Temporal Constraints in Planning: Free or not Free ?

Thierry Vidal
thierry@laas.fr

Malik Ghallab
malik@laas.fr

LAAS-CNRS - 7, avenue du Colonel-Roche, 31077 Toulouse, France

Abstract

In temporal planning, one needs an explicit representation of time. Our system IxTeT relies on a time-point based constraint graph, and handles symbolic as well as numerical temporal constraints (convex durations and dates). Classical CSP techniques allows to check the consistency of such a graph in polynomial-time.

We also need to distinguish between *controllable* numerical constraints, that can be freely instantiated, and *contingent* ones, that will take random values at execution time. Classical CSP techniques are no longer sufficient, since the consistency of the constraint graph only means that the underlying plan may possibly succeed. Being certain of its success requires a graph that remains consistent whatever values are taken by the contingent constraints. Our method relies upon the translation of the initial graph into a dual one, requiring the definition of compound constraints. We propose a sound and complete propagation, with complexity comparable to the classical Temporal CSPs.

1 . Introduction

Planning in the partial plan space consists in modifying and extending a partial plan in order to meet prespecified goals. A partial plan evolves by propagating new precedence or unification constraints, or by adding new tasks as planning operators. Tasks are defined over some interval of time, and expected events are also predicted to occur at some given time. To the former are attached durations, and to the latter dates, both generally not precisely known, and new tasks together with their effects are added while planning with respect to tasks and events already existing in the plan.

A natural way to deal with temporal data is to refer to an explicit representation of time, and to manage it through a temporal constraints network, separated from the planning search algorithms. Each time a new task or event is added to the partial plan, corresponding temporal constraints are propagated. Specific algorithms for constraint propagation are used to check the consistency of the network, and update it for allowing sound and complete answers to temporal queries.

We first have to distinguish between numerical constraints (durations and dates) and symbolic ones. The latter are either precedence constraints of the time-point algebra, or primitive relations of the interval algebra calculus [Allen84] and their disjonctions permitted by the restricted interval algebra [Vilain86, Nebel94] that can be translated into time-point algebra relations and propagated in polynomial time. Numerical constraints are bounds on the difference between two time-points. We restrict ourself to the non-disjunctive polynomial case (the Simple Temporal Problems [Dechter90]).

Furthermore, we need to distinguish between two kinds of numerical constraints: the controllable ones, i.e. those that can be freely instantiated, and the contingent ones, i.e. those that take random values corresponding to an uncertainty in the application. Occurrence times of expected

events, or the duration of some tasks, are typical contingent constraints. Classical CSP techniques are no longer sufficient, since the consistency of the constraint graph only means that the underlying plan may possibly succeed. Being certain of its success requires a graph that remains consistent whatever values are taken by the contingent constraints. The graph will then be said to be controllable, and the corresponding plan assumed to be sure.

The representation issues are given in section 2, then in section 3 we develop this special kind of Temporal CSP, and the constraint assignments that are related to it. We propose a dual CSP that we call the Decision Graph, in which original constraints are translated into meta-constraints, and for which sound and complete propagation can be processed, with a complexity comparable to the classical Temporal CSPs. We will describe how a constraint insertion while planning can be processed using this paradigm, and what are the benefits for the plan execution control system. Then, section 4 will conclude surveying some related works.

2. Representation and Temporal Management

Temporal CSPs [Dechter90] are dedicated to temporal problems in which the variables are time-points, ranging over convex intervals of $\mathbb{R}$. Binary constraints are bounds on the possible durations between two time-points. Dates as well as durations are sets of possible values represented through temporal intervals. The STP (Simple Temporal Problems) restriction applies when those temporal windows are only non-disjunctive intervals. Namely, we get for example the duration constraint between the time-points i and j as $C_{ij}=[a,b]$, with $a \leq b$.

This restriction suits well the notion of imprecise numerical temporal data in planning and allows to get a polynomial time complete propagation algorithm, namely a 3-consistency (or path-consistency) algorithm PC-2 [Mackworth85] that consists in incrementally propagating any new constraint to the overall network. Restriction applies to triples of time-points: C_{ij} is modified through $(C_{ik} + C_{kj}) \cap C_{ij}$, + being here the composition operation of two constraints, and $\cap$ the intersection. An inconsistency is detected as soon as we get $C_{ij}=[a,b]$ such that $a > b$. Thanks to algebraic properties, this local consistency enforcement entails global consistency, which means that we get an inconsistency if and only if there is no solution to the CSP. Moreover, path-consistency allows to get the minimal network, which is the complete network where the constraints domains are restricted to the only values that belong to a solution.

Our planning system IxTeT uses a reified logic formalism: we explicitly represent and reason upon time in a separated graph-based constraint manager. This temporal manager is based on the time-points algebra [Vilain86], and deals with a directed graph in which nodes are the time-points and edges are simple precedence relations between time-points. Those are called symbolic constraints. We can also use this graph to represent numerical constraints, taking the STP formalism [Dechter90]. We get two CSP structures on a unique graph: let V be the set of time-points, R_s be the set of symbolic constraints (simple precedence relations), and R_n be the set of numerical constraints, then

- $\mathcal{S}=(V,R_s)$ is the symbolic CSP, for which linear time algorithms have been developed (relying on a maximal indexed spanning tree, see [Ghallab89]) for constraint addition and deletion through consistency-checking, and query answering.
- $\mathcal{N}=(V,R_n)$ is the numerical CSP, for which we use the path-consistency algorithm PC-2, that runs in $O(n^3)$, n being the number of time-points [Ghallab94]. Getting the minimal network at each addition step allows to:

1. check the consistency,
2. get any new symbolic precedence resulting from numerical propagation (defining a *strong* numerical constraint as being $C_{ij}=[a,b]$ such that $\boldsymbol{a}$ 0, which enforces a symbolic precedence between *i* and *j*),
3. get constant time queries,
4. find a solution instance in linear time, instantiating opportunistically the time-points while executing the plan. Each instantiation is propagated for maintaining the minimal network and the possible values of remaining variables.

To take advantage of the greater efficiency of symbolic constraints management, we maintain two structures. We restrict the numerical management to a numerical subgraph. We will in the following assume that $\mathcal{N}$ is such a structure, containing only the time-points concerned with numerical constraints (see [Ghallab94] for details). We can define a complete and sound interaction processes between $\mathcal{S}$ and $\mathcal{N}$. We will now see how we can take into account in $\mathcal{N}$ the distinction between contingent and free constraints.

Notice that $\mathcal{S}$ only relies on the reduced time-point algebra, not taking explicitly into account the relation ≠, that scarcely appears in planning applications. We manage it in a separate structure (see [Ghallab89&94]), with sound and complete interactions with $\mathcal{S}$ and $\mathcal{N}$.

3 . Controllability in Temporal Constraint Networks

3.1 Motivation

Let us now develop the notion of uncertainty in the numerical constraints that has been introduced earlier. A Contingent STP (CSTP) will be a STP where the following distinction is made:

« Definition 1 »

• A **free constraint** (referred as **Free**[(i)] in the following) $C_{ij}=[a,b]$ is a numerical constraint whose value can be freely assigned at execution time, which means that the propagation can reduce its domain.

• A **contingent constraint** (referred as **Ctg** in the following) $C_{ij}=[a,b]$ is a numerical constraint whose value is a random variable in the domain *[a,b]*, which thus should not be reduced during the propagation.

As we have made a difference between two kinds of constraints, we can also distinguish between two kinds of time-points:

« Definition 2 »

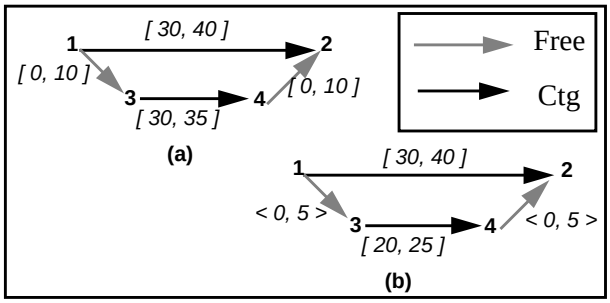
• the **decisional time-points** are those for which one freely assign a date (i.e. the beginning time-points of Ctg).

• the **random time-points** are those that may take an unpredictable date (i.e. ending time-points of Ctg, and expected events).

3.2 Illustration Through a Small Example

Figure 1(a) illustrates the example of two tasks, whose durations are contingent, and represented through the constraints C_{12} and C_{34}, the latter occurring «during» the former:

Figure 1. Example of contingent constraints:



i. notice that we use the term *free* instead of *controllable*, since it encompasses constraints that are not truly under the control of the execution monitoring system, but which domains can be freely reduced by propagation, like for example a precedence constraint between the ends of two tasks.

The above graph, when filtered through path-consistency, appears to be consistent in the sense of the classical STP paradigm, which means that there exists at least one labelling satisfying all the constraints, for example $\{C_{13}=0, C_{34}=30, C_{42}=10, C_{12}=40\}$. But C_{12} and C_{34} being contingent, they may take for example respectively the values 30 and 35, and then the set of constraints can no longer be satisfied. In fact, we have to satisfy $C_{13} + C_{34} + C_{42} \leq C_{12}$. Then, we can intuitively assume the following:

- The example is consistent iff there exist some values of C_{12} and C_{34} within allowed bounds such that $C_{13} + C_{34} + C_{42} \leq C_{12}$ (possible condition)
- The example is **controllable** iff for any value of C_{12} and C_{34} within allowed bounds we have $C_{13} + C_{34} + C_{42} \leq C_{12}$ (necessary condition).

A similar example in Figure 1(b) is controllable, but we also need to get values on the Free for which the whole set of constraints will remain controllable: Figure 1(b) informs us that one cannot wait more than 5 time units between the execution release of the two tasks, otherwise the plan may fail. Therefore, we may try to find a subinterval of a controllable constraint $C_{ij}=[a,b]$, let us call it the «necessary form» of the constraint $C_{ij}^{N}=[\mathbf{L,U}] \subseteq [a,b]$, such that

• any value within [L,U] leads to a necessarily consistent graph; a value in [a,b] outside of [L,U] is possibly but not necessarily consistent.

• the constraint network is consistent iff $L \leq U$ for all the Free.

3.3 Representation and Basic Reasoning Issues

In the following, a Ctg g_k will be denoted as beginning at point b_k and ending at point e_k, such that $b_k \leq e_k$ (≤ means here «is before»). In an incremental process, we will consider that at each step, we add a Ctg g_k together with the relations linking it with other Ctg already in the network.

« Definition 3 »

• a **MetaSymb** is a symbolic relation between two Ctg: it is defined by the set of the four symbolic constraints between the beginning and ending points of the two Ctg g_1 and g_2: $\{(b_1,b_2), (b_1,e_2), (e_1,b_2), (e_1,e_2)\}$.

We have completely characterized the set of MetaSymb. From the restricted time-point algebra (see section 2), 256 relations may be defined between any four points. But a lot of them are not permitted, either because some combinations, together with the already existing constraints $b_k \leq e_k$, may induce negative cycles, or because they are beforehand uncontrollable (for example, we can never make sure that $e_1 = e_2$ will be true at execution time, since they are both random variables). The admissible set of MetaSymb finally reaches 24 relations, that are drawn in Table 1, not considering the reverse relations from $\overline{R}_2$ to $\overline{R}_{12}$, such that if g_1 R_k g_2, then g_2 $\overline{R}_k$ g_1.

Composition and intersection laws can be easily listed in a similar exhaustive way, which will not be reported here. Notice that the relation R_{12} is a special case, as here b_2 equals $e_1$and therefore can no longer be considered as a decisional variable, but we prove that we can translate R_{12} into the relation R_2, replacing the Ctg g_2 by a virtual Ctg corresponding to $g_1 + g_2$.

Then, a MetaSymb between two Ctg g_1 and g_2 can be translated into a set of relations between their beginning points b_1 and b_2 (see Table 1, where D_i means the random duration of the Ctg g_i), which gives the controllable constraint between b_1 and b_2:

« Definition 4 »

• A **MetaNum** is the numerical relation between two Ctg g_1 and g_2 and is only defined by the Free between b_1 and b_2: $C_{b_1b_2}^{N}$.

We prove that this is a sufficient, sound and complete characterisation of a numerical con-

straint between two Ctg. This leads to the definition of the Decision Graph:

« Definition 5 » The **Decision Graph** of a CSTP is $\mathcal{D}$ = {**G, Ms, Mn**}, where

- **G** is the set of variables, i.e. the Ctg of the CSTP,
- **Ms** is the set of the MetaSymb relating Ctg,
- **Mn** is the set of the MetaNum relating Ctg.

In other words, {**G, Ms**}=$\mathcal{D}s$ is a symbolic CSP, and {**G, Mn**}=$\mathcal{D}n$ is a numerical CSP. We then prove the following:

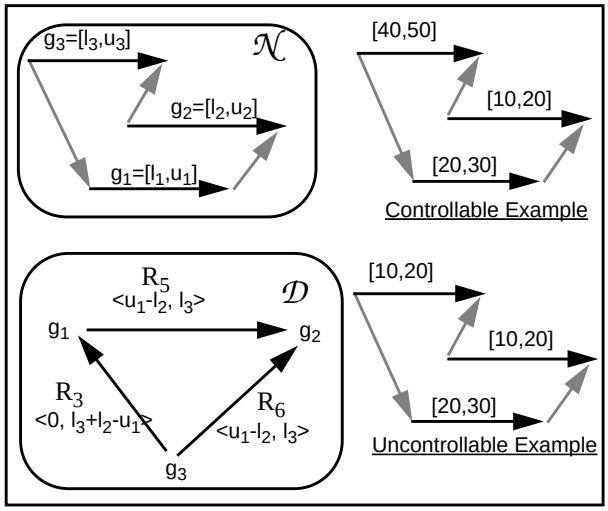
« Property »

- Checking the controllability of $\mathcal{N}$={$\mathbf{V_n}$, $\mathbf{R_n}$} is strictly equivalent to checking the consistency of $\mathcal{D}n$={**G, Mn**}.

Then, Table1 gives the MetaNum corresponding to each MetaSymb. We can also easily put forward the reverse correspondence: each modification of a MetaNum may modify the corresponding MetaSymb: for example, if L in $C_{b_1b_2}^{N}$ becomes positive, we infer $b_1 \leq b_2$, and the MetaSymb may be changed from R_5 to R_7. This defines the interaction processes between $\mathcal{D}s$ and $\mathcal{D}n$. Moreover, the restriction of a MetaNum leads to complement the MetaSymb with a new elementary symbolic constraint, that will have to be added in $\mathcal{S}$ as well. This allows to take into account symbolic constraints resulting from numerical propagation (see section 2).

To end with this section, Figure 2 gives an example of propagation step in $\mathcal{D}n$: 3 Ctg and the set of precedence constraints linking them in $\mathcal{N}$, are reported into $\mathcal{D}$, in which a propagation step is processed: we represent both MetaSymb and propagated MetaNum. The consistency of $\mathcal{D}n$, and thus the controllability of $\mathcal{N}$, is then checked as soon as $u_1 \leq l_2 + l_3$. We give two numerical examples in the initial graph, one that is controllable following this condition, and the other not.

Figure 2. Controllability checking through $\mathcal{D}$:



3.4 Reasoning Issues in Planning and Plan Execution

Let us now quickly sketch what is done in an incremental planning process at each insertion step. The insertion of a task can be translated into the addition of a Ctg g in $\mathcal{N}$ and of precedence constraints (relating the ending points of g to other numerical points), in $\mathcal{S}$ and $\mathcal{N}$ (through [0,+∞[intervals). We get the following steps going through the Decision Graph:

(1) Report the Ctg from $\mathcal{N}$ in $\mathcal{D}n$ and $\mathcal{D}s$, as a new node.

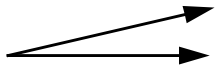
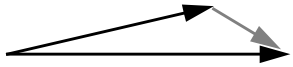
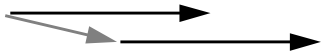
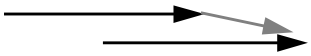
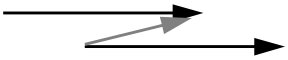
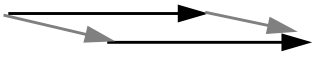
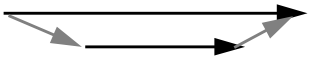
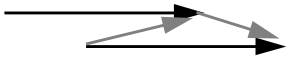
(2) Translate the precedence constraints into MetaSymb in $\mathcal{D}s$ between g and other Ctg already in $\mathcal{D}s$.

(3) Compute the corresponding MetaNum in $\mathcal{D}n$, using the correspondence Table 1.

(4) Propagate into $\mathcal{D}n$. Each time a MetaNum is modified, check if the underlying MetaSymb is modified, again going through the correspondence Table 1 (5). If so, identify the new mere symbolic constraints in the MetaSymb, and add them in $\mathcal{S}$ (6).

This defines the interaction processes between $\mathcal{S}$ and $\mathcal{N}$, and $\mathcal{D}$. The complexity of the whole is

Table 1. MetaSymb and MetaNum:

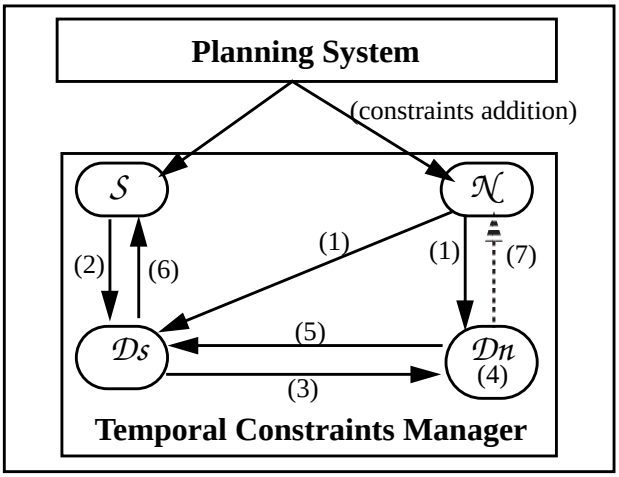
MetaSymb Identifier	representation of the MetaSymb	relations between b_1 and b_2	MetaNum: $C_{b_1b_2}{}^N$
R_0	b_1 —$D_1=[l_1,u_1]$→ e_1 b_2 —$D_2=[l_2,u_2]$→ e_2	$\varnothing$	< -oo, +oo>
R_1		$b_1 = b_2$	< 0, 0 >
R_2		$b_1 = b_2$ $b_1 + D_1 \le b_2 + D_2$	< max (u_1-l_2, 0), 0 >
R_3		$b_1 \le b_2$	< 0, +oo >
R_4		$b_1 + D_1 \le b_2$	< u_1, +oo >
R_5		$b_1 + D_1 \le b_2 + D_2$	< u_1-l_2, +oo >
R_6		$b_2 \le b_1 + D_1$	< -oo, l_1 >
R_7		$b_1 + D_1 \le b_2 + D_2$ $b_1 \le b_2$	< max (u_1-l_2, 0), +oo >
R_8		$b_1 \le b_2$ $b_2 + D_2 \le b_1 + D_1$	< 0, l_1-u_2 >
R_9		$b_1 \le b_2$ $b_2 \le b_1 + D_1$	< 0, l_1 >
R_{10}		$b_2 \le b_1 + D_1$ $b_1 + D_1 \le b_2 + D_2$	< u_1-l_2, l_1 >
R_{11}		$b_1 \le b_2$ $b_2 \le b_1 + D_1$ $b_1 + D_1 \le b_2 + D_2$	< max (u_1-l_2, 0), l_1 >
R_{12}		$b_1 + D_1 = b_2$	translated into relation R_2

shown to be comparable to the $O(m^3)$ one in classical Temporal CSPs (m being the number of time-points in the graph), provided that the number of symbolic constraints added at each step is bounded, which is true in planning applications (see [Ghallab94]). Some easy enhancements, that will not be reported here, allow to make a priori consistency checking, before propagating, which is much useful in planning (we can backtrack on an uncontrollable insertion choice without having the graph modified).

The behaviour at the execution step will be the following: every new expected event arriving to the control system will update the Decision Graph, maintaining what we may call the minimal controllable graph, from which we will infer in $\mathcal{N}$ the values that can be assigned to the forthcoming decisional variables (7). This therefore can be compared to classical decision-making problems under uncertainty.

All the steps from (1) to (6) at planning level, and (7) at execution level, are summarized in Figure 3:

Figure 3. Overall Process:



4. Conclusion through Related Work

Our work can be compared to the classical critical path methods in task networks[Malcolm59], especially in the tasks-constraints form, and when considering tasks durations as random variables. The task durations are then by nature contingent, and the margins represent in a certain sense the amount of controllability on the starting events of the tasks. Some more recent works, based upon fuzzy logic, takes into account the uncertainty on some task durations, like [Dorn93], in which a schedule is computed according to average expected values and fuzzy rules, then dynamically modified in a repair-based perspective if some tasks may possibly overlap, according to their fuzzy temporal distribution. Taking explicitly into account free constraints as well is only touched on in a further work [Dorn94], in which one forbids the restriction of an uncontrollable constraint, but without weighting the consequences in terms of overall consistency. The most advanced work upon this strict distinction between what is controllable and what is not is certainly the one by [Dubois93], in which a scheduling problem is formalised through fuzzy temporal constraints, distinguishing between possible values for controllable constraints and necessary values for contingent constraints[(i)].

But all these techniques are dedicated to scheduling, where task intervals are only partially ordered, which means that the temporal satisfaction of a schedule is only related to the general requirement of minimizing the overall schedule duration, or with respect to due-dates. Considering planning applications leads to take into account more general temporal relationships between overlapping intervals. Our contribution is then to show that we can keep the basic scheduling concept of considering start times of tasks as decision variables, and to extend it to planning problems, through the generalisation of simple tasks precedence constraints to any type of constraints (what we call the symbolic meta-constraints). We then get a complete characterisation of the numerical meta-

i. The same team went further, making a most relevant work [Fargier95] that generalises the distinction free / contingent in general CSPs.

constraints relating those decisional variables, thus making it possible to apply classical constraints propagation used in planning. Those meta-structures exist together with the classical time-points graph structures that are necessary at execution time, as we need to hold all the events (beginning and ending of tasks, instantaneous events).

Lastly, it is important to notice that the different CSP structures (symbolic, numerical, meta-symbolic, meta-numerical) are based upon a unique real overall time-points graph, allowing flexible interactions between them, especially enforcing numerical-to-symbolic inferences that are much useful in planning (see also [Ghallab94]).

We are at the present time working on the extension of the formalism to more general cases, the addition of free numerical constraints, or of weak contingent constraints (as [-10,10] for example).

The next step is now to characterise in more details what uncertainty in temporal constraints means, what a planning system needs to handle and reason upon. Distinguishing between contingent and free constraints is not sufficient, one wants to also refer to preference, priority or utility concepts, to classify constraints upon their degree of confidence, relevance or violation-cost for example, leading to a more flexible way of defining the degree of feasibility of a plan in accordance with the degree of uncertainty weighting on the constraints. Fuzzy-set theory, used for example by [Dubois93] may be a much suitable tool, among others, to model it. We also claim that enhanced relaxation techniques will also earn some gain from this research area.

References

[Allen84] James F. Allen - *Towards a General Theory of Action and Time* - Artificial Intelligence, 23, 1984 - pp. 123-154.

[Dechter90] Rina Dechter, Itay Meiri & Judea Pearl - *Temporal Constraint Networks* - Artificial Intelligence, 49, 1991 - pp. 1-95.

[Dorn93] J.Dorn, R.Kerr & G.Thalhammer - *Reactive Scheduling in Fuzzy-Temporal Framework* - Tech Report CD-TR 93/55, Technische Universität Wien, 1993.

[Dorn94] J.Dorn - *Hybrid Temporal Reasoning* - In Proc. ECAI, Amsterdam, Netherlands, 1994 - pp. 625-629.

[Dubois93] D.Dubois, H.Fargier & H.Prade - *The Use of Fuzzy Constraints in Job-Shop Scheduling* - In Proc. IJCAI-93 Workshop on Knowledge-Based Planning, Scheduling and Control. Chambery(France), 1993.

[Fargier95] H.Fargier, J.Lang & T.Schiex - *Mixed Constraint Satisfaction: a Framework for Decision Problems under Incomplete Knowledge* - submitted to IJCAI‘95.

[Ghallab89] Malik Ghallab & Amine Mounir-Alaoui - *Managing Efficiently Temporal Relations Through Indexed Spanning Trees* - In Proc. IJCAI, 1989, Detroit - pp. 1297-1303.

[Ghallab94] M.Ghallab & T.Vidal - *Focusing on a Sub-Graph for Managing Efficiently Numerical Temporal Constraints* - In Proc. FLAIRS, Melbourne Beach (FL), 1995 (to appear).

[Malcolm59] D.G.Malcolm, J.H.Clark, & al. - *Application of a Technique for Research and Development Program Evaluation* - Operations Research, 7(5), 1959.

[Mackworth85] A.K. Mackworth & E.C. Freuder - *The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems* - Artificial Intelligence, 25(1), 1985 - pp.65-74.

[Vilain86] M.Vilain & H.A. Kautz - *Constraint Propagation Algorithms for Temporal Reasoning* - In Proc. AAAI, Aug 1986, Philadelphia - pp 377-382.