UDC 004.4:37

PROPAGATION-BASED CONSTRAINT SOLVER IN IMS

Igor Ol. Blynov

Kherson State University

Abstracts. *Article compiling the main ideas of creating propagation-based constraint solver, theoretical basis of constraint programming and its implementation in IMS (Insertion Modeling System)*

Keywords. *IMS, constraint programming, solver.*

1. Introduction

Constraint programming is a powerful method for solving combinatorial (optimization) problems, which has proven effective and efficient in a wide range of application areas.

CSPs. A combinatorial problem is modeled as a set of variables, representing the objects the problem deals with, and a set of constraints, representing the relationships among the objects. Such a combinatorial problem is called a Constraint Satisfaction Problem (CSP). The common case where the variables can only take values from a finite universe is called a finite domain constraint satisfaction problem. A constraint programming system implements variables and constraints and provides a solution procedure for CSPs, which tries to find an assignment to the variables that satisfies all of the constraints. Clearly, solving CSPs is NP-hard in general, as the satisfiability of Boolean formulas (SAT) is one instance.

Application areas. Many hard, real-world combinatorial problems lend themselves to modeling as constraint satisfaction or optimization problems. The Handbook of Constraint Programming (Rossi et al., 2006) lists example applications in the areas of scheduling and planning, vehicle routing, configuration, networks (such as power or pipeline networks), and bioinformatics. Further application areas include computational linguistics (for example Duchier, 1999), as well as verification (Yuan et al., 2006) and optimization (van Beek and Wilken, 2001) of computer programs.

Constraint solvers. The success of constraint programming as a field is due to the availability of effective and efficient solution procedures that can solve these practical problems. This dissertation concentrates on finite-domain constraint programming, implemented in a propagation-based constraint solver, based on exhaustive search. This class of solvers has been successful because of its best-of-several worlds approach. They combine classic AI search methods with advanced implementation techniques from the Programming Languages community and efficient algorithms from Operations Research. Furthermore, the Constraint Programming community has identified global constraints as an important tool to make the structure of constraint problems explicit and achieve strong propagation. Dedicated propagation algorithms for many different global constraints are available.

Propagation-based constraint solving. At the heart of a propagation-based constraint solver, propagators realize the constraints of a CSP by pruning the variabledomains. A propagator removes values from variable domains that cannot be part ofany solution of its constraint. Propagators for particular constraints are usually implemented as specialized algorithms. The constraint solver computes a fixed pointof all propagators, maximizing the amount of inference they can contribute. It thensplits the problem and solves the resulting smaller problems recursively.

This process of inference is called **constraint propagation.** As the main inference method in constraint programming systems, constraint propagation infers that certain values cannot be part of

certain variable domains any more because they violate some constraint. The entities that perform constraint propagation are called propagators.

Constraint satisfaction problems are modeled with respect to a finite set of variables X and a finite set of values V . We typically write variables as x, y, z ∈ X, and refer to values as v, w ∈ V .

2. Assignments and constraints

A solution of a constraint satisfaction problem must assign a single value to each variable. A constraint restricts which assignments of values to variables are allowed. The following definition captures assignments and constraints.

Definition 1 An assignment a is a function mapping variables to values. The set of all assignments is Asn := X → V . A constraint c is a set of assignments, c ∈ Con := P (Asn) = P (X → V) (we write P (S) for the power set of S). It corresponds to a relation over the variables in X. Any assignment a Є c is a solution of c.

Guido Tack (see [1]) bases constraints on full assignments, defined for all variables in X. However, for typical constraints, only a subset vars(c) of the variables is significant; the constraint is the full relation for all x ∉ vars(c). More formally, a constraint c is the full relation for a variable x if and only $\forall v \in V, \forall a \in c : a[v/x] \in c$, where a[v/x] is the assignment a′ where

a′(x) = v and a′(y) = a(y) for all variables y ≠ x.

Consequently, the significant variables of c are defined as

$$\text{var}s(c) := \{x \in X \mid \exists v \in V; \exists a \in c : a[v/x] \notin c\}$$

Constraints are either written as sets of assignments, or just stated as mathematical expressions with the usual meaning. We use the notation · when we want to stress that we mean the constraint; for example, we write x < y to denote the constraint

a ϵ Asn a(x) < a(y) .

3. Domains and constraint satisfaction problems. Propagators

Constraints constitute one of the two crucial ingredients of constraint satisfaction problems. The other part is the initial set of values that each variable can take. For example in a Sudoku (as introduced in Section 2.1), each variable must take a value from the set {1, . . . , 9}. A mapping from variables to sets of possible values is a domain.

Definition 2 A domain d is a function mapping variables to sets of values, such that d(x) ⊆ V . The set of all domains is Dom := X → P (V). The set of values in d for a particular variable x, d(x), is called the variable domain of x. A domain d represents a set of assignments, a constraint, defined as

$$con(d) := \{a \in Asn \mid \forall x \in X : a(x) \in d(x)\}$$

Said that an assignment $a \in con(d)$ is licensed by d.

Definition 3 A constraint satisfaction problem (CSP) is a pair <d, C> of a domain d and a set of constraints C. The constraints C are interpreted as a conjunction of all c Є C and are thus equivalent to the constraint $\{a \in Asn \mid \forall c \in C : a \in c\}$. The solutions of a CSP d, C are the assignments licensed by d that satisfy all constraints in C, defined as $sol(< d, C >) := \{a \in con(d) \mid \forall c \in C : a \in c\}$.

Propagators. The basis of a propagation-based constraint solver is a search procedure, which systematically enumerates the assignments licensed by the domain d of a CSP <d, C> .

For each assignment, the solver uses a decision procedure for each constraint to determine whether the assignment is a solution of the CSP. Enumerating all assignments would be infeasible in practice, so in addition to the decision procedure, the solver employs a pruning procedure for each constraint, which may rule out assignments that are not solutions of the constraint.

These two tasks, the decision and the pruning procedure for a constraint, are realized by propagators. Each propagator induces a particular constraint. A propagator decides for a given assignment whether it satisfies the induced constraint, and it may prune those assignments from a domain that do not satisfy the constraint. Interleaving propagation and search yields a sound and complete solution procedure for the CSP. It is complete, because only assignments that are not solutions are pruned by the propagators, and all remaining assignments are enumerated. It is sound, because for each of the enumerated assignments, the propagators decide whether it is a solution. The formal definition of propagators author (see [1]) developed below captures the minimal properties that are required in order to get a sound and complete solver. In this way, this model differs from the definitions usually found in the literature. Furthermore, knowledge the characterization of propagators by unique induced constraints is novel. Authors define propagators in terms of domains. A propagator is a function p that takes a domain as its argument and returns a stronger domain, it may only prune assignments. If the original domain was an assigned domain {a}, the propagator either accepts it (p({a}) = {a}) or rejects it (p({a}) = 0), realizing the decision procedure for its constraint. In fact, each propagator induces a unique constraint, the set of assignments that it accepts. To make this setup work, we need one additional restriction. The decision procedure and the pruning procedure must be consistent: if the decision procedure accepts an assignment, the pruning procedure must never remove this assignment from any domain—this property is called soundness.

Definition 4 A propagator is a function $p \in Dom \to Dom$ that is:

- contracting: $p(d) \subseteq d$ for any domain d
- sound: for any domain $d \in Dom$ and any assignment $a \in Asn$, if $\{a\} \subseteq d$, then $p(\{a\}) \subseteq p(d)$

The set of all propagators is Prop. If a propagator p returns a strictly stronger domain ($p(d) \subset d$), we say that p prunes the domain d. The propagator p induces the constraint cp defined by the set of assignments accepted by p:

$$c_p := \{a \in Asn \mid p(\{a\}) = \{a\}\}$$

Soundness expresses exactly that the decision and the pruning procedure realized by a propagator are consistent. A direct consequence is that a propagator never removes assignments that satisfy its induced constraint.

Propagation problems. Propagators were defined as a refinement of constraints—each propagator induces one particular constraint, but in addition has an operational meaning, its pruning procedure. Its possible define the operational equivalent of a CSP, a propagation problem. Propagation problems realize all constraints of a CSP using propagators.

Definition 5 A propagation problem (PP) is a pair <d, P> of a domain *d* and a set of propagators P. The induced constraint satisfaction problem of a propagation problem <d, P> is the CSP $< d, \{c_p \mid p \in P\} >$. The solutions of a PP <d, P> are the solutions of the induced CSP, $sol(< d, P >) := sol(< d, \{c_p \mid p \in P\} >)$.

The set of solutions of a PP d, P can be defined equivalently as $sol(< d, P >) := \{a \in Asn \mid \forall p \in P : p(\{a\}) = \{a\}\}$, just applying the definitions of induced constraints and solutions of CSPs.

Existence of strongest and weakest propagators. Propagators combine a decision procedure with a pruning procedure. While the decision procedure determines the constraint a propagator induces, there is some liberty in the definition of the pruning, as long as it is sound. Thus, there are different propagators for the same constraint, and they can be arranged in a partial order according to their strength:

Definition 6 Let p1 and p2 be two propagators that induce the same constraint. Then p1 is stronger than p2 (written $p_1 \subseteq p_2$) if and only if for all domains d, $p_1(d) \subseteq p_2(d)$.

Propagation as a Transition System. A propagation-based solver interleaves constraint propagation and search, where constraint propagation means to prune the domain as much as possible using propagators, before search resorts to enumerating the assignments in the domain. Propagating as much as possible means, in the context of propagation problems, to compute a mutual fixed point of all propagators.

Transitions. Let <d, P> be a propagation problem. If there is a propagator $p \in P$ that can prune the domain d, that is, if $p(d) \subset d$, then applying p yields a new, simpler propagation problem, <p(d), P> . Soundness of p makes sure that the new problem has the same set of solutions as the original problem, $sol(< d, P >) = sol(< p(d), P >)$.

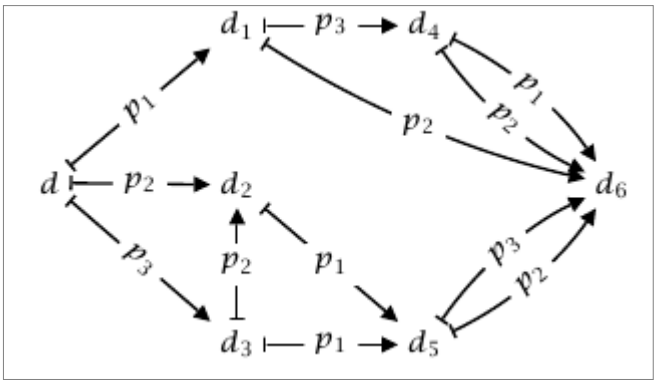
A propagation problem thus induces a transition system, where a transition is possible from a domain d to a domain $d' \subset d$ if there is a propagator $p \in P$ such that p(d) = d′ . Written such a transition

$$d \mid\!- p \rightarrow d'$$

Figure 1 shows how the transitions from a given initial domain d may look like.

Definition 7 Let d be a domain. A transition $d \mid\!- p \rightarrow d'$ with a propagator p to a domain d′ is possible if and only if d′ = p(d) and $d' \subset d$. The transition system of a propagation problem <d, P> consists of all the transitions that are possible with propagators $p \in P$, starting from d. A terminal domain, that is, a domain d such that there is no transition $d \mid\!- p \rightarrow p(d)$ for any propagator $p \in P$, is called stable.

Written $d \Rightarrow d'$ if there is a sequence of transitions that transforms d into a stable domain d′ . This sequence is empty, $d \Rightarrow d$, if d is stable.



Example (Transitions) Let d be a domain such that d(x) = d(y) = d(z) ={1, 2, 3, 4}, and assume three domain-complete propagators such that cp1 = x < y , cp2 = x + y = z , and cp3 = y < z . Then Figure 3.1 shows the transitions that are possible for the propagation problem d, {p1 , p2 , p3 } . The transition system has a unique stable domain d6 . The values of the domains are

d1 (x) = {1, 2, 3}
d1 (y) = {2, 3, 4}
d1 (z) = {1, 2, 3, 4}
d2 (x) = {1, 2, 3}
d2 (y) = {1, 2, 3}
d2 (z) = {2, 3, 4}
d3 (x) = {1, 2, 3, 4}
d3 (y) = {1, 2, 3}
d3 (z) = {2, 3, 4}
d4 (x) = {1, 2, 3}
d4 (y) = {2, 3}
d4 (z) = {3, 4}
d5 (x) = {1, 2}
d5 (y) = {2, 3}

d5 (z) = {2, 3, 4}
d6 (x) = {1, 2}
d6 (y) = {2, 3}
d6 (z) = {3, 4}

The transition system of a propagation problem is non-deterministic, as there are many possible chains of propagation that result in a stable domain.

Implementation of the above methods for creating Propagation-based constraint solver may by means of IMS - insertional modelling system, created at the Glushkov Institute of Cybernetics. (see [2]). Earlier in the article ICTERI 2011 conference (see [3]) was described by means of the implementation of constraint programming in IMS. This article briefly outline the main points of the general theory of IMS and a prototype implementation of the propagation-based constraint solver. The main theory of IMS is based on theory of agents and enviroment. Its implies existence of insertion procedure, which can change the state of enviroment by acting it with agent.The main different in implementation of constraint programming and propagation-based constraint solver is in insertion procedure. In propagation-based constraint solver insertion procedures should interleaves two action -
propagation and search, so it could be implement in two different agents with rewritten insertion procedure which should use it together in controlled parallel interaction.

REFERENCES

1. Guido Tak. Constraint Propagation. Models, Techniques, Implementation. Saarbrucken, 2009.
2. Alexander Letichevsky, Olexander Letichevskyi, Vladimir Peschanenko, Igor Blinov and Dmitriy Klionov: (en) Insertion Modeling System And Constraint Programming. In: Ermolayev, V. et al. (eds.) Proc. 7-th Int. Conf. ICTERI 2011, Kherson, Ukraine, May 4-7, 2011, CEUR-WS.org/Vol-716, ISSN 1613-0073, <51-64>, online CEUR-WS.org/Vol-716/
3. D.R. Gilbert, A.A. Letichevsky: A universal interpreter for nondeterministic concurrent programming languages.In M. Gabbrielli (eds.), Fifth Compulog network area meeting on language design and semantic analysis methods (1996).
4. A. Letichevsky and D. Gilbert: A general theory of action languages.Cybernetics and System Analyses, vol. 1, 16–36 (1998).
5. A. Letichevsky and D. Gilbert: A Model for Interaction of Agents and Environments. In D. Bert, C. Choppy, P. Moses, (eds.). Recent Trends in Algebraic Development Techniques. LNCS, vol. 1827, pp.311–328. Springer (1999).
6. A. Letichevsky: Algebra of behavior transformations and its applications. In V.B.Kudryavtsev and I.G.Rosenberg (eds). Structural theory of Automata, Semigroups, and Universal Algebra, NATO Science Series II. Mathematics, Physics and Chemistry, vol. 207, pp. 241–272. Springer (2005).
7. G. Martin, and B.Selic, (eds.). UML for Real: Design of Embedded Real-Time Systems. Kluwer Academic Publishers. Amsterdam (2003).
8. A. Letichevsky, J. Kapitonova, A. Letichevsky Jr., V. Volkov, S. Baranov,V.Kotlyarov, T. Weigert: Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications. Computer Networks, vol. 47, 662–675 (2005).
9. J. Kapitonova, A. Letichevsky, V. Volkov, and T.Weigert: Validation of Embedded Systems. In R. Zurawski, (eds.). The Embedded Systems Handbook, CRC Press, Miami (2005).
10. A. Letichevsky, J. Kapitonova, V. Volkov, A. Letichevsky, jr., S. Baranov, V. Kotlyarov, and T. Weigert: System Specification with Basic Protocols. Cybernetics and System Analyses, vol. 4, 479–493 (2005).