Principles of Constraint Programming *By Krzysztof R. Apt.* Cambridge University Press, Cambridge, 2003. GBP 35.00. xii + 407 pp., hardcover. ISBN 0-521-82583-0.
Constraint Processing *By Rina Dechter.* Morgan Kaufmann Publishers, 2003. $ 65.95. xx + 481 pp., hardcover. ISBN 1-55860-890-7.

Of the most recent disciplines to arise in computer science, one is devoted to modelling with constraints and to solving the resulting constraint satisfaction problems. Some authors call it "constraint programming", which I will avoid to prevent confusion with "programming" as used in, say, "linear programming" or "object-oriented programming". Other authors prefer "constraint processing", which I will avoid in favour of the more specific *constraint satisfaction*.

Constraint satisfaction started as part of artificial intelligence. In 1993 annual workshops started, which became the CP (for "Constraint Processing") series of conferences. Since 1996 Kluwer has published a journal named *Constraints* exclusively devoted to the subject. The blurb for the journal lists as application domains artificial intelligence, discrete mathematics, neural networks, operations research, design and configuration, graphics, visualization and interfaces, hardware verification and software engineering, molecular biology, scheduling, planning, resource allocation. Though it sounds like a blurb, it is a justified claim.

Early books on the subject [7, 8] were focused on constraint satisfaction as a basis for a programming language. Tsang's [13] is an exception: it is a pioneering precursor to the books under review. It is fitting that they were published shortly after [13] went out of print.

Considering how constraint satisfaction arose in artificial intelligence, which is the conventionally accepted misnomer of what is, in effect, the interdisciplinary and experimental branch of computing, one might consider it to be marginal to the interests of *SIAM Review*. As I hope convince readers of this review, this is marginality of the beneficial kind. Accordingly, I include some background, showing how constraint programming originated in numerical computation, diverged from its origins, and seems to be settling down in a way that makes its relevance to optimization easy to overlook.

Logically, optimality and feasibility are equally important in constrained optimization. But to get anything done, one needs to concentrate on one or the other. In optimization, optimality is put first. Constraints are incorporated in the objective function, whether by Lagrange multipliers or by the use of penalty functions.

The opposite point of view regards the constraints as primary. As an exercise to learn to see the world from the point of view of constraint satisfaction rather than of optimization, let us consider an underdetermined system of equations. If the system were linear and algebraic, then it is clear how to obtain a neat characterization of the set of solutions. In the presence of transcendental functions no such characterization is available; one should be content with a single solution. Which? If we select this single solution by a preference function, then we have arrived at a constrained optimization problem with the preference function as objective function. And of course, even with an underdetermined system of linear algebraic equations one may prefer to select a single solution by means of a preference function. If this function is linear, then we have arrived at a linear programming problem.

The two books under review do not seem to be connected to the numerical world of optimization. To make such a connection, one has to consider a constraint satisfaction setting where the variables do not necessarily range over the reals and where the relations in the constraints are not necessarily equality or inequality. Typically, variables range over small finite sets; often variables are boolean. The relations are often not binary. For example, in the modelling of digital circuits, gates can be modelled by relations between the boolean values on the terminals.

For the inverter this relation is binary, but that is an exception. Dechter's book is completely combinatorial, emphasizing search and propagation techniques. Though Apt does not exclude real valued variables, and has a section on arithmetic constraints on reals, this book is also far from the world of optimization.

Yet the history of constraint satisfaction is rooted in the classical notion of constraint. The oldest scientific use of the term "constraint" is in the sense of "constrained motion". Such motion presents a difficulty in Newtonian dynamics, which is satisfactory for the analysis of dynamical systems such as that of the Moon orbiting Earth. New techniques were needed to describe the motion of systems like a pendulum. If the bob were to follow the force of gravity only, it would go straight down. However, its motion is constrained by the distance to the hinge being constant.

The new technique, analytical dynamics, appeared a century after Newton's Principia. The principle of d'Alembert, which was first stated in its full generality by Lagrange, recognizes two kinds of forces: "impressed forces" and "constraint forces". The principle uses the fact that the latter do no work under virtual displacements. As pointed out by Lagrange, the principle provides a much-needed simplification, which enables easy description of the accelerations of the constrained system.

Though the books under review are far removed also from the world of analytical dynamics, it seems likely that the word "constraint" in the sense of constraint satisfaction was borrowed from analytical dynamics by the engineers at MIT who authored the theses that can be regarded as the origin of constraint satisfaction. The first of these was Ivan Sutherland. In his 1963 thesis [12] he defines "constraint" as

> A specific storage representation [in computer memory] of a relationship between variables which limits the freedom of the variables, i.e., reduces the number of degrees of freedom of the system.

The subject of Sutherland's thesis is Sketchpad, an interactive drawing program. It allows the user to create, replicate, and modify geometric objects. The relations between these objects give rise to sizable systems of equations. Their provenance caused Sutherland to think of them as constraints in the sense of analytical dynamics.

The two salient features of the system of equations generated by the program were (a) that many equations were not linear, and (b) the system was sparse. Sketchpad employed two methods for solving it, both exploiting the sparseness of the equations. The first method used a graph representing dependencies among the variables. In favourable cases, variables were found whose values did not depend on those of any other variables. Their values were then "propagated" through the graph. For the propagation algorithm Sutherland acknowledged inspiration provided by Moore's shortest path algorithm [9].

What to do if the system of equations does not unravel in this way? It is unlikely that Newton's method would have worked. Instead, Sutherland applied to his nonlinear systems the Gauss-Seidel method that had been popularized in engineering by R.V. Southwell under the term "relaxation" [11]. After all, linearity is not essential for relaxation to work: it is just one way that enables one to isolate one variable in each equation and to ensure that each of the variables is isolated in at least one of the equations.

In typical applications of relaxation, the system of equations is sparse. But the typical system is linear, and the usual way of dealing with sparseness is to order the equations and the variables in such a way that the coefficient matrix gets an advantageous block or band structure. In the case of nonlinear systems, as in Sutherland's case, the matrix would be the Jacobian.

The other way of dealing with sparseness is to consider what could be called the "dependency network", where the nodes are variables and where nodes are linked if the corresponding variables occur in the same equation. Instead of first determining the Jacobian and attempting to transform it into a favorable block or band-structured form, one can avoid both steps and let the solving process automatically determine what its structure is: note which of the variables have changed sufficiently in the previous step and pick for the next equation one of the few that contains at least one of these variables. In this way, change in variable values *propagates* by means of constraints. Refinements of the idea, under the name of "constraint propagation", have become an important part of research in constraint satisfaction. It is data-directed control of the relaxation algorithm.

To conclude that Sutherland used constraint propagation, one has to read between the lines. I think it likely that he did. If so, he is the first that I know of. Whoever was the first started a rich new research area that is central to constraint satisfaction. Thus we see in Sutherland's 1963 thesis, which was focused on the development of a practical drawing tool, the emergence of two themes that recurred throughout the new discipline of constraint satisfaction: constraint propagation and relaxation.

However, the setting of Sketchpad, real variables and equality as only constraint relation, is not representative of what constraint programming came to be. It is typical for variables to range over finite sets, often very small ones, like the two truth values or the colours of a graph-colouring problem. It is also typical for the constraint relations to be taken from a large assortment that includes some recently invented ones, such as the seven relations between time intervals [1]. An important relation on small finite sets is the disequality relation (a name chosen to distinguish it from $\leq$ and $\geq$ among numbers). Especially important is the version of the disequality relation that takes an arbitrary number of arguments.

An important step away from numerical constraint satisfaction problems was taken in the 1972 MIT thesis by David Waltz [15, 14]. It was concerned with computer recognition of three-dimensional polyhedral bodies in two-dimensional images. This requires the lines in the image to be labelled according to whether they arise from a shadow, from one body obscuring another, from the intersection of two faces of the same body, among other possibilities.

In Waltz's thesis, variables range over small, application-specific finite sets. In addition, his relations are also application-specific, in this case arising from the various ways in which lines in the image can intersect. This is about as different from equations in real-valued variables as one can get. Yet, as a constraint satisfaction problem, Waltz's had in common with Sutherland's that it was sparse: most constraints related but a small subset of the variables. That again suggested data-directed control.

Waltz's algorithm contains another important innovation. In conventional relaxation, one always associates one value with each variable, even though these values may not be anywhere near their solution values. That is, one starts with a guess. Each step in the relaxation changes (for the better, as one hopes) the guess for one variable. In favourable cases, these guesses approach a solution.

Waltz did not guess; he knew what he did not know. Accordingly, he associated with every variable a "domain": the set of its possible values. Using a constraint in x and y could at best mean that the non-occurrence of certain values in the domain for x allowed the removal of one or more values from the domain of y. If y occurs in another constraint, say, with a variable z, then this reduction of its domain may trigger a reduction of the domain of z. This propagation process is often referred to as "relaxation", even though it is far removed from the setting of equations in real variables.

A typical example of reasoning à la Waltz is the situation where all of the variables x, y,

and z have to be different. If the sets of values associated with these variables are $\{a, b, c\}$, $\{b, c\}$, and $\{b, c\}$ respectively, then one can eliminate b and c as possible values for x. In the terminology of the field, these values can be eliminated as being *inconsistent*; inconsistent with the constraint that the three variables be different.

If a constraint satisfaction problem has one solution, then all one has to do to find that solution is to allow initially all values as possible for all variables and then to remove all inconsistent values. A typical constraint satisfaction problem is NP-hard, so this is easier said than done. The field has advanced by identifying various feasibly computable levels of consistency short of complete absence of inconsistent values. These are identified by qualifying the word "consistency" in various ways. A bewildering variety of such terms are used. Apt's chapter "Local Consistency Notions" lists nine varieties of local consistency, followed by a final section "Graphs and Consistency". In Dechter this last section is expanded into an entire chapter. It represents the culmination, so far, of Sutherland's new way of approaching sparseness by exploiting the structure of the dependency graph of the constraints.

Next to consistency, the propagation introduced by Waltz is an important topic in constraint satisfaction. Both Apt and Dechter devote a chapter to it. Apt's is notable for his surprisingly general framework, going far beyond the confines of constraint satisfaction. He starts with properties of sets of functions on partially ordered sets that have one or more of the properties of being inflationary, monotonic, idempotent, and mutually commutative. He gives a compelling development of fixpoint algorithms that iterate such functions. One would expect results such as these to be included in set theory texts, were it not for the fact that they were only recently obtained (by Apt) in order to better understand the wide variety of ad-hoc propagation algorithms found in the literature.

Next to consistency and propagation, the main topic of constraint satisfaction is *search*. In any nontrivial constraint satisfaction problem, propagation to some level of consistency is not enough to obtain a solution. When propagation has exhausted all possibilities of removing inconsistent values from the domains, further progress depends on splitting and repeating propagation in the resulting constraint satisfaction problems. The properties of the resulting tree of constraint satisfaction problems, and the algorithms for traversing it, constitute the topic of search. Apt has an excellent one-chapter introduction; Dechter digs deeper with three chapters: Look-Ahead, Look-Back, and Stochastic Search (which includes simulated annealing).

We have seen how constraint satisfaction arose with Sutherland's work on numerical problems and how Waltz adopted Sutherland's propagation for his combinatorial problem in scene recognition. In the combinatorial setting constraint satisfaction developed into a new discipline. As a sign of its maturity, it started competing with Integer Programming as the method of choice for solving combinatorial problems. Take for example the notorious 10-machine/10-job benchmark `mt10` in job-shop scheduling. It was proposed by Muth and Thompson [10] in 1963. For some time various researchers succeeded in finding ever better solutions. These improvements came to a halt after about ten years. For another ten years failure to improve the world record on `mt10` hardened the suspicion that the optimum had been found. At the end of this period, Carlier and Pinson [2] succeeded in reducing the search space sufficiently to prove that the long-standing record is indeed the optimum.

All this work was done in integer programming: it consisted of improvements in search, in generating cutting planes, and, of course, it benefited from increased computer performance. One sign of the coming of age of constraint satisfaction in the 1990s was that `mt10` was more and more routinely solved by constraint satisfaction methods, without reliance on integer programming.

Although in this instance a purely combinatorial constraint satisfaction attack was successful, it is likely that there is great potential in developing constraint satisfaction for numerical

problems. That is, to exploit the possibility that the variables range over the reals and that the domains take the form of intervals bounded by floating-point numbers. So far this approach to numerical computation has remained outside of the main stream of constraint satisfaction. It is not mentioned in Dechter. In Apt there is a brief section on arithmetic constraints between real-valued variables. For this neglected area, the best source is still "Numerica" by Van Hentenryck, Michel and Deville [5], where constraint satisfaction can be seen to be a competitive alternative to the continuation method for solving nonlinear equations. The methods in this book are applicable to systems of nonlinear inequalities as well and also to nonconvex global optimization.

Constraint satisfaction as understood by Apt and by Dechter seems to have no connection with optimization as understood by the applied mathematics community. Yet it is highly relevant. It is the great merit of Hooker's "Logic-Based Methods for Optimization" [6] to make the connection. This is not obvious from title because of the emphasis on logic. Indeed, Hooker's motivation is to break out of the classical optimization framework by the use of logic as a more flexible modelling tool. In this Hooker was preceded by Van Hentenryck who noticed that languages like AMPL and GAMS lacked the scope for the enlarged modelling capabilities of constraint satisfaction. This led him to the language OPL [4], now used in some of the ILOG software. Early implementations of constraint satisfaction used logic as programming framework [3, 8]. This connection led Hooker to include an excellent account of constraint satisfaction in [6].

For those in the optimization field to benefit from the developments in constraint satisfaction, my summary recommendation is to study both books under review first, to obtain the necessary distance from the traditional optimization mind set. There is surprisingly little overlap between the two books. Apt emphasizes the roots in logic; Dechter the algorithmic complexity aspects. It is best to start with Apt, for the larger picture and for an introduction to the topics where they overlap. Then Numerica [5] for a glimpse of the potential of constraint satisfaction in classical numerical computation. Then back to optimization with Hooker [6].

References

[1] J. Allen. Maintaining knowledge about temporal intervals. *Comm. ACM*, 26:832 – 843, 1983.

[2] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176, 1989.

[3] Pascal Van Hentenryck. *Constraint Satisfaction in Logic Programming.* MIT Press, 1989.

[4] Pascal Van Hentenryck. *The OPL optimization programming language.* MIT Press, 1999.

[5] Pascal Van Hentenryck, Laurent Michel, and Yves Deville. *Numerica: A Modeling Language for Global Optimization.* MIT Press, 1997.

[6] J. Hooker. *Logic-Based Methods for Optimization - Combining Optimization and Constraint Satisfaction.* Wiley-Interscience series in discrete mathematics and optimization. John Wiley and Sons, 2000.

[7] William Leler. *Constraint Programming Languages: Their Specification and Generation.* Addison-Wesley, 1988.

[8] Kim Marriott and Peter J. Stuckey. *Programming With Constraints : An Introduction*. The MIT Press, 1998.

[9] E.F. Moore. On the shortest path through a maze. In *International Symposium on the Theory of Switching*, 1959.

[10] J.F. Muth and G.L. Thompson. *Industrial Scheduling*. Prentice Hall, 1963.

[11] R.V. Southwell. *Relaxation Methods in Engineering*. Oxford University Press, 1940.

[12] I. Sutherland. *Sketchpad: a Man-Machine Graphical Communication System*. PhD thesis, Dept. of Electrical Engineering, MIT, 1963.

[13] Edward Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[14] D. Waltz. Understanding line drawings in scenes with shadows. In Patrick Henry Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, 1975.

[15] David L. Waltz. *Generating Semantic Descriptions From Drawings of Scenes With Shadows*. Technical report AITR-271, Computer Science and Artificial Intelligence Laboratory, 1972.

Maarten van Emden
University of Victoria