VISEXP: visualizing constraint solver dynamics using explanations*

Mohammad Ghoniem and **Narendra Jussien**
École des Mines de Nantes
4 rue Alfred Kastler
F-44307 Nantes Cedex 3, France
{mghoniem,jussien}@emn.fr

Jean-Daniel Fekete
INRIA Futurs
Bat. 490, Université Paris-Sud
F-91405 Orsay Cedex, France
Jean-Daniel.Fekete@inria.fr

Abstract

In this paper, we introduce VISEXP: a new visualization tool designed to explore relations between constraints and variables in constraint problems. This tool uses the explanation network built throughout computation. We show that VISEXP is able to provide much more information about how search is performed than classical representations. Moreover, we illustrate the animation feature of VISEXP that provides invaluable tools for visualization and therefore analysis of the dynamics of constraint solvers.

Introduction

Constraint programming has now proven its effectiveness for a wide range of problems and applications. However, debugging constraint applications or analyzing the behavior of a constraint solver is still an important issue. Indeed, as applications become more complex, adapted tools need to be provided to users and application developers.

Various work have considered using explanations (subsets of constraints justifying solver actions (Jussien 2003)) to overcome that issue. For example, (Junker 2001; Amilhastre, Fargier, & Marquis 2002) explain inconsistencies in configuration applications, (Ouis, Jussien, & Boizumault 2003) introduce explanation-based tools for user-interaction, (Jussien 2003; Freuder, Likitvivatanavong, & Wallace 2001; O'Callaghan, Freuder, & O'Sullivan 2003) introduce explanation-based tools for user interaction in constraint programming, etc.

In this paper, we show that explanations considered as a trace of the behavior of the solver (as in the PALM system (Jussien & Barichard 2000)) for example can be used, with the proper visualization tools, as a source of information about relations between variables and constraints that arise through propagation. We show that visualizing those relations and their dynamics is an invaluable tool that: exhibits static or dynamic propagation related structure between variables or constraints, exhibits ineffective decisions made during search, exhibits hard resolution steps, exhibits the real dynamics of the solver, etc.

*This work has been partially supported by the French RNTL project OADYMPPAC.

We claim that using the information contained in the explanation network helps discover information that is not available when looking only at the structure of the solved problem or at domain reductions as they are performed during search. Moreover, the explanation network provides insights into the dynamics of search (as opposed to static exploration of conflicts or search procedures as introduced for example in (Pu & Lalanne 2000)).

We present VISEXP, a visualization tool that takes as input a generic trace for constraint solvers as defined in (Deransart, Ducassé, & Langevine 2002) making it compatible with any constraint solver able to provide explanations.

This paper is organized as follows. Explanations for constraint programming are introduced and their use as a link between variables and constraints is described. Then, we introduce VISEXP, the visualization tool we developed in this specific context and its application to the visualization of the explanation network. Finally, various experiments illustrate the power of such a tool.

Explanations for constraint programming

Explanations (specializing (A)TMS (Doyle 1979)) and generalizing nogoods (Schiex & Verfaillie 1994)) have been initially introduced to improve backtracking-based algorithms (Ginsberg 1993). However, they have been recently used for many other purposes (Jussien 2003) including debugging and analysis of the behavior of constraint solvers.

Definition

An explanation contains enough information to justify a decision (throwing a contradiction, reducing a domain, etc.): it is composed of the constraints and the choices made during the search which are sufficient to justify such an inference.

Definition 1 (Explanation) *An explanation of an inference ($\mathcal{X}$) consists of a subset of original constraints ($\mathcal{C}' \subset \mathcal{C}$) and a set of instantiation constraints (choices made during the search: $d_1, d_2, \ldots, d_k$) such that:*

$$\mathcal{C}' \wedge d_1 \wedge \ldots \wedge d_n \Rightarrow \mathcal{X}$$

$\mathcal{C}' \wedge d_1 \wedge \ldots \wedge d_n$ is called an explanation.

An explanation e_1 is said to be *more precise* than explanation e_2 if and only if $e_1 \subset e_2$. The more precise an explanation, the more useful it is.

Computing explanations

The most interesting explanations are those which are minimal regarding inclusion. Those explanations allow highly focused information about dependencies between constraints and variables. Unfortunately, computing such an explanation can be exponentially costly. A good compromise between precision and ease of computation consists in using the solver embedded knowledge to provide explanations (Jussien & Barichard 2000). Indeed, constraint solvers always know, although it is scarcely explicit, *why* they remove values from the domain of the variables. By making that knowledge explicit, quite precise and interesting explanations can be computed as constraint solvers are supposed to efficiently perform their task! Therefore, explanations strongly depend both on the constraint solver at hand and on the way the problem is modelled. For example, let us consider variables v_1 and v_2 with domains $\{1, 2, 3\}$.

- Let c_1 be a first decision constraint: $c_1 : v_1 \geq 3$. Let us assume that the filtering algorithm in use is arc-consistency filtering (Mohr & Henderson 1986). The constraint c_1 leads to the removal of $\{1, 2\}$ from the domain of v_1. An explanation for the new domain $\{3\}$ of v_1 is thus $\{c_1\}$.
- Let c_2 be a second constraint: $c_2 : v_2 \geq v_1$. Value 1 and value 2 of v_2 have no support in the domain of v_1, and thus c_2 leads to the removal of $\{1, 2\}$ from v_2. An explanation of the removal of $\{1, 2\}$ from v_2 will be: $c_1 \wedge c_2$ because c_2 precipitates that removal only because previous removals occurred in v_1 due to c_1.

The explanation network

Usually, explanations may be used for user interaction (Ouis, Jussien, & Boizumault 2003; Freuder, Likitvivatanavong, & Wallace 2001), dynamic constraint handling (Debruyne *et al.* 2003) or even improvement of search algorithms (Ginsberg 1993; Jussien, Debruyne, & Boizumault 2000; Jussien & Lhomme 2002). However, explanations, as we defined them, provide insightful information about the constraint solver dynamics.

Explanations induce a pair of networks, the first one links the constraints that cooperate at some level to solve the problem, whereas the second network relates the variables of the problem with regard to their impact on one another. Therefore, considering explanations or, more precisely, the pair of networks they induce between constraints and between variables, proves to be fruitful for understanding the behavior of the constraint solver on a given problem. Indeed, constraints appearing in an explanation are cooperating (event if they do not share any variable) to remove some values and are therefore related. Similarly, variables related to constraints appearing in an explanation are related during resolution.

The aim of this paper is to show that it may be interesting to visualize and analyze the relations (and their dynamics) between constraints and between variables. Notice that we are interested in the use of the explanation network as a representation of the solver dynamics. Therefore, *a priori* (Amilhastre, Fargier, & Marquis 2002) or *a posteriori* (Junker 2001) explanation computation techniques would be of no use here.

`VISEXP`: our visualization tool

The number of explanations may be really large when solving large problems. Therefore, the number of links between constraints or between variables may be extremely large and the resulting graph really dense. This introduces two issues: how to differentiate strong links (constraints/variables often cooperating) and loose links (constraints/variables seldom cooperating); how to represent such a dense graph. The first issue is easily overcome by introducing a notion of weight (representing the number of relations between constraints/variables) in the graph. The second issue demands a new representation of graphs.

An alternate representation of graphs

So far, visualization of networks has mainly focused on node-link diagrams because they are popular and well understood. However, node-link diagrams do not scale well: their layout is slow and they become quickly unreadable when the size of the graph and link density increase.

In this paper, we present a recent technique that uses adjacency matrices instead of node-link diagrams to interactively visualize and explore large graphs, with thousands of nodes and any number of links. This technique relies on the well known property that a graph may be represented by its connectivity matrix, which is an N by N matrix, where N is the number of vertices in the graph, and each row or column in the matrix stands for a vertex. When two vertices V_i and V_j are linked, the corresponding coefficient m_{ij} in the matrix is set to 1, otherwise it is set to 0.

From a visualization standpoint, not only do we switch on or off the cell located at the intersection of V_i and V_j, but we use color coding as well when dealing with weighted links: the heavier the weight (here the number of interactions), the darker a link. Unlike node-link diagrams, matrix-based representations of graphs do not suffer from link and node overlappings. Virtually every link (out of the $N^2/2$ links) in the graph can be seen separately (see figure 1). With this technique, we can show as many links as the display hardware resolution allows, roughly 2 million pixels on a regular 1600×1200 display. Moreover, advanced information visualisation techniques such as dynamic queries (Card, Mackinlay, & Shneiderman 1999), fisheye lenses (Carpendale & Montagnese 2001) and excentric labels (Fekete & Plaisant 1999) enhance the exploration of large graphs and push the matrix-based visualization one step further in coping with large networks.

The main tradeoff of such a technique lies in the fact that vertices are no longer represented by a unique graphic symbol, they are rather distributed on both axes of the matrix. This is why users may often need some training before they get familiar with the matrix metaphor. Further investigation of this technique in terms of human-computer interaction is in progress. It aims at assessing more formally the advantages and weaknesses of the matrix metaphor compared to the traditional node-link metaphor. At this stage, one may already note that the node-link representation becomes quickly unreadable while the number of nodes and the density of links increase, whereas users can still perform

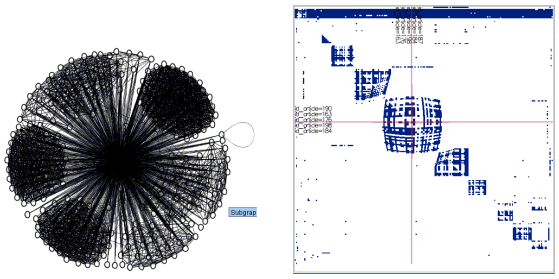

Figure 1: Representing a graph with 220 vertices and 6291 links using a node-link classical diagram (**left**) and an adjacency matrix (**right**). The matrix view is produced by VISEXP. A fisheye magnifies the central part of the display. Notice that the node-link diagrams in this paper are produced by neato, an open-source graph layout program provided by AT&T. It relies on the force-directed layout algorithm of Kamada and Kawai (Kamada & Kawai 1989).

various tasks on the equivalent matrix-based representation. Only tasks related to finding a path in the network appear to be performed more easily on a node-link representation, as long as it remains readable.

Making sense of graphs

Making sense out of network data often depends on the ability to understand its underlying structure. Therefore, cluster detection has been an active topic of research for a long time. Many works have concentrated on data analysis techniques in order to aggregate the graph into its main components. From a different standpoint, Bertin (Bertin 1983) has shown that the discovery of the underlying structure of a graph can be achieved through successive permutations over the rows and columns of the grid representing it. This idea relies on the fact that the rows and columns of a matrix can be ordered according to a given criterion, which is another advantage of the matrix metaphor as ordering the vertices and links in a node-link diagram is not straightforward.

In VISEXP, we achieve clustering through successive permutations of rows and columns according to two generic algorithms (a hierarchical agglomerative algorithm and a partition-based algorithm). The reader may refer to (Berkhin 2002) for a good survey on clustering techniques. Other domain-specific algorithms can be fit in our system effortlessly *e.g.* algorithms tailored for constraint programming graphs. In the following, we will present our early experiments in making use of matrix-based visualizations with constraint programming graphs.

Visualizing the explanation network

As we saw above, explanations introduce dynamic relations between variables and constraints that depend both on the constraint network itself and the resolution dynamics of the solver. In order to illustrate the interest of explanations for providing new information about resolution, we introduce three different types of graphs that are provided by VISEXP.

Graph parameters

Constraints c_i and c_j are connected in three different ways when solving a constraint problem:

- considering only the static structure information: c_i is linked to c_j each time c_i reduces a variable shared with c_j. Linking those two constraints expresses the fact that the activity of c_i will awake c_j in all future computations even if nothing new happens (no domain reductions). Graphs representing such a relation will be denoted `cc-static` in the future.
- considering dynamic relations arising through computation: c_i is linked to c_j each time c_i reduces a variable and c_j and c_i share any common variable. This relation states that all constraint c_j with their past effects are helping c_i adding information to the constraint store. More links are taken into account in this graph. Such a linking can be considered as an *a posteriori* explanation (this is how DNAC4 works for example(Bessière 1991)). Graphs representing such a relation will be denoted `cc-dynamic` in the future.
- considering explanation: c_i is linked to c_j each time c_i and c_j appear in the same explanation during computation. This relation represents the fact that c_i and c_j concurrently worked to provide new information to the solver. It represents some dynamic structure appearing during computation as constraints cooperate to solve the problem. Graphs representing such a relation will be denoted `cc-explain` in the future.

Notice that these graphs, being undirected, will result in symmetric matrices. Directed graphs could have been considered taking into account the way constraints cooperate: this is left for future work. Notice that all the above defined relations can easily be extended to variables.

Representing the dynamics of solving

As already mentioned, all links in those graphs are weighted with the number of times the relation is actually established throughout computation. This helps enhance the static structure with dynamic information pointing out *active* relations. More precisely, we keep a full history of activity within the graph. In this way, we can dynamically query the graph for links that are active within a user-controlled time range and compare the amount of activity between links in that range. The user may visualize the activity in the graph throughout the whole resolution process or in a smaller time range whose bounds and extent are interactively parameterized. By sweeping the time range from the beginning to the end of the history, the user may play back the resolution process and see which links are established, when, and how often. VISEXP also offers user-defined time slices. Simply put, a time slice is a time range between two events of interest. For instance, in our experiments, we were interested in activity between pairs of successive solutions. VISEXP computes the relevant time slices and allows the user to jump between successive time slices through direct interaction as well.

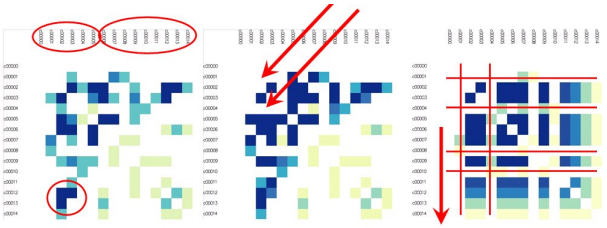

Figure 2: From left to right: `cc-static`, `cc-dynamic`, and `cc-explain` graphs for the computing of the first solution of our all-different problem

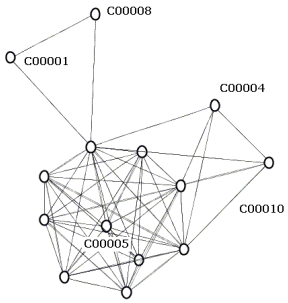

Figure 3: An all-different problem (first solution): a classical representation of the `cc-explain` graph

Experiments

In the following, we report various experiments with `VISEXP`. First, we show on a small problem how only `cc-explain` graphs allow both the visualization of common sense information about the resolution and the retrieval of new information. Second, we show how the animation features of `VISEXP` help understand constraint solving.

Retrieving known information

Let us consider a problem involving 13 variables $a_{i\in]3]}$, $b_{i\in]3]}$, d_1, d_3 whose domain is $[1..3]$ and $c_{i\in]5]}$ whose domain is $[1..5]$. Five `all-different` (Régin 1994) constraints are posted on these variables: three on respectively the a_is (constraint c_{00001}), the b_is (constraint c_{00004}) and the c_is (constraint c_{00002}) and two others relating the different sets of variables, respectively on d_1,c_2 and d_3 (constraint c_{00003}) and on a_1, b_1, c_1, and d_1 (constraint c_{00005}).

We consider the search of the first solution of this problem. Constraint propagation is not powerful enough to exhibit a solution without any enumeration. Nine value assignments (considered here as constraints) need to be made in order to reach a solution. We will therefore report 14 (5 + 9) constraints on the following graphs (see the ellipses in figure 2).

What are we looking for ? The definition of the problem itself gives insight on how constraints should interact during solving: *eg.* sharing at least one variable, c_{00002} and c_{00003} should often interact; moreover, c_{00001} and c_{00004} should not have much impact on solving as they are really easy to satisfy; finally, c_{00002}, c_{00003}, and c_{00005} should be the *hard* part of the problem. Regarding search, as usual early choices (*i.e.* here with smaller constraint index) should have a deep impact on late choices as they are used to direct search.

Retrieving the information We report in figure 2 the three constraint-constraint graphs that can be obtained using the generic trace generated by our solver. The `cc-static` graph confirms strong links between c_{00002}, c_{00003}, and c_{00005}. But, it also shows misleading relations: constraints c_{00012} (enumeration constraint $c_2 = 2$) and c_{00013} (enumeration constraint $c_3 = 1$) are strongly related to c_{00002} and c_{00003} (the circled part of the graph on the bottom left). The `cc-dynamic` graph provides some more information: constraints c_{00004} and c_{00001} are unrelated to other `all-different` constraints ((see the arrows)) as expected. However, the expected relation between enumeration constraints is still not apparent.

Only the `cc-explain` graph gives the full information: links between the `all-different` constraints, and more importantly the strong impact of early enumeration constraints (c_{00009}) compared to late ones (c_{00014}) (downward arrow). Moreover, some new information appear: obviously enumeration constraint c_{00008} (and similarly constraint c_{00010} – see the symmetric lines) had no impact in search (it had almost no subsequent relations with other constraints).

Figure 3 shows a classical node-link representation of the `cc-explain` graph. In this representation, some of the structure information can be retrieved (the central role of c_{00005} appears and the peripheral role of constraints c_{00001}, c_{00004}, c_{00008}, and c_{00010}). However, information about late impact of early decisions (the solver dynamics) cannot be visualized.

Retrieving non trivial information

Let us now consider a set of one hundred variables $x_{i\in]100]}$ whose domain is $[1..100]$. A set of 99 constraints is posted: $\forall i \in]99], x_i < x_{i+1}$. Arc-consistency enforcement on this problem is sufficient to lead to a solution. This is therefore a purely (long) propagation problem for sorting the variables. Figure 4 gives the three different graphs that can be drawn from the trace of the resolution of this problem.

What do we see ? As expected, the `cc-static` graph is of no use as it only gives what we already know: constraints are only statically related to their preceding and following constraints in the posting sequence. Notice that the `cc-dynamic` graph does not help much more. However, the `cc-explain` graph gives much more information about what happened during propagation: indeed, we find again the strong interaction between close (in the posting sequence) constraint, but we also see that all constraints are related to each other. Moreover, we can see that the farther the constraints from one another in the posting sequence, the less they interact (but they still interact). More-

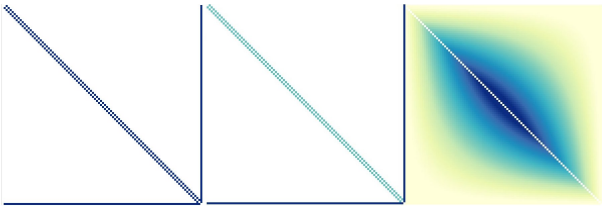

Figure 4: From left to right: `cc-static`, `cc-dynamic`, and `cc-explain` graphs for the propagation phase of our sorting problem. Constraints are ordered in the posting sequence order from right to left and from top to bottom.

over, the *middle* constraints seem to interact more often than *peripheral* constraints: see the darker part in the center of the `cc-explain` graph.

What do we learn ? Those observations on the `cc-explain` graph help us understand why the propagation takes so long in such a problem: all constraints need to be propagated several times as they all have both a short-term and a long-term impact during the resolution. Moreover, we illustrate here the interest of the visualization of the explanation network *wrt* classical analysis of the constraint graph which is not able to provide any information. However, one question remains: why *middle* constraints appear to be so active ? For that, we need to observe the dynamics of propagation.

Resolution dynamics

One of the key feature of `VISEXP` is that it makes it possible to visualize the dynamics of search and propagation. Animations are not easy to render on a printed paper. However, Figure 5 shows 7 time slices from the `cc-explain` graph reported in figure 4 for our sorting problem.

This visualization of the dynamics of the propagation phase on the sorted problem clearly shows that, in this problem, two phases are clearly identified. This illustrates the way our solver works. The first constraint to be propagated is the last posted one (constraints are not propagated upon posting but all at once) inducing a chain of modifications of the upper bounds of the variables through the awakening of all the other constraints. Then, the remaining constraints are propagated each modifying the lower bound of one variable. The two phases showed in figure 5 illustrate those two waves of modifications. Our animation is invaluable to illustrate such an intimate behavior of the solver as it gives information about the way propagation works.

Moreover, this information provides an explanation of the fact that *middle* constraints appear to have more activity than others. This is due to the fact that those constraints provide propagation both on upper and lower bounds of the variables whereas *peripheral* constraints impact only one bound of the variables.

Related work

Classical constraint solver embedded visualization tools (such as Explorer in Oz (Schulte 1997) or Visual Search Tree in Chip (Dincbas *et al.* 1988)) use the search tree of a constraint problem as their central metaphor. Exploration and visualization of the search tree are user-driven and interactive. `CLPGUI` (Fages 2002) allows a 3D manipulation of the search tree and provides some tools to visualize domains.

However, search tree based tools cannot provide information about propagation and dynamic relations between constraints as they are strongly related to the underlying logic of computation (observation points are limited to choice points in the search). Moreover, search tree visualisation tools are misled by the branching schemes used in constraint solving that may not really use the underlying structure of the problem therefore linking (in sequence) unrelated decisions.

`VISEXP` is the first tool that allows the visualization of dynamic information that do not consider the order in which decisions are taken, thus allowing a complete view of the problem.

Conclusion and further work

In this paper, we introduced `VISEXP`: a new visualization tool well suited for exploring relations between constraints and variables that uses the explanation network provided by a constraint solver. We showed that visualizing the explanation network provided much more information about how search is performed than classical representation. Moreover, we illustrated the animation feature of `VISEXP` that provides invaluable tools for visualizing the dynamics of constraint solvers.

`VISEXP` is only a first step in new visualization tools for constraint programming. We are currently investigating the extension of our tool in various directions, including: considering at the same time constraints and variables in the same representation in order to have a complete view of the solved problem; providing state of the art automatic clustering algorithms in order to bring closer related constraints and variables; visualizing and analyzing large scale problems. Moreover, we strongly believe in the capability of such a tool to provide debugging and dynamic analysis tools for constraint solvers.

Finally, we plan in a near future to use `VISEXP` to analyse the performance of explanation-based search algorithms (such as `decision-repair` (Jussien & Lhomme 2002)) in order to explain their behavior. The idea here is to provide new techniques exploiting the explanation network that could lead to new powerful algorithms.

References

Amilhastre, J.; Fargier, H.; and Marquis, P. 2002. Consistency restoration and explanations in dynamic csps - application to configuration. *Artificial Intelligence* 135(2002):199–234.

Berkhin, P. 2002. *Survey of clustering data mining techniques*. San Jose, USA: Accrue Software, Inc.

Bertin, J. 1983. *Semiology of Graphics: Diagrams, Networks, Maps.* University of Wisconsin Press.

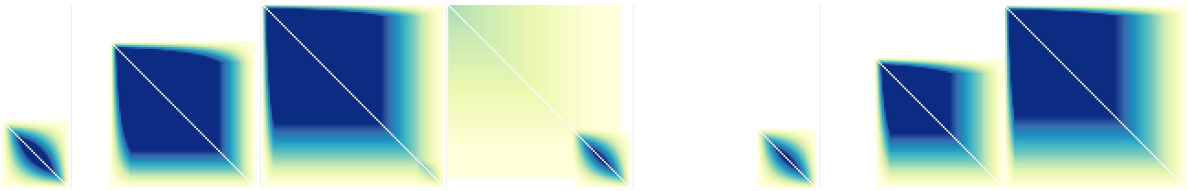

Figure 5: Visualizing the dynamics of propagation in our sorted problem. From left to right: a first slice, in the middle of the first phase, at in the end of the first phase, changing phases, at the beginning of the first phase, in the middle of the second phase, and the last slice.

Bessière, C. 1991. Arc consistency in dynamic constraint satisfaction problems. In *Proceedings AAAI'91*.

Card, S.; Mackinlay, J.; and Shneiderman, B. 1999. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann. chapter Dynamic Queries, 235–261.

Carpendale, M. S. T., and Montagnese, C. 2001. A framework for unifying presentation space. In *Proceedings of the 14th annual ACM symposium on User inter-face software and technology (UIST'01)*, 61–70.

Debruyne, R.; Ferrand, G.; Jussien, N.; Lesaint, W.; Ouis, S.; and Tessier, A. 2003. Correctness of constraint retraction algorithms. In *FLAIRS'03: Sixteenth international Florida Artificial Intelligence Research Society conference*, 172–176. St. Augustine, Florida, USA: AAAI press.

Deransart, P.; Ducassé, M.; and Langevine, L. 2002. A generic trace model for finite domain solvers. In *CP02 Workshop on User-Interaction in Constraint Satisfaction (UICS'02)*.

Dincbas, M.; Van Hentenryck, P.; Simonis, H.; Aggoun, A.; and Herold, A. 1988. The CHIP System: Constraint Handling in Prolog. In *9th International Conference on Automated Deduction*. Argonne: Springer.

Doyle, J. 1979. A truth maintenance system. *Artificial Intelligence* 12:231–272.

Fages, F. 2002. CLPGUI: a generic graphical user interface for constraint logic programming over finite domains. In *ICLP'02 12th Workshop on Logic Programming Environments (WLPE'02)*.

Fekete, J.-D., and Plaisant, C. 1999. Excentric labeling: Dynamic neighborhood labeling for data visualization. In *Proceedings of the International Conference on Human Factors in Computing Systems (CHI 99)*, 512–519. ACM.

Freuder, E. C.; Likitvivatanavong, C.; and Wallace, R. J. 2001. Deriving explanations and implications for constraint satisfaction problems. In *Principles and Practice of Constraint Programming (CP 2001)*, LNCS, 585–589.

Ginsberg, M. 1993. Dynamic backtracking. *Journal of Artificial Intelligence Research* 1:25–46.

Junker, U. 2001. QUICKXPLAIN: Conflict detection for arbitrary constraint propagation algorithms. In *IJCAI'01 Workshop on Modelling and Solving problems with constraints*.

Jussien, N., and Barichard, V. 2000. The PaLM system: explanation-based constraint programming. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP 2000*, 118–133.

Jussien, N., and Lhomme, O. 2002. Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence* 139(1):21–45.

Jussien, N.; Debruyne, R.; and Boizumault, P. 2000. Maintaining arc-consistency within dynamic backtracking. In *Principles and Practice of Constraint Programming (CP 2000)*, LNCS 1894 , 249–261. Singapore: Springer-Verlag.

Jussien, N. 2003. The versatility of using explanations within constraint programming. Research Report 03-04-INFO, École des Mines de Nantes, Nantes, France.

Kamada, T., and Kawai, S. 1989. An algorithm for drawing general undirected graphs. *Information Processing Letters* 31(1):7–15.

Mohr, R., and Henderson, T. C. 1986. Arc and path consistency revisited. *Artificial Intelligence* 28:225–233.

O'Callaghan, B.; Freuder, E. C.; and O'Sullivan, B. 2003. Useful explanations. In *Principles and Practice of Constraint Programming (CP 2003)*, LNCS, 988.

Ouis, S.; Jussien, N.; and Boizumault, P. 2003. k-relevant explanations for constraint programming. In *FLAIRS'03: Sixteenth international Florida Artificial Intelligence Research Society conference*, 192–196. St. Augustine, Florida, USA: AAAI press.

Pu, P., and Lalanne, D. 2000. Interactive problem solving via algorithm visualization. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2000)*, 145–154.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in CSPs. In *AAAI 94, Twelth National Conference on Artificial Intelligence*, 362–367.

Schiex, T., and Verfaillie, G. 1994. Nogood Recording for Static and Dynamic Constraint Satisfaction Problems. *International Journal of Artificial Intelligence Tools* 3(2):187–207.

Schulte, C. 1997. Oz Explorer: A visual constraint programming tool. In Naish, L., ed., *Proceedings of the Fourteenth International Conference on Logic Programming (ICLP'97)*, 286–300. The MIT Press.