interaction, reactive and robust reasoning, and automated modeling.

Blackbox constraint-based solvers

The lack of fast blackbox constraint-based solvers is one issue that has somewhat hampered the growth and impact of the constraint-based community. What we mean is the availability of constraint solvers, ideally publicly available open-source solvers, that can be easily used, even by researchers from other communities. Of course, key issues that have to be considered are trade offs between expressiveness and complexity, probably leading to compromises in terms of restricting the constraint languages, not necessarily formally but at least in practice. Again, contrast the situation in CP with the SAT community: over the last ten years the explosion of freely available SAT solvers has been remarkable. These solvers are easy to use and employ a standardized input format based on a clausal normal form. We believe that the fast growth of the SAT community has been in part due to the availability and ease of use of so many fast SAT solvers. This has enabled researchers from other communities, such as the hardware and software verification community, to use such solvers *without needing to develop their own search technology.* The CP community would clearly benefit if similarly efficient CP solvers with a common input language became widely available.

Applications and challenges

It is essential for the advancement of the field that we identify key application areas, with compelling computational challenges, such as in computational biology, AI, and computational economics. The community should formulate a few "killer" challenge problems to further boost the field, the same way that Deep Blue, the proof of the Robbins conjecture, and the DARPA Grand Challenge have boosted AI, Theorem Proving, and Robotics, respectively.

2. The Future of CP: The Science of Constraints

An interesting issue to reflect on is: What are the core scientific questions that the field of CP is pursuing? In other areas, there is a ready answer to this question. For example, in theoretical computer science, the leading open challenge is to determine the fundamental limits of computation. In artificial intelligence, the key issue is to obtain an understanding of what constitutes "intelligence", i.e., how to replicate "intelligent" behavior, and how to build intelligent systems. In machine learning, the challenge is the discovery of the processes behind learning and scientific discovery. Unfortunately, it is less clear what the the broader scientific challenge underlying CP is. We believe this is a good question for the community to contemplate, and we will provide one possible response below.

One position is to view the study of CP as part of algorithm design, and therefore, in a sense, aligned with theoretical computer science. However, from the perspective of theoretical computer science, constraint reasoning does not receive much attention. The reason is simple: CSP problems are generally NP-complete. This means that interesting solution methods are generally based on heuristic search strategies, for which there is little hope of obtaining interesting complexity bounds, beyond the worst-case exponential bounds. However, a worst-case analysis seems overly pessimistic. In fact, a key underlying assumption behind constraint reasoning approaches is the believe that worst-case exponential complexity can be avoided in many cases of practical interest because of special domain structure. Clever propagation techniques based on local or global information have been developed to exploit such problem structure, but the structure is generally not sufficiently well-behaved or understood to develop a CSP formulation that fits any known syntactic tractable class.

These considerations lead us to another perspective that has not been traditionally pursued within computer science: *To view constraints structures as natural phenomena and therefore constraint reasoning as part of the natural sciences.* In this context, even though almost all interesting constraint reasoning tasks are worst-case NP-complete (or worse), real-world structure may allow for practically effective and scalable solution strategies. However, to understand such underlying structure, one has to study the constraint reasoning problem as a *natural phenomenon, and combine principled experimentation with formal modeling.* We believe that such a perspective on CP leads to a fundamentally new way of studying and solving constraint reasoning problems. In particular, in our view the central question for the field is the understanding of constraint structures as occurring in the real-world — not just as abstract formal or mathematical objects. Of course, formal models and mathematical characterizations will still be useful in modeling and understanding the constraints, but such models will, in a sense be, "in service" of our domain of study — real-world constraint structures — and not the prime object of study themselves. We believe that such a perspective will add significantly to the already vibrant CP research effort. We arrived at this perspective mainly through reflection on our own research over the years. In fact, our work has generally been driven by a desire to understand interesting empirical phenomena in constraint reasoning through the development of formal models and techniques. We will briefly highlight three examples of this work.

Phase transition in combinatorial problems

Phase transition phenomena with the associated easy-hard-easy pattern in combinatorial problems were first empirically observed in the early nineties. The empirical data was so compelling that it was clear that the phenomena were real, even though there was relatively little formal understanding. In fact, the first formal proof of the existence of a phase transition in k-SAT was only obtained in 1997 by Friedgut. The exact location of the k-SAT phase transition has still not been obtained, but good upper and lower bounds are now known. The connection between phase transitions, i.e., a sudden change from satisfiable to unsatisfiable instances, and a peak in search complexity is still largely based on empirical observations, even though there has been substantial progress in our general understanding of the complexity of search on random problem ensembles. A major advance in our understanding of phase transition phenomena and search complexity came when it was established that models from statistical physics for understanding phase transitions in physical systems, in particular spin glasses, could also be used to model phase transitions and complexity in computational systems. This research area has now blossomed into an active collaboration of physicists, computer scientists, and mathematicians, who are building ever more sophisticated mathematical models and pursuing increasingly complex analytical tools for studying phase transitions and search complexity in computational systems. It is important however to keep in mind that this research effort fundamentally arose out of careful empirical observations, *which first established phase transitions as a concrete natural phenomenon, worthy of study.* Moreover, the fact that phase transition phenomena are linked to constraint problems becoming “critically constrained,” i.e., a point at which resources and demands are carefully balanced, provides another connection to real-world domains. It seems quite likely that phase transitions and complexity would not have gathered as much attention if these phenomena were first established in a purely formal manner, with empirical results being only an afterthought.

Heavy-tailed phenomena in combinatorial search

Another example of a research effort that was driven first by empirical observations is our work on *heavy-tailed* phenomena in the runtime of backtrack style search procedures. We were led to these phenomena in trying to characterize the highly erratic behavior of the runtime distributions of backtrack-style search on real-world problem domains. In order to further understand heavy-tailed phenomena in combinatorial search, we also developed formal models explaining the occurrence of heavy-tails in backtrack search as well as models of “statistical regimes of heavy-tailed behavior” in combinatorial domains, i.e., a general characterization of parameter regions where heavy-tailed behavior is prevalent. This work draws on early work on coding theory by Berlekamp, in particular on decoding of sequential convolutional codes, where similar heavy-tailed phenomena had been observed. A practical payoff of this work was the observation that randomization and restarts can significantly enhance complete, backtrack style search procedures, to combat heavy-tailed behavior. In fact, randomization and restarts have become an integral part of current state-of-the-art SAT solvers and are now also being integrated into commercial CP solvers. Restarting is combined with learning, further boosting its power. Again, this line of research provides an example of how an initially empirically observed phenomenon can drive the development of better formal models to enhance our understanding of practical constraint satisfaction tasks. It’s unlikely that purely formal work could have led us to these insights.

Impact of problem sub-structure on problem hardness and backdoor variables

In order to better understand the complexity of real-world problems, we studied the impact of (hidden) tractable sub-structure on search procedures. In particular, we considered the notion of so-called *backdoor variables.* These variables capture the combinatorics of the underlying problem with respect to a given polytime solver: Backdoor variables are a special subset of the problem variables, such that, when they are assigned values, the remaining problem instance can be solved by a polytime algorithm, i.e., the problem instances reduces to a tractable class (not necessarily syntactically defined). We first introduced the notion of backdoor variables as a formal notion to model heavy-tailed behavior of complete randomized backtrack search methods. As mentioned above, heavy-tailed distributions are characterized by a wide range of runtime values. Backdoors provide a justification for short runs in backtrack search, explaining why current state-of-the art constraint solvers often seem to defy the worst-case theoretical results by finding satisfying solutions (or proving unsatisfiability) for instances with many thousands of variables and constraints. It is exciting to see that small backdoor sets do occur in real-world domains such as in planning, software and hardware verification. Different formal backdoor models based on restrictions of the target tractable class have been proposed. Further study may lead to a better understanding how small backdoor sets arise naturally in many real-world problem encodings.

3. Conclusions

In summary, we argue that CP would benefit from pursuing a science driven research agenda, in addition to an engineering approach. In such an approach, constraint structures are viewed as natural phenomena, instead of abstract mathematical structures. The study of the properties of constraint structures will require the scientific methodology from the natural sciences, in which empirical observations play as prominent a role as formal models and analysis. In fact, a close interaction between empirical studies and formal analysis will be key. The work on phase transitions, heavy-tailed phenomena, and backdoor variable sets shows that such a perspective on CP research can be quite fruitful in obtaining deeper insights into the process of constraint reasoning. Ultimately, such an endeavor may lead us to a rich Science of Constraints.

References

Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming.* Elsevier, 2006.