Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Explaining Ourselves: Human-Aware Constraint Reasoning

Eugene C. Freuder

Insight Centre for Data Analytics, School of Computer Science and IT, University College Cork, Ireland
eugene.freuder@insight-centre.org

Abstract

Human-aware AI is increasingly important as AI becomes more powerful and ubiquitous. A good foundation for human-awareness should enable ourselves and our "AIs" to "explain ourselves" naturally to each other. Constraint reasoning offers particular opportunities and challenges in this regard. This paper takes note of the history of work in this area and encourages increased attention, laying out a rough research agenda.

Explaining Ourselves: Me and My AI

Human-aware AI (the IJCAI-16 Special Theme) is increasingly important as AI becomes more powerful and ubiquitous. A good foundation for human-awareness should enable ourselves and our "AIs" to "explain ourselves" naturally to each other. Constraint reasoning offers particular opportunities and challenges in this regard. This paper takes note of the history of work in this area and encourages increased attention, outlining a rough research agenda.

Constraint reasoning (or "constraint programming") involves finding values for problem variables that satisfy constraints on which combinations of values are allowed. The constraints may be "soft", representing preferences or probabilities or costs, introducing an element of optimization. Constraint reasoning has been widely used in AI, from vision to planning, and a great many practical problems, from factory scheduling to molecular biology, can be modeled and solved as constraint satisfaction problems (CSPs) (Freuder 2006).

Constraints are a natural, and as opposed to rules, a declarative way of expressing our needs. We encounter constraints in everyday interactions with computers, whether or not there is specifically constraint programming technology operating behind the scenes. We are able to convey simple constraints to our phones in plain English: "I want a restaurant that has vegetarian options." We express constraints that "filter" our purchasing options, e.g. for flights by checking boxes for which airlines to use or moving sliders to constrain arrival times.

On the one hand, we are increasingly being led to expect that we can simply explain to "intelligent assistants", like Siri or Alexa, what our needs or preferences are, and these programs can then decide how best to satisfy them. On the other hand, by 2018 the European Union General Data Protection Regulation may give citizens the right to demand explanations of decisions that computer algorithms make about them (Goodman and Flaxmanar 2016).

Of course, the constraint programming community has been addressing related issues for some time, but it has not been a primary focus, and now is a good time for a renewed emphasis on "user-friendly I/O".

Twenty years ago, in a paper entitled "In Pursuit of the Holy Grail" (Freuder 1997), I laid out a strategic goal for constraint programming where the user simply states the problem and the computer solves it. Fifteen years ago, in an invited talk for the International Conference on Principles and Practice of Constraint Programming (CP), I borrowed James Carville's famous line about a U.S. presidential election "It's the economy stupid" to claim "It's usability stupid", and Barry O'Sullivan and I organized the First International Workshop on User-Interaction in Constraint Satisfaction. Ten years ago, in a paper entitled "Holy Grail Redux" (Freuder 2007), I spoke of the exciting opportunity for members of the constraint programming community to be pioneers of a new "usability science" and to go on to "engineer usability". Of course, I am hardly the only one to have recognized this challenge. For example, Jean-Francois Puget, now an IBM Distinguished Engineer, gave an invited talk at CP 2004 entitled "The next challenge for CP: Ease of use" (Puget 2004).

However, as one crude example of the relative attention to these issues, a search of over two decades of papers in the CP conferences reveals an approximate average of only one paper every two years with "explanation(s)" in the title. So I expect there even remains some "low hanging fruit" to be picked. In the following I lay out some "dimensions of the orchard". I also cite a few examples of "al-

ready picked fruit" for illustration, but this short paper should by no means be considered a survey.

I divide the discussion into "Me" and "My AI". The former is usually treated as "problem acquisition" or "modelling", the latter as "explanation', but here I view them both as forms of explanation: users explaining their problems and computers explaining their success or failure at solving those problems.

Also, while I treat these separately and sequentially, it is important to note that ideally these can operate in a collaborative cycle: a user provides a problem, or an initial portion of a problem specification, a solver provides a response, based on this response the user alters the problem to seek a different or more satisfactory result, or augments/refines the problem specification, and around we go until the user is satisfied. This unified view of a user/computer dialogue is particularly relevant to the current interest in such themes as Human-Aware AI, Human-Computer Collaboration, and Intelligent Personal Assistants.

Me

For general constraint programming, high level languages aim to simplify the problem modeling process (Frisch et al. 2008, Marriott et al. 2008). For industrial applications in specific application domains, specialized programs may facilitate the entry of constraints.

Expressing constraints is naturally embedded in the familiar spreadsheet environment (Sample and Mouhoub 2011). (Hammond and O'Sullivan 2007) describe work on processing problems presented as hand-drawn "constraint networks". (Kiziltan et al. 2016) have recently made progress in extracting constraints from natural language.

One problem with letting a naïve user describe a CSP is that the resulting model may be prohibitively inefficient to solve. However, considerable progress has been made on automating the reformulation of models into more 'solver-friendly' form (Nightingale et al. 2014).

Nevertheless, there is still a considerable way to go before the computer can fully replace the human personal assistant on the one hand and the professional knowledge engineer on the other. This challenge overlaps with the general challenge of natural language processing. However, users may have difficulty even articulating their constraints in natural language. As problems scale, we need to find methods to keep the interaction required manageable for the user.

A back and forth dialogue may better simulate the interaction with a human assistant or consultant programmer. (Freuder and Wallace 2002) studied a rudimentary "matchmaker" system that simulated the kind of interaction one might have, for example, with a salesperson. The computer makes a suggestion (a solution to its current CSP model of the user's problem), the user says 'no, because' providing an additional constraint that the system uses to refine its understanding of the user's needs (update its CSP model), then the system makes another suggestion (a solution to the updated model), and user and system continue this dialogue until the user is satisfied.

Machine learning may reduce the burden on the user. Acquiring constraints from examples has a long history, e.g. "Boltzmann Machines: Constraint Satisfaction Networks that Learn" (Hinton, Sejnowski, and Ackley 1984) and "Constraint Acquisition" (Bessiere et al. in press). (Beldiceanu and Simonis 2012) describes a system that generates constraint models from example solutions, using "global constraints" from a global constraint catalog as primitives from which the models are created. (Shchekotykhin and Friedrich 2009) allows users to provide "arguments" that can reduce the number of examples that they need to classify.

Articulating one's constraints becomes an even more challenging issue when soft constraints are involved. (Rossi and Sperduti 2004) describes a system that learns preferences over constraints from preferences over solutions.

A further complication arises when several users are working together to find a mutually acceptable solution, and this can be complicated even more when the users are reluctant to reveal their constraints because of privacy issues. (Gelain et al. 2010) interleaves search and preference elicitation, seeking to ask the user to reveal as few preferences as possible.

Of course, one's needs and preferences may change, so maintaining an up-to-date understanding of and by the user also presents a challenge (Nordlander, Freuder, and Wallace 2007). Fortunately, there is a body of work on "dynamic constraint satisfaction" that may mitigate the computational issues that can be especially challenging in a fast changing real-time environment.

My AI

Explaining a successful solution for a constraint satisfaction problem may seem straightforward, which is probably one reason why there has been little attention paid to doing so. One can simply show that the solution does indeed satisfy each constraint. However, there may be occasions when more information is sought by the user. The user may be unsatisfied with the provided solution, because the problem was incompletely, incorrectly or inadequately specified. When there are multiple possible solutions the user may want to know why one was chosen, especially in the context of soft constraints, where indeed the user may want some assurance that a solution is optimal. Users may wish assistance in exploring the space of possible solu-

tions, including those introduced by alternative choices for the users' constraints. One of the difficulties users may have with constraint-based solutions is that a small change in the formulation of a problem may lead to a large change in the difficulty of the problem (or conceivably make it unsolvable) and they do not know why. Users might also want to see how the computer solved the problem as a way of learning how to solve such problems themselves.

A natural approach to providing a richer explanation of a solution would be to 'trace' the program's solution process. However, constraint solvers generally employ search, and tracing search tends not to provide a very satisfying explanation. For backtrack search: "I tried this and then that and hit a dead end, so I tried the other instead". Even worse, for local search: "I kept getting better, but then I tried some other random thing". However, constraint solving also employs inference. (Sqalli and Freuder 1996) provided explanations for logic puzzles by providing additional opportunities for inference so the puzzles could be solved entirely by inference, without search. A subsequent trace of the inference, with some rudimentary natural language processing, provided explanations for puzzles taken from newsstand puzzle booklets that were reasonably similar to the answer explanations provided in the back of the booklets.

Much of the work on explanation for constraint satisfaction, however, has dealt with situations in which the problem as stated is unsolvable. The user then needs assistance in weakening or altering the problem specification to permit solution. (Amilhastre, Fargier, and Marquis 2002) studies this problem for soft constraints, in particular for interactive configuration problems. They wanted to be able to answer user questions like:

- "Which choices should I relax in order to recover consistency?"
- "Which choices should I relax in order to render such a value available for such a variable?"
- "From which subsets of current choices did inconsistency follow?"
- "Why is this value not available any longer for this variable?"

In (Jussien and Barichard 2000) we encounter questions like:

- "How come I do not get value x for variable v?"
- "How come that problem has no solution?"
- "How come y is the only remaining possible value for variable v?"

(Freuder, Likitvivatanavong, and Wallace 2001) uses "explanation trees" that show the basis for assignments and deletions in terms of previous selections, whether leading to success or failure. The intent is to provide explanations that help the user understand the following situations:

- "Why did we get this as a solution?"
- "Why did this choice of labels lead to a conflict?"
- "Why was this value chosen for this variable during processing?"

and also to help users in an interactive problem-solving setting understand the implications of their choices by providing help with these kinds of questions:

- "Is there a basis for choosing among values in a future domain?"
- "Are there values whose choice will lead to conflict, even though they are consistent with the present domains?"

(Junker 2004) describes work that provided the technological basis for the explanation facility in commercial products produced by ILOG, now part of IBM. (Liffiton et al. 2016) describes recent progress in computing "minimal unsatisfiable subsets", which provide a kind of minimal explanation of infeasability. (O'Sullivan et al. 2007) seeks to assist users by identifying subsets of explanations that are "representative" of all possibilities. (Wallace and Freuder 2001) explores general issues such as what makes for a "good" explanation.

Much of the work on explaining failure actually is focused on programs explaining intermediate failures to themselves in order to reach a solution more efficiently. There has also been work on providing explanations of program operation to programmers for development or debugging. Some of this work might be repurposed to provide user explanations.

Challenges and Opportunities

The efforts cited here, and others like them, represent encouraging progress. At the same time, the issues they address stand as continuing, and ever more timely, challenges and opportunities for increasing mutual understanding of user and machine for Human-Aware Constraint Reasoning.

A research roadmap might include the following signposts:

- Multiple interaction modalities: natural language, examples, analogies, stories, histories, …
- Efficient interaction: reducing user effort
- Multiple users: cooperative and adversarial
- Maintainability: as needs and contexts change
- Transparency and privacy: "need to know"

- Scaling up to large-scale problems
- Extracting constraints and CSP models from "big data" and the web and using "deep learning"
- Learning and improving models by observing or interacting with human domain experts
- Learning from experience with users
- Measuring confidence in, quality of, and completeness of acquired models
- User interfaces for interactive problem solving, especially for real-time, dynamic problems
- Explaining the implications of choices or changes in problem specification
- Providing appropriate insight and guidance for human-machine collaboration
- Acquiring, improving and utilizing "background knowledge" of user and problem domain constraints to facilitate future problem solving.
- Incorporating "outside" constraints, legal, regulatory, contractual, ethical
- Specialized methods for different classes of constraints
- Specialized tools for specific application domains
- Specialized methods for answering specific user questions
- Distinguishing and evaluating varieties of explanation
- Measuring and improving explanation quality

Such a research program can draw on, as well as contribute to, the work of many fields of AI that bear upon human-machine communication, including machine learning, recommender systems, knowledge acquisition, intelligent user interfaces, and natural language understanding.

Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

References

Amilhastre, J., Fargier, H., Marquis, P. 2002. Consistency restoration and explanations in dynamic CSPs—application to configuration, *Artificial Intelligence* 135: 199–234.

Beldiceanu, N., Simonis, H. 2012. A Model Seeker: Extracting Global Constraint Models from Positive Examples. In *Proceedings of the Eighteenth International Conference on Principles and Practice of Constraint Programming*, 141-157. Springer LNCS 7514.

Bessiere, C., Koriche, F., Lazaar, N., O'Sullivan, B. in press. Constraint acquisition. *Artificial Intelligence*.

Freuder, E. 1997. In Pursuit of the Holy Grail. *Constraints* 2(1): 57-61.

Freuder, E. 2006. Constraints: The Ties that Bind. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence*, 1520-1523. Menlo Park, Calif.: AAAI Press.

Freuder, E. 2007. Holy Grail Redux. *Constraint Programming Letters* 1: 3-5.

Freuder, E., Likitvivatanavong, C., Wallace, R. 2001. Deriving Explanations and Implications for Constraint Satisfaction Problems. In *Proceedings of the Seventh International Conference on Principles and Practice of Constraint Programming*, 585-589. Springer LNCS 2239.

Freuder, E., Wallace, R. 2002. Suggestion Strategies for Constraint-Based Matchmaker Agents. *International Journal on Artificial Intelligence Tools* 11(1): 3-18.

Frisch, A., Harvey, W., Jefferson, C., Hernández, B., Miguel, I. 2008. Essence: A constraint language for specifying combinatorial problems. *Constraints* 13(3): 268-306.

Gelain, M., Pini, M., Rossi, F., Venable, K., Walsh, T. 2010. Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. *Artificial Intelligence* 174(3–4): 270-294.

Goodman, B., Flaxmanar, S. 2016. European Union regulations on algorithmic decision-making and a "right to explanation". Xiv:1606.08813v3 [stat.ML].

Hammond, T., O'Sullivan, B. 2007. Recognizing Free-form Hand-sketched Constraint Network Diagrams by Combining Geometry and Context. In Proceedings of Eurographics Ireland 2007, 67-74.

Hinton, G., Sejnowski, T., and Ackley, D. 1984. Boltzmann Machines: Constraint Satisfaction Networks that Learn, Technical Report, CMU-CS-84-119, Carnegie Mellon University, Pittsburgh, PA.

Junker, U. 2004. QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 167-172. Menlo Park, Calif.: AAAI Press.

Jussien, N., Barichard, V. 2000. The PaLM system: explanation-based constraint programming. In Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a workshop of CP 2002, 118–133, Technical Report TRA9/02, School of Computing, National University of Singapore, Singapore.

Liffiton, M., Previti, A., Malik, A., Marques-Silva J. 2016. Fast, flexible MUS enumeration. *Constraints* 21(2): 223-250.

Marriott, K., Nethercote, N., Rafeh, R., Stuckey, P., de la Banda, M., Wallace, M. 2008. The Design of the Zinc Modelling Language. *Constraints* 13(3): 229-267.

Nightingale, P., Akgün, Ö, Gent, I., Jefferson, C., Miguel, I. 2014. Automatically Improving Constraint Models in Savile Row through Associative-Commutative Common Subexpression Elimination. In *Proceedings of the Twentieth International Conference on Principles and Practice of Constraint Programming*, 590-605. Springer LNCS 8656.

Nordlander, T., Freuder, E., Wallace, R. 2007. Maintaining constraint-based applications. In Proceedings of the 4th International Conference on Knowledge Capture, 79-86. New York, NY: ACM.

O'Sullivan, B., Papadopoulos, A., Faltings, B., Pu, P. 2004. Representative Explanations for Over-Constrained Problems. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, 323-328. Menlo Park, Calif.: AAAI Press.

Puget, J.-F. 2004. Constraint programming next challenge: Simplicity of use. In *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming*, 5–8. Springer LNCS 3258.

Rossi, F., Sperduti, A. 2004. Acquiring Both Constraint and Solution Preferences in Interactive Constraint Systems. *Constraints* 9(4): 311-332.

Sample, T., Mouhoub, M. 2011. Augmenting spreadsheets with constraint satisfaction. In Proceedings of the 2011 24th Canadian Conference on Electrical and Computer Engineering(CCECE), 1028-1031. New York, NY: IEEE.

Shchekotykhin, K., Friedrich, G. 2009. Argumentation Based Constraint Acquisition. In Proceedings of the Ninth IEEE International Conference on Data Mining, 476-482. New York, NY: IEEE.

Sqalli, M., Freuder, E. 1996. Inference-based constraint satisfaction supports explanation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 318-325. Menlo Park, Calif.: AAAI Press.

Wallace, R., Freuder, E. 2001. Explanations for Whom?. In Working Notes of the First International Workshop on User-Interaction in Constraint Satisfaction, 119-130.