

Constraint-Based Programming for Scheduling: An Historical Perspective¹

Claude Le Pape

ILOG S.A.

2 Avenue Galliéni - B.P. 85 - 94253 Gentilly Cedex - France

Constraint-based programming is a problem-solving method which establishes a neat distinction between:

- on the one hand, a precise definition of the constraints that define the problem to be solved;
- on the other hand, the algorithms and heuristics enabling the selection, ordering and cancellation of decisions to solve the problem.

Applied to scheduling problems, constraint-based programming enables the implementation of precise, efficient, flexible and extensible scheduling systems [Le Pape 92]: precise and flexible as the system can take into account any constraint expressible in the constraint language; efficient inasmuch as highly optimized constraint propagation techniques, tailored to the requirements of scheduling applications, are now available [Aggoun 92] [Le Pape 93]; extensible as the consideration of a new type of constraint may require (especially in an object-oriented framework) only an extension to the constraint system or, in the worst case, the implementation of additional decision-making modules (without needs for modification of the existing code).

The development of numerous constraint-based scheduling systems such as ISIS [Fox 83] [Fox 84] [Smith 83], SOJA [Le Pape 85], OPIS [Smith 86a] [Smith 86b] [Smith 87], FLYPAST [Mott 88], SONIA [Collinot 88] [Collinot 91] [Le Pape 88], DAS [Burke 89a] [Burke 89b] [Prosser 90] and, more recently, the implementation of resource constraints within industrial constraint-based programming tools such as CHIP [Van Hentenryck 89] [Aggoun 92] and ILOG SOLVER [Puget 94] [Le Pape 94b] constitute significant steps toward the development of precise, flexible and efficient scheduling systems.

For many researchers in the field, it is Mark Fox who, at the beginning of the 1980s, initiates the development of constraint-based scheduling techniques. The ISIS system [Fox 83] [Fox 84] [Smith 83] is indeed the first constraint-based system dedicated to scheduling problems.

However, from an historical point of view, it is interesting to go back to the first Operations Research results — to the design of the first algorithms enabling the determination of shortest

¹Working Paper, Operations Research Society Seminar on Constraint Handling Techniques, London, United Kingdom, 1994.

and longest paths in graphs. These algorithms enable the resolution of the so-called “central” scheduling problem, i.e. the problem of scheduling n activities subjected to precedence constraints, delay constraints (minimal and maximal delays between activities), and time-bound constraints (earliest and latest start and end times for each activity), but not subjected to resource constraints [Roy 70]. One of these algorithms is particularly interesting from a constraint-based programming point of view: Ford’s algorithm [Ford 56], which computes the length of the shortest (respectively longest) paths from a node N_1 to the other nodes $N_2 \dots N_n$ of a valued oriented graph.

1. Let $\pi(N_1) = 0$ and $\pi(N_i) = +\infty$ (respectively $-\infty$) for every i between 2 and n .
2. For every i between 2 and n , replace $\pi(N_i)$ with $\min(\pi(N_i), \min(\pi(N_j) + l_{ji}))$ (respectively $\max(\pi(N_i), \max(\pi(N_j) + l_{ji}))$) where l_{ji} denotes the length of the arc going from N_j to N_i when this arc exists.
3. Iterate until $\pi(N_i)$ becomes stable for every i .

This algorithm is interesting because one can implement an incremental version of it. Indeed, at each iteration, it is useless to re-compute $(\pi(N_j) + l_{ji})$ if the preceding iteration did not result in an update of $\pi(N_j)$. Consequently, the algorithm can be modified to make the update of $\pi(N_j)$ trigger, at the next iteration, the update of $\pi(N_i)$. This is what *constraint propagation* systems do [Le Pape 88].

The beginning of the 1960s is marked by the design and implementation of heuristic search procedures, in particular “Branch and Bound” procedures (Little’s algorithm [Little 63], for example). A heuristic search procedure is obtained by associating *heuristics* with a *search algorithm*. The search algorithm implicitly defines a space of *partial problem solutions* and explores it until an acceptable or optimal *complete solution* is found. The mechanism used to perform the exploration is iterative and very simple. The algorithm starts from an obvious or from an already known partial solution and determines the changes that can be made to this partial solution. Heuristic knowledge is used to evaluate the possible changes and the quality of the resulting partial solutions. This knowledge is used to select, among the resulting partial solutions, those that are worth modifying in turn. At each step in the problem-solving process, the algorithm updates (for example, through constraint propagation!) various pieces of information to which the heuristics refer and *cuts* the search by eliminating partial solutions from which no complete solution (or no complete solution better than already known solutions) can be obtained.

When the *unification* and *resolution* principles (foundations of the PROLOG programming language) are proposed by Robinson in 1965 [Robinson 65], Operations Research and Artificial Intelligence seem to constitute separate disciplines. Jean-Louis Laurière proposes to use Artificial Intelligence techniques for the resolution of time-tabling problems, and to “harmonize and generalize certain methods” (Branch and Bound) “born in the two domains” [Laurière 71]. Five years later, Jean-Louis Laurière presents ALICE, “a langage and a program for stating and solving combinatorial problems” [Laurière 76] [Laurière 78] — hence (in particular) for stating and solving scheduling problems. ALICE uses (1) heuristics to make decisions, (2) propagation methods to deduce consequences of these decisions, and (3) a tree search strategy to guarantee the generation of a complete solution when one exists (and guarantee the generation of an optimal solution in the case of an optimization problem). In this respect, ALICE is the ancestor of many constraint-based scheduling systems.

The 1970s are quite important years in the scheduling domain. Two important books are edited in 1974 [Baker 74] and 1976 [Coffman 76]; these books sum up the existing knowledge on “scheduling theory” [Baker 74]. In addition, many scheduling problems are shown NP-complete [Garey 79]. An NP-complete problem is a problem for which it is conjectured that there exists no algorithm enabling to solve the problem in an amount of time bounded by a polynomial function of the size of the data. For example, the problem of scheduling n activities of given durations on two identical resources is NP-complete: it is conjectured that, for every algorithm enabling to minimize the duration (makespan) of the overall schedule, and for every polynomial function $P(n)$, there exists an integer m and a set of m durations such that the execution of the algorithm requires more than $P(m)$ units of CPU time. In the face of these complexity results, two possibilities are considered: (1) design robust algorithms, to generate solutions (optimal solutions) as often as possible in an acceptable amount of time; (2) design approximate algorithms, to generate approximate (sub-optimal) solutions in a bounded amount of time.

In 1976, Jacques Erschler [Erschler 76] proposes methods of “constraints analysis” aimed at making tree search procedures more robust. In fact, a significant number of recent Operations Research results derive from the same idea: a mechanism alternating the making of decisions and the deduction (propagation) of the consequences of these decisions [Carlier 84] [Esquirol 87] [Carlier 88] [Pinson 88] [Carlier 89] [Carlier 90] [Lopez 91] [Applegate 91]. Some specialists of the Operations Research and the constraint programming communities accomplish very complementary pieces of work: on the one hand, determining which heuristics and which propagation methods to use to efficiently solve scheduling problems; on the other hand, determining how to implement, in a generic yet economical way, such heuristics and such constraint propagation methods.

The fundamental principles of constraint programming become precise in the late 1970s and early 1980s: separation between constraint propagation methods and search algorithms [Stallman 77]; exploitation of propagation results to perform “dependency-directed” rather than “chronological” backtracking [Stallman 77] [Latombe 79]; “locality principle”, which states that each constraint must propagate as locally as possible, independently of the existence or the non-existence of other constraints [Steele 80]; distinction between the logical representation of constraints and the control of their use, in accordance with the equation stated by Kowalski for logic programming:

$$\text{ALGORITHM} = \text{LOGIC} + \text{CONTROL} \text{ [Kowalski 79].}$$

The first significant applications in the planning and scheduling domain appear: Ira Goldstein and Bruce Roberts develop a system for planning appointments [Goldstein 75] [Goldstein 77]; James Allen develops a formalism enabling the representation (and propagation) of symbolic temporal constraints in a plan of actions [Allen 81] [Allen 83] [Allen 84] (work extended and generalized by Marc Vilain and Henry Kautz [Vilain 82] [Vilain 86], Peter Ladkin and Roger Maddux [Ladkin 88], Jean-François Rit [Rit 86] [Rit 88], Peter Van Beek [Van Beek 89] [Van Beek 90], Malik Ghallab and Amine Mounir Alaoui [Ghallab 89a] [Ghallab 89b]); Steven Vere develops a planner which propagates numeric temporal constraints [Vere 83]; Yannick Descotte and Jean-Claude Latombe develop a planner which makes compromises between relaxable constraints [Descotte 81] [Descotte 85]; and Isis, the first constraint-based scheduling system, is designed and implemented by Mark Fox and his team at Carnegie-Mellon University [Fox 83] [Fox 84] [Smith 83].

The most important issue considered by the designers of ISIS is the representation of the overall set of constraints that define the scheduling problem to be solved. ISIS may be considered mainly as a tool for representing the constraints of the manufacturing environment. The Schema Representation Language SRL [Fox 83] [Wright 84] is used to represent, in a structured fashion, the objects manipulated in the factory, the activities to perform, and the constraints to be satisfied. The scheduling method per se derives from two important design decisions, aimed at breaking the overall scheduling problem into simpler, more tractable, subproblems:

- ISIS relies on an order-based decomposition of the scheduling problem. This method consists of successively scheduling each manufacturing order. A complete schedule is obtained by incrementally constructing a schedule for each order, taking into account the decisions made for the preceding orders.
- For each order, ISIS introduces the scheduling constraints in a hierarchical fashion (from the most important to the less important constraints), thereby constructing an order schedule through successive refinements.

Constraints are used (1) to organize the successive problem-solving steps and (2) to evaluate the characteristics of the obtained schedule. The order-based decomposition enables the establishment of good compromises between the constraints that concern the most important orders. For these orders, a significant number of scheduling alternatives can be considered, combined and compared in an acceptable amount of computational time. On the other hand, the order-based decomposition does not enable a good satisfaction of resource-based optimization criteria. The successors of ISIS do not impose similar static problem decompositions.

The SOJA system [Le Pape 85] is a short-term scheduling system composed of a “selection” module and a “scheduling” (constraint satisfaction) module. These two modules rely on an inference engine to apply selection and scheduling rules. The user of the system elaborates and orders the rules with respect to the scheduling criteria (s)he wants to satisfy in priority. The selection module computes the overall capacity of each resource over the time period to schedule (typically, a day or a week) and selects a set of activities to schedule on these resources. The scheduling module determines start and end times for the selected activities, so as to satisfy both absolute (non-relaxable) constraints and preferences. Scheduling consists of an iterative decision-making process. At each step, a decision is made to guarantee the satisfaction of a constraint. This decision is immediately translated into new constraints and propagated. When a contradiction is detected, the scheduling module backtracks and cancels its most recent decisions. If too many activities are selected, the scheduling module fails and rejects some activities. On the contrary, if some resources are under-loaded at the end of the scheduling process, the user is entitled to require the selection of additional activities. Then, the scheduling module schedules these activities without cancelling the preceding decisions. The SOJA system, designed to enable the user to adapt the system’s heuristic knowledge (expressed in rules) to his or her own scheduling problem, was industrialized and marketed by GSI-INDUSTRIE.

Very flexible scheduling systems, e.g., OPIS [Smith 86a] [Smith 86b] [Smith 87], FLYPAST [Mott 88], SONIA [Collinot 88] [Collinot 91] [Le Pape 88] and DAS [Burke 89a] [Burke 89b] [Prosser 90] are built at the end of the 1980s. OPIS, FLYPAST, SONIA and DAS are *reactive* scheduling systems. This means that they do not only allow the generation of a schedule, but also the incremental

revision of the schedule in response to the unavoidable occurrence of unexpected events such as delays or machine breakdowns [Graves 81]. To allow schedule revision, these four systems maintain information about the origins of the constraints deduced by propagation. This information allows the system (1) to construct a comprehensive description of the contradictions detected by propagation and (2) to backtrack in a selective (non-chronological) fashion.

Each of these systems relies on its own constraint propagation methods:

- OPIS includes an object-oriented constraint propagation module, which simplifies the implementation of hierarchical relations between resources and between activities [Le Pape 87].
- SONIA includes a constraint propagation module based on a logical representation of constraints. This enables the user to *control* constraint propagation with respect to the characteristics of the problem-solving context [Collinot 87].
- FLYPAST uses a constraint propagation module coupled to an assumption-based truth maintenance system [Kelleher 90]. As a result, dependencies between constraints are managed in a generic fashion. This compares favorably with the “ad hoc” truth maintenance methods used in OPIS and SONIA. (See [De Kleer 86] [McDermott 91] for an introduction to truth maintenance systems and [Pfeffer 92a] [Pfeffer 92b] for an interesting study on the coupling of constraint propagation and truth maintenance.)
- Finally, DAS is a system in which the constraint propagation and the constraint satisfaction processes are distributed among different processors.

However, despite those differences, it is the same fundamental principle which guarantees the correctness, flexibility and extensibility of the four systems: the separation between (1) the definition of the constraints, (2) the propagation of the constraints, and (3) the specific problem-solving strategies used to solve the scheduling problem under consideration [Le Pape 92] [Le Pape 94a].

The CLP(R) [Jaffar 87] and PROLOG-III [Colmerauer 87] [Colmerauer 90] constraint logic programming languages are created in 1987, soon followed by the CHIP language [Van Hentenryck 89], to which Abderrahmane Aggoun and Nicolas Beldiceanu add cumulative resource constraints in 1992 [Aggoun 92]. The ILOG SOLVER object-oriented constraint-based programming library is created in 1991 (LE-LISP version) and 1992 (C++ version) [Puget 91] [Puget 92] [Puget 94]. A generic mechanism for the management of resource constraints is added to ILOG SOLVER in 1993, in the form of a higher-level library: ILOG SCHEDULE [Le Pape 93] [Le Pape 94b]. These tools constitute a significant basis for the development of industrial scheduling and resource allocation systems. They shall also facilitate research and development on a number of important topics such as:

- the extension of existing constraint propagation methods: generalization of existing constraint propagation methods [Le Pape 93]; design of new constraint propagation methods (for example, by systematizing the use of global, redundant, constraints [Beck 92] [Caseau 93]); better evaluation of partial schedules;
- the relaxation of a scheduling problem through the omission of constraints or through data aggregation [Berry 93];

- the implementation of incremental, reactive, scheduling techniques, facilitating user intervention (as in WEDGE - COMPASS [Fox 89] [McMahon 89] [Fox 90]), without paying (as in SONIA or FLYPAST) the cost of memorizing dependences between constraints [Kelleher 93a] [Kelleher 93b];
- the design and implementation of probabilistic [Muscettola 87] [Sadeh 90] [Berry 91] and/or fuzzy constraint models [Kerr 89], to reason about the imprecisions and inaccuracies of scheduling data;
- the generation of robust schedules, i.e., of schedules which are likely to remain “good” despite the unavoidable occurrence of unexpected events.

Constraint-based applications enabling the resolution of very large and complex problems have been implemented in the recent years (see, for example, the 10,000 variables problem described in [Le Pape 94c]). There is no doubt that the current state of the constraint-based programming technology and the accelerating accumulation of results in this domain will, in the forthcoming years, enable the development of solutions to even more complex problems, some of which are still believed unsolvable today.

References

- [Aggoun 92] Abderrahmane Aggoun and Nicolas Beldiceanu. *Extending CHIP in Order to Solve Complex Scheduling and Placement Problems*. Actes des premières journées francophones sur la programmation en logique, Lille, France, 1992.
- [Allen 81] James F. Allen. *An Interval-Based Representation of Temporal Knowledge*. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981.
- [Allen 83] James F. Allen. *Maintaining Knowledge about Temporal Intervals*. Communications of the ACM, 26(11):832-843, 1983.
- [Allen 84] James F. Allen. *Towards a General Theory of Action and Time*. Artificial Intelligence, 23(2):123-154, 1984.
- [Applegate 91] David Applegate and William Cook. *A Computational Study of the Job-Shop Scheduling Problem*. ORSA Journal on Computing, 3(2):149-156, 1991.
- [Baker 74] Kenneth R. Baker. *Introduction to Sequencing and Scheduling*. John Wiley and Sons, 1974.
- [Beck 92] Howard Beck. *Constraint Monitoring in TOSCA*. Working Papers of the AAAI Spring Symposium on Practical Approaches to Planning and Scheduling, Stanford, California, 1992.
- [Berry 91] Pauline M. Berry. *A Predictive Model for Satisfying Conflicting Objectives in Scheduling Problems*. PhD Thesis, University of Strathclyde, 1991.

- [Berry 93] Pauline M. Berry. *Uncertainty in Scheduling: Probability, Problem Reduction, Abstractions and the User*. Proceedings of the IEE Colloquium on Advanced Software Technologies for Scheduling, London, United Kingdom, 1993.
- [Burke 89a] Peter Burke and Patrick Prosser. *A Distributed Asynchronous System for Predictive and Reactive Scheduling*. Technical Report, University of Strathclyde, 1989.
- [Burke 89b] Peter Burke. *Scheduling in Dynamic Environments*. PhD Thesis, University of Strathclyde, 1989.
- [Carlier 84] Jacques Carlier. *Problèmes d’ordonnancement à contraintes de ressources : algorithmes et complexité*. Thèse de Doctorat d’Etat, Université Paris VI, 1984.
- [Carlier 88] Jacques Carlier et Philippe Chrétienne. *Problèmes d’ordonnancement : Modélisation / Complexité / Algorithmes*. Masson, 1988.
- [Carlier 89] Jacques Carlier and Eric Pinson. *An Algorithm for Solving the Job-Shop Problem*. Management Science, 35(2):164-176, 1989.
- [Carlier 90] Jacques Carlier and Eric Pinson. *A Practical Use of Jackson’s Preemptive Schedule for Solving the Job-Shop Problem*. Annals of Operations Research, 26:269-287, 1990.
- [Caseau 93] Yves Caseau, Pierre-Yves Guillo and Eric Levenez. *A Deductive and Object-Oriented Approach to a Complex Scheduling Problem*. Proceedings of the International Conference on Deductive and Object-Oriented Databases, Phoenix, Arizona, 1993.
- [Coffman 76] Edward G. Coffman Jr. (editor). *Computers and Job-Shop Scheduling Theory*. John Wiley and Sons, 1976.
- [Collinot 87] Anne Collinot and Claude Le Pape. *Controlling Constraint Propagation*. Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Milan, Italy, 1987.
- [Collinot 88] Anne Collinot. *Le problème du contrôle dans un système flexible d’ordonnancement*. Thèse de l’Université Paris VI, 1988.
- [Collinot 91] Anne Collinot and Claude Le Pape. *Adapting the Behavior of a Job-Shop Scheduling System*. International Journal for Decision Support Systems, 7(3):341-353, 1991.
- [Colmerauer 87] Alain Colmerauer. *Opening the Prolog III Universe*. Byte, 12(9):177-182, 1987.
- [Colmerauer 90] Alain Colmerauer. *An Introduction to Prolog III*. Communications of the ACM, 33(7):69-90, 1990.
- [De Kleer 86] Johan De Kleer. *An Assumption-Based TMS. Extending the ATMS. Problem Solving with the ATMS*. Artificial Intelligence, 28(2):127-224, 1986.

- [Descotte 81] Yannick Descotte. *Représentation et exploitation de connaissances "expertes" en génération de plans d'actions. Application à la conception automatique de gammes d'usinage*. Thèse de troisième cycle, Institut National Polytechnique de Grenoble, 1981.
- [Descotte 85] Yannick Descotte and Jean-Claude Latombe. *Making Compromises among Antagonist Constraints in a Planner*. *Artificial Intelligence*, 27(2):183-217, 1985.
- [Erschler 76] Jacques Erschler. *Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnement*. Thèse de Doctorat d'Etat, Université Paul Sabatier, 1976.
- [Esquirol 87] Patrick Esquirol. *Règles et processus d'inférence pour l'aide à l'ordonnement de tâches en présence de contraintes*. Thèse de l'Université Paul Sabatier, 1987.
- [Ford 56] Lester R. Ford Jr. *Network Flow Theory*. Rand Corporation, 1956.
- [Fox 89] Barry R. Fox. *Mixed Initiative Scheduling*. Proceedings of the AAAI Spring Symposium on Artificial Intelligence in Scheduling, Stanford, California, 1989.
- [Fox 90] Barry R. Fox. *Chronological and Non-Chronological Scheduling*. Proceedings of the First Annual Conference on Artificial Intelligence, Simulation and Planning in High Autonomy Systems, Tucson, Arizona, 1990.
- [Fox 83] Mark S. Fox. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. PhD Thesis, Carnegie-Mellon University, 1983.
- [Fox 84] Mark S. Fox and Stephen F. Smith. *ISIS: A Knowledge-Based System for Factory Scheduling*. *Expert Systems*, 1(1):25-49, 1984.
- [Garey 79] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
- [Ghallab 89a] Malik Ghallab and Amine Mounir Alaoui. *Managing Efficiently Temporal Relations Through Indexed Spanning Trees*. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan, 1989.
- [Ghallab 89b] Malik Ghallab et Amine Mounir Alaoui. *Relations temporelles symboliques : représentations et algorithmes*. *Revue d'intelligence artificielle*, 3(3):67-115, 1989.
- [Goldstein 75] Ira P. Goldstein. *Bargaining Between Goals*. Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilissi, USSR, 1975.
- [Goldstein 77] Ira P. Goldstein and Bruce R. Roberts. *NUDGE: A Knowledge-Based Scheduling Program*. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts, 1977.
- [Graves 81] Stephen C. Graves. *A Review of Production Scheduling*. *Operations Research*, 29(4):646-675, 1981.

- [Jaffar 87] Joxan Jaffar and Jean-Louis Lassez. *Constraint Logic Programming*. Proceedings of the Fourteenth Annual ACM Symposium on Principles of Programming Languages, Munich, Germany, 1987.
- [Kelleher 90] Gerald Kelleher. *Extending the Expressive Power of de Kleer's ATMS*. Technical Report, University of Leeds, 1990.
- [Kelleher 93a] Gerald Kelleher. *Emerging Reason Maintenance System Technologies and Their Application to Constraint-Based Scheduling*. Proceedings of the Twelfth UK Planning SIG, Cambridge, United Kingdom, 1993.
- [Kelleher 93b] Gerald Kelleher and Philippe Retif. *Controlling Constraint-Based Scheduling Using Focussed RMS*. Proceedings of the AAAI-SIGMAN Workshop on Knowledge-Based Production Planning, Scheduling and Control, IJCAI, Chambéry, France, 1993.
- [Kerr 89] R. M. Kerr and R. N. Walker. *A Job-Shop Scheduling System Based on Fuzzy Arithmetic*. Proceedings of the Third International Conference on Expert Systems and the Leading Edge in Production Planning and Control, Charleston, South Carolina, 1989.
- [Kowalski 79] Robert Kowalski. *Algorithm = Logic + Control*. Communications of the ACM, 22(7):424-436, 1979.
- [Ladkin 88] Peter B. Ladkin and Roger D. Maddux. *On Binary Constraint Networks*. Technical Report, Kestrel Institute, 1988.
- [Latombe 79] Jean-Claude Latombe. *Failure Processing in a System for Designing Complex Assemblies*. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan, 1979.
- [Laurière 71] Jean-Louis Laurière. *Sur la coloration de certains hypergraphes. Application aux problèmes d'emploi du temps*. Thèse de troisième cycle, Université Paris VI, 1971.
- [Laurière 76] Jean-Louis Laurière. *Un langage et un programme pour énoncer et résoudre des problèmes combinatoires*. Thèse de Doctorat d'Etat, Université Paris VI, 1976.
- [Laurière 78] Jean-Louis Laurière. *A Language and a Program for Stating and Solving Combinatorial Problems*. Artificial Intelligence, 10(1):29-127, 1978.
- [Le Pape 85] Claude Le Pape. *SOJA: A Daily Workshop Scheduling System. SOJA's System and Inference Engine*. Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems, Warwick, United Kingdom, 1985.
- [Le Pape 87] Claude Le Pape and Stephen F. Smith. *Management of Temporal Constraints for Factory Scheduling*. Proceedings of the Working Conference on Temporal Aspects in Information Systems, Sophia-Antipolis, France, 1987.
- [Le Pape 88] Claude Le Pape. *Des systèmes d'ordonnancement flexibles et opportunistes*. Thèse de l'Université Paris XI, 1988.

- [Le Pape 92] Claude Le Pape. *Programmation par contraintes et ordonnancement : réalités et perspectives*. Actes du deuxième congrès “Systèmes Experts en Informatique de Gestion”, Nice, France, 1992.
- [Le Pape 93] Claude Le Pape. *The Cost of Genericity: Experiments with Constraint-Based Representations of Time-Tables*. Proceedings of the Sixth International Conference on Software Engineering and its Applications, Paris La Défense, France, 1993.
- [Le Pape 94a] Claude Le Pape. *Validity and Completeness Properties of Blackboard Systems Using Constraint Propagation*. Jayantha Herath (editor), Readings in Computer Architectures for Intelligent Systems, IEEE Computer Society Press (to appear).
- [Le Pape 94b] Claude Le Pape. *Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems*. Intelligent Systems Engineering (to appear).
- [Le Pape 94c] Claude Le Pape, Jean-François Puget, Colonel Moreau and Philippe Darneau. *PMFP: The Use of Constraint-Based Programming for Predictive Personnel Management*. Proceedings of the Eleventh European Conference on Artificial Intelligence, Amsterdam, The Netherlands, 1994 (to appear).
- [Little 63] John Little, Katta Murty, Dura Sweeney and Caroline Karel. *An Algorithm for the Traveling Salesman Problem*. Operations Research, 11(6):972-989, 1963.
- [Lopez 91] Pierre Lopez. *Approche énergétique pour l’ordonnancement de tâches sous contraintes de temps et de ressources*. Thèse de l’Université Paul Sabatier, 1991.
- [McDermott 91] Drew McDermott. *A General Framework for Reason Maintenance*. Artificial Intelligence, 50(3):289-329, 1991.
- [McMahon 89] Mary Beth McMahon and Jack Dean. *A Simulated Annealing Approach to Schedule Optimization for the SES Facility*. Proceedings of the AAAI Spring Symposium on Artificial Intelligence in Scheduling, Stanford, California, 1989.
- [Mott 88] David H. Mott, Jon Cunningham, Gerald Kelleher and Julie A. Gadsden. *Constraint-Based Reasoning for Generating Naval Flying Programmes*. Expert Systems, 5(3):226-246, 1988.
- [Muscettola 87] Nicola Muscettola and Stephen F. Smith. *A Probabilistic Framework for Resource-Constrained Multi-Agent Planning*. Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Milan, Italy, 1987.
- [Pfeffer 92a] Nathalie Pfeffer. *Intégrer le maintien de cohérence dans un système combinant inférences expertes et propagation de contraintes*. Douzièmes journées internationales sur les systèmes experts et leurs applications, Avignon, France, 1992.
- [Pfeffer 92b] Nathalie Pfeffer. *Connaissances expertes, propagation de contraintes et maintien de cohérence pour l’aide à la conception*. Thèse de l’Université Paris XI, 1992.

- [Pinson 88] Eric Pinson. *Le problème de job-shop*. Thèse de l'Université Paris VI, 1988.
- [Prosser 90] Patrick Prosser. *Distributed Asynchronous Scheduling*. PhD Thesis, University of Strathclyde, 1990.
- [Puget 91] Jean-François Puget et Patrick Albert. *PECOS : programmation par contraintes orientée objets*. Génie logiciel et systèmes experts, (23):100-105, 1991.
- [Puget 92] Jean-François Puget. *Programmation par contraintes orientée objet*. Douzièmes journées internationales sur les systèmes experts et leurs applications, Avignon, France, 1992.
- [Puget 94] Jean-François Puget. *A C++ Implementation of CLP*. Technical Report, ILOG S.A., 1994.
- [Rit 86] Jean-François Rit. *Propagating Temporal Constraints for Scheduling*. Proceedings of the Fifth National Conference on Artificial Intelligence, Philadelphia, Pennsylvania, 1986.
- [Rit 88] Jean-François Rit. *Modélisation et propagation de contraintes temporelles pour la planification*. Thèse de l'Institut National Polytechnique de Grenoble, 1988.
- [Robinson 65] J. A. Robinson. *A Machine-Oriented Logic Based on the Resolution Principle*. Journal of the ACM, 12(1):23-41, 1965.
- [Roy 70] Bernard Roy. *Algèbre moderne et théorie des graphes (tome 2)*. Dunod, 1970.
- [Sadeh 90] Norman Sadeh and Mark S. Fox. *Variable and Value Ordering Heuristics for Activity-Based Job-Shop Scheduling*. Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, Hilton Head, South Carolina, 1990.
- [Smith 83] Stephen F. Smith. *Exploiting Temporal Knowledge to Organize Constraints*. Technical Report, Carnegie-Mellon University, 1983.
- [Smith 86a] Stephen F. Smith, Peng Si Ow, Claude Le Pape, Bruce McLaren and Nicola Muscettola. *Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans*. Proceedings of the SME Conference on Artificial Intelligence in Manufacturing, Long Beach, California, 1986.
- [Smith 86b] Stephen F. Smith, Mark S. Fox and Peng Si Ow. *Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems*. AI Magazine, 7(4):45-61, 1986.
- [Smith 87] Stephen F. Smith. *A Constraint-Based Framework for Reactive Management of Factory Schedules*. Proceedings of the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control, Charleston, South Carolina, 1987.

- [Stallman 77] Richard M. Stallman and Gerald J. Sussman. *Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis*. Artificial Intelligence, 9(2):135-196, 1977.
- [Steele 80] Guy Lewis Steele Jr. *The Definition and Implementation of a Computer Programming Language Based on Constraints*. PhD Thesis, Massachusetts Institute of Technology, 1980.
- [Van Beek 89] Peter Van Beek. *Approximation Algorithms for Temporal Reasoning*. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan, 1989.
- [Van Beek 90] Peter Van Beek. *Reasoning about Qualitative Temporal Information*. Proceedings of the Eighth National Conference on Artificial Intelligence, Boston, Massachusetts, 1990.
- [Van Hentenryck 89] Pascal Van Hentenryck. *Constraint Satisfaction in Logic Programming*. MIT Press, 1989.
- [Vere 83] Steven A. Vere. *Planning in Time: Windows and Durations for Activities and Goals*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(3):246-267, 1983.
- [Vilain 82] Marc Vilain. *A System for Reasoning about Time*. Proceedings of the Second National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania, 1982.
- [Vilain 86] Marc Vilain and Henry Kautz. *Constraint Propagation Algorithms for Temporal Reasoning*. Proceedings of the Fifth National Conference on Artificial Intelligence, Philadelphia, Pennsylvania, 1986.
- [Wright 84] Mark Wright, Mark Fox and David Adam. *SRL/1.5 User Manual*. The Robotics Institute, Carnegie-Mellon University, 1984.