Consistency of Temporal Constraint Networks for Medical Monitoring : Binary or Nary Constraints ?

A. Liret[1], **N. Ramaux**[2], **F. Pachet**[3], **D. Fontaine**[2] and **M. Dojat**[4]

Abstract. This paper addresses a new method using nary linear constraints for scenario recognition in the context of on-line monitoring of patients hospitalized in intensive care units. This method is compared with the classical and binary constraints-based method. Our system, called *DéjaVu*, compares on the fly dynamic scenes which represent the real evolution of a patient, with predefined scenarios which represent an expected evolution of the patient. Both scenarios and scenes are represented as temporal networks of numerical constraints between instants (TCSP). The recognition task is performed by minimizing a particular simple temporal network (STP) which always has very few cycles. Classically the Floyd-Warshall algorithm is well adapted to the minimization of STP, and it is equivalent to path-consistency. Exploiting the topological characteristic of the STP, we propose a more efficient method, using arc-consistency. It is based upon a reformulation of the STP by a dual CSP on nary linear equality constraints. We prove that minimizing the primal STP is equivalent to making the dual CSP arc-consistent. We explain our method and give theoretical and practical results. Although the STP-to-CSP transformation is theoretically more expensive than the Floyd-Warshall algorithm, we show that, in practice, it leads to better results for networks with few cycles, such as the ones used for on-line medical scenario recognition.

1. INTRODUCTION

The problem of automatic monitoring of patients has been tackled by many recent works (see [1] for a review of recent systems). In [2], M. Dojat proposed a mechanism based upon artificial intelligence techniques, to perform this task without human intervention. The efficiency of this system, named *NéoGanesh*, is currently being improved by adding to it a scenario recognition module that should recognize on the fly some typical behaviors of the patient, in order to adapt its response (see the *DéjaVu* system [3]). The scenario recognition method handles two entities: some predefined scenarios designed off-line by a physician, and the dynamic clinical session, constructed from physiological measurements. It consists of comparing the session to a set of predefined scenarios and deciding which scenarios describe the real evolution of the patient. As it should be performed in real-time we have to use the most efficient algorithm to realize the comparison task.

1.1 Medical scenarios

Representing the scenarios requires the ability of representing time and events. Such a representation is usually done by using networks of temporal constraints. As we need to represent numerical constraints, we chose to use networks of numerical binary constraints between instants, called Temporal Constraint Satisfaction Problem (TCSP) [4, 5].

We recall that a TCSP is a graph of interval constraints between real variables. Every constraint is a union of separate intervals. In particular, when the constraints are defined by one continuous interval, the graph is called a Simple Temporal Problem (STP). A TCSP can be seen as a theoretical binary-CSP where the goal is to find the instants values such as the difference between instants belongs to the interval [5].

In our application domain, both session and scenarios are represented by a STP. Below is an example of a typical scenario ($S0_1$) used as data for recognition on the fly.

e1: (DisconnectionPatient=initiates), e2: (DisconnectionPatient=terminates),
e3: (Respiratory.normal=initiates), e4: (Respiratory.normal=terminates),
e5: (Respiratory.tachypnea=initiates), e6: (DisconnectionPatient=initiates),
e7: (DisconnectionPatient=terminates), e8: (Respiratory.tachypnea=terminates),
e9: (Respiratory.normal=initiates), e10: (Respiratory.normal=terminates),
e11: (RepiratoryRate.increasing=inèitiates), e12: (RespiratoryRate.increasing=terminates).

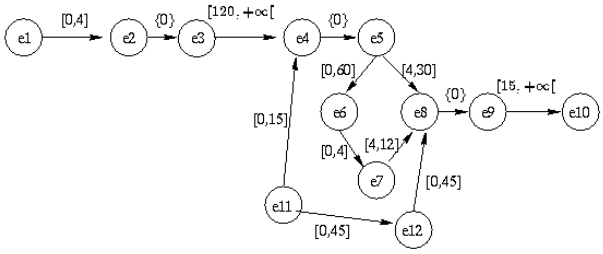


Figure 1. Medical scenario for a mechanically ventilated patient

This scenario is an excerpt from mechanical ventilation management. It indicates a progressive obstruction of the endotracheal tube followed by ventilation instabilities after suctioning (see [3] for more details).

1.2 Scenario recognition

The difficult problem of recognition has already received much attention [6, 7, 8] and more precisely in the range of real-time monitoring [9, 10, 11]. Classically, the method of checking the compatibility of a session and a predefined scenario, consists in merging the session with each predetermined scenario, and minimizing the "fusion graph". To obtain the fusion graph, we first compute the over-graphs of the session and each scenario, then we

[1] Université Pierre & Marie Curie, LIP6, 4,place Jussieu, BP 169, 75252 Paris cedex 05, France. Email: Anne.Liret@lip6.fr
[2] Université de Technologie de Compiègne, HeuDiaSyC UMR CNRS 6599, BP 20529, 60205 Compiègne Cedex, France. Email: Nicolas.Ramaux@hds.utc.fr
[3] SONY CSL-Paris. 6, rue Amyot, 75005 Paris, France. Email: pachet@csl.sony.fr
[4] Institut National de la Santé et de la Recherche Médicale, U438 - RMN Bioclinique, Centre Hospitalier Universitaire - Pavillon B, BP 217, 38043 Grenoble Cedex 9, France. Email: michel.dojat@ujf-grenoble.fr

perform the intersection of these over-graphs. The existence of one empty constraint in the minimal graph means that it is inconsistent, and then the session is incompatible with the scenario. The minimization of a temporal graph has already been studied in [5]. They proposed the Floyd-Warshall algorithm to perform the minimization of a STP and proved its equivalence with the algorithm of path-consistency (PC1). The complexity of Floyd-Warshall never exceeds $O(n^3)$ and no better algorithm in complexity has been found so far. Nevertheless Floyd-Warshall has a major drawback: it completes the graph, which may be an expensive hypothesis when the graph has a lot of nodes. Thus by creating universal constraints, it uses all of the information, initially expressed or not, in the network. In our context, all the knowledge the experts need, is expressed in a non complete graph. Therefore the completion leads to the creation of a lot of useless constraints. The monitoring system is not supposed to deal with these additional constraints. Starting from this point, we propose a new method for minimizing a temporal constraint network, that selects the expressed constraints (**exclusively** the useful ones) and works with an incomplete graph.

The method uses a dual reformulation of the original TCSP into a CSP. The resultant CSP only contains linear nary equalities on real variables. Then the method reduces the dual CSP into an arc-consistent CSP, by filtering the nary constraints. The idea consists of defining the semantic of the equality constraints from the topological configuration of the primal TCSP: each constraint expresses the existence of a cycle in the primal TCSP. Moreover the reformulation only uses the useful information since it considers the expressed cycles in the scenario (see section 2). The efficiency of this method is based upon an interesting property of all the scenarios provided by the medical experts: Once represented as TCSP, the scenarios have a basis of cycles with a small number of cycles in comparison with their number of vertices. Generally, the graph provided by the medical experts contains less than 20 cycles and a great number of vertices. Although our method was NP-complete in general, we show that in our context, it leads to better results than the algorithm of Floyd-Warshall.

The following table gives the main characteristics of the two approaches: the classical one, based on Floyd-Warshall algorithm and the one we propose, called Dual&Arc-Consistency.

Floyd-Warshall	Dual&Arc-Consistency
use binary constraints: continuous real intervals including real variables	use nary constraints: linear equality on real variables
goal: reduce the constraints	goal: reduce the domains
completes the initial graph and uses the whole information	reformulates the graph and uses the expressed information (the existence of cycles).

The idea of reformulating a problem into a CSP and using the CSP techniques to solve it more efficiently, has already been explored in [12], in the context of Truth Maintenance Systems (TMS). The goal is to deduce more logical formulae from the TMS, while keeping both completeness and reasonable time-consuming. The TMS is then encoded into a dynamic CSP whose filtering process computes more deductions than the classical algorithm used in the TMS model.

The rest of the paper is constructed as follows. The next section details the three main steps of our approach and the algorithm we use to compute the dual reformulation. Theoretical and practical results are given in section 3 to compare the two methods, and confirm the interest of our approach. Section 4 provides some properties. In section 5, reusing a part of this method, we give a way of processing an incremental minimization of dynamic temporal scenarios. Section 6 is a conclusion of the interest of our approach and our work under development.

2. MINIMIZATION BY ARC-CONSISTENCY ON NARY-CONSTRAINTS

In this section, we describe a method to detect the inconsistency of a temporal constraint network representing a scenario. Previous approaches [13, 7] have shown that there exists a dual correspondence between any temporal network and a particular CSP. Starting from this idea, we propose to combine two approaches: 1) The formalism of temporal constraint networks to express scenarios in a natural way ; 2) a CSP solver to detect the relevant (not inconsistent) scenario. This requires the reformulation of the TCSP in a particular dual CSP with finite domain linear constraints and then the application of arc-consistency [14, 15] on this CSP. To achieve this reformulation, we need to identify all the cycles of the graph, which is a NP-complete problem in general. But we justify our choice by a particular property of the real-world medical scenarios: they have a low density of cycles. This section describes our method and the algorithm we use to compute all the cycles of a graph. We implemented our method using the object-oriented framework for Constraint Satisfaction BackTalk [16]. We obtained quite reasonable execution times for the task of recognition.

2.1 Reformulation of a TCSP into a dual CSP

Let G=(X,A), a given oriented TCSP. X (respectively A) is the set of nodes (respectively. links). |X| =N. |A| =M.

Let G'=(X',D',C'), the resulting CSP.

The reformulation of G into G' consists of two steps:

First, we look for all the cycles of G, using the algorithm described in the subsection 3.1.c. We compute for each cycle e the set of its arcs (a_1,...,a_k) and the orientation vector $\overrightarrow{mu_e}$ defined by the following relation:

For all arc a_i in A,

$mu_e(i) = 0$ if e doesn't contain the arc [a_i=(s,t)].

$mu_e(i) = 1$ if e contains the arc a_i

$mu_e(i) = -1$ if e contains the arc [$-a_i$=(t,s)].

Then, we define the CSP G'.

1. X' = {$v(e_i,e_j)$ / e_i in X, e_j in X and there is a link from e_i to e_j in the primal graph G}
2. D' = Scalar of D_{ij}, where D_{ij} is the domain of the variable $v(e_i,e_j)$. D_{ij} is exactly the interval labeling the arc from e_i to e_j in the primal graph G.
3. The constraints represent the existence of cycles in the primal graph and they are deduced from the property of consistency of a cycle. They are nary and linear equalities. Figure 2 shows this deduction on a simple instance of TCSP.

```
Algorithm dualOf(G):
result = a CSP
temporal variables : C
Begin
        C := AllCycles(G).
        X' := ∅.
        For each arc (s,t) in A, X' := X' U {v(s,t)}.
        For each v(s,t) in X', D'(v(s,t)) := Label((s,t),G).
        For each cycle e=(a1,...ak) in C,
        Begin
          create the vector ⟶ ,mue.
          create the constraint [ ∑ (mue(i)*vi) = 0 ], where vi is
the variable corresponding to the i-th arc of A.
          Add the constraint in C'.
        End.
        Return (X',D',C')
End.
```

The AllCycles function takes one temporal constraint network G and returns the list of all the simple cycles of G. The variable *D'* is represented by an array in which each entry is indexed by one variable of *X'*.

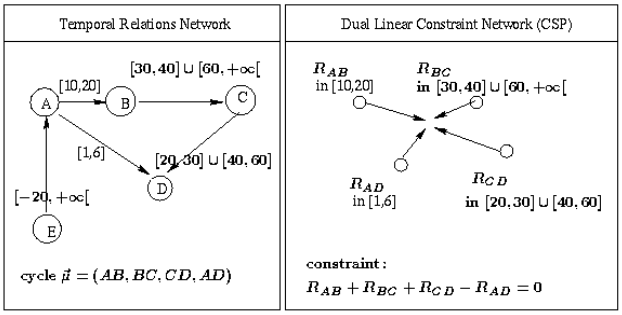


Figure 2. Table of conversion of a cycle of binary temporal constraints in a linear arithmetic equality constraint.

In this example, the constraint *EA* will not be reduced by the minimization process (see the section 4 for a proof), so it will be removed and only the arc *AB*, *BC*, *AD* and *DC* will be used to compute the dual CSP. This one is shown on the right box. It has 4 variables related by one constraint which expresses the fact that there is only one cycle in the STP.

Minimizing the STP is strictly equivalent to perform an arc-consistency procedure on its dual CSP. The method and its proof are described in the section 2.2 and 4. Notice that this technique can be used for the minimization of TCSP (the temporal constraints are disjunctive). In this case, the variables will have discontinuous domains and the complexity of arc-consistency will be increased, lowering the interest of our approach.

2.2 Detection of inconsistency and minimization

In section 4, some theoretical properties show that if any primal graph G is such that every cycle is consistent and minimal then the corresponding dual CSP G' is arc-consistent. Moreover, as far as domain reduction is concerned, on linear constraints and interval domains, arc-consistency produces the same result as filtering. These properties allow us to detect any inconsistency in G by simply performing a procedure of filtering on the dual graph G'. The obtained graph is exactly the minimal graph of the initial temporal network G.

```
Algorithm minimize(G):
result = G', the minimal graph of G

Begin
        G' := dualOf(G).
        arc-consistency(G').
        affectDomainsToArc(G', G).
        return G        "If G is inconsistent, G' contains
empty domains, otherwise G' is arc-consistent"
End.
```

The affectDomainsToArc procedure is responsible for the update of the primal graph G, from the dual one G'. The cost of this task is negligible if one maintains a link between every arc of G and its constrained variable in G'.

```
Algorithm isInconsistent?(G):
result = true if G is inconsistent

Begin
        G' := dualOf(G).
        arc-consistency(G').
        if there is an error then Return true "G is inconsistent"
        return false "G has no inconsistency"
End.
```

The following figure (Figure 3) illustrates the dual CSP constructed by our method from the scenario of Figure 1. As the latter includes three cycles, the dual CSP has three constraints.

Variables:

$X' = \{V_{4,5}; V_{5,8}; V_{12,8}; V_{7,8}; V_{6,7}; V_{5,6}; V_{11,4}; V_{11,2}\}$

Domains:

$V_{4,5}$ in {0}; $V_{5,8}$ in [4,30]; $V_{12,8}$ in [4,12]; $V_{7,8}$ in [4,12]; $V_{6,7}$ in [0,4]; $V_{5,6}$ in [0,60]; $V_{11,4}$ in [0,15]; $V_{11,12}$ in [0,45].

Constraints:

[1] : $V_{4,5} + V_{5,8} - V_{12,8} - V_{11,12} + V_{11,4} = 0$.
[2] : $V_{4,5} + V_{5,6} + V_{6,7} + V_{7,8} - V_{12,8} + V_{11,12} + V_{11,4} = 0$.
[3] : $V_{5,8} - V_{7,8} - V_{6,7} - V_{5,6} = 0$.

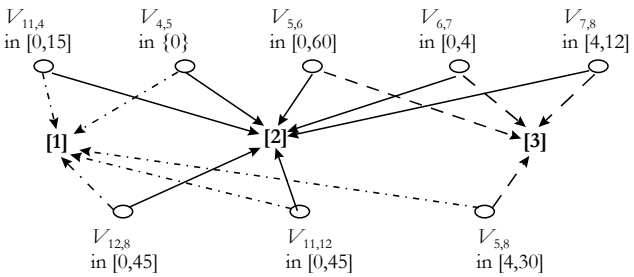


Figure 3. Dual CSP G' represented by a text or by a hyper-graph.

This scenario is consistent but not minimal. Indeed the algorithm returns that the corresponding TCSP has no

inconsistencies. More precisely, the TCSP obtained by our method is the minimal graph.

2.3 The identification of all the cycles in a connected graph

The Dual CSP construction requires the set of all the cycles of the graph. As the temporal constraint networks representation used for the medical scenarios, does not include this knowledge, we need to compute it. This task is the crucial point of our approach. Hence, its complexity is mainly responsible for the time requirements of the whole approach and great care should be given to its implementation.

In [17], Read and Tarjan propose an algorithm that computes very efficiently all the simple cycles of a connected graph. This algorithm is based on an optimized backtracking procedure and has $O(N+M+MC)$ time requirements, where C stands for the number of cycles in the graph.

In the worst case, C equals 2^k-1, with $k = M-N+1$. As it is not possible to have information about the topology of our temporal constraint network before its minimization, we don't know the number of cycles, *a priori*. Thus we present a theoretical complexity study based on the computation of all the possible cycles. It corresponds to the case when 2^k-1 cycles really exist in the graph. In this context, the above algorithm performs no backtracking at all and its time bounds become $O(N+M+M*(2^k-1))$. We implemented a simple algorithm that tries the 2^k-1 possible cycles in a connected graph and we use it to compare the efficiencies of the Floyd-Warshall algorithm and our approach. It calculates a basis of simple cycles and then combines them to deduce all the other cycles.

3. RESULTS

This section details the theoretical and practical studies of the evolution of our method, when the size of the primal graph changes. We compare this evolution with the Floyd-Warshall method, relatively to the value of k (the cardinality of a basis of cycles). We give a limit value of k, under which the Dual&Arc-Consistency method (Dual&AC) is faster than the algorithm of Floyd-Warshall, and the experiments confirm this theoretical study.

3.1 Theoretical study

First, we recall that the number of cycles in a basis B, is $M-N+1$ and noted k (we use the notations of section 2). Let us study the time complexity of our method and compare it with the classical one.

The search of the basis B is in $O(NM)$ complexity. The chosen algorithm builds a covering tree ($N-1$ nodes) and successively completes it with the remaining arcs ; thus its complexity is $O((N-1)*(M-N+1))=O(MN)$. Starting from B, the computational burden for the search of all the cycles is in $O((2^k-1-k)*M)$ (see section 2.3). The cost of the arc-consistency process is $O(b*C)$ where b is constant and C is the number of constraints expressed in the dual CSP (*i.e.* the number of cycles existing in the STP) Therefore, the time requirement of this task should be negligible in comparison with the reformulation process one. Thus the complexity of this approach is theoretically exponential. But if k is small, i.e. M is closed to N, then the complexity becomes $O(a*N^2)$, which is better than the Floyd-Warshall algorithm. Below we compare the time complexities.

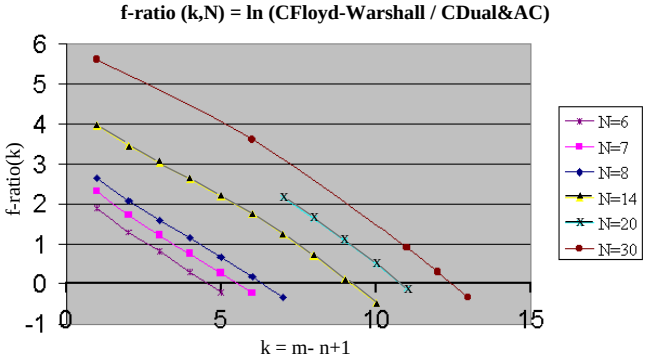


Figure 4. Theoretical study of the function f-ratio(k,N) = ln(FloydWarshall-Complexity / Dual&AC-Complexity).

The Figure 4 shows the curve of the function f-ratio depending on k and N. f-ratio allows us to compare the two complexities and gives a theoretical heuristic to choose between the two methods : if f-ratio(k,N)>0 then Dual&AC must be preferred to Floyd-Warshall. In the figure, the curves are calculated keeping N constant and changing k. To build the curves, we used a program that simulates a random generation of STP for a given number of nodes. It appears that the trends of the curves are similar for all the values of N. This result shows that the Dual&AC approach is consistent compared to Floyd-Warshall. Note that the results do not take into account the initialization nor the Arc-Consistency complexity.

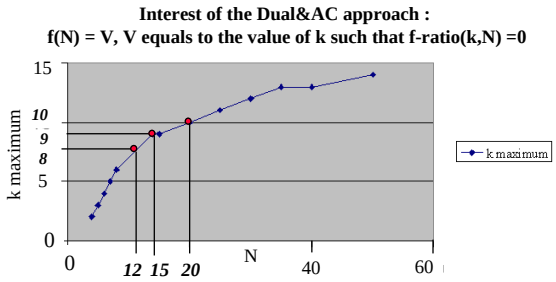


Figure 5. The maximal threshold V of k beneath which Dual&AC is interesting.

The interest of Dual&AC can be given by the maximal value V of k : for every k <V, the Dual&AC method leads to better result than Floyd-Warshall. We obtained this value (V) by noting the values k such as f-ratio(k,N)=0 (see the function in Figure 5). Theoretically, the maximal value of k is given by $ln($**Erreur !** $)=ln(N)$. As shown in Figure 5, we find that V depends on N with the relation : $4,8*ln(N)) - 4,38$, deduced from the curves in Figure 4.

3.2 Practical results

To evaluate our method, we compare the time complexity for the two approaches, depending on k and N, the number of nodes in the graph.

Figure 6 shows the complexities of the two methods, Floyd-Warshall and Dual&AC. Three cases are considered : N=12, N=15 and N=20. The cases N=12 and N=15 correspond to real-world scenarios, specified by medical experts. The case N=15 is inspired by [18].

The practical results confirm the theoretical ones: there is a limit value V for k, beneath which our method is clearly preferable to Floyd-Warshall ; For N = 12, V = 5,6 ; for N = 15, V = 6,0 and for N=20, V = 7,0. These values are lower than the theoretical limit we have found ($V_{N=12} \approx 8$, $V_{N=15} \approx 9$, $V_{N=20} = 10$). This is due to the board effect of the initialization and the BackTalk treatment. However the trend of the curves are very similar to the theoretical ones and it suffices to "check out" that the Dual&AC approach gives better results than Floyd-Warshall for k lower than a limit value. When k is higher than V, the Dual&AC approach becomes exponentially expensive. Concerning the gap between theoretical and practical values of V, it is important to notice that the theoretical results are true "more or less a constant" which depends on the computer used to do the experiments. As the theoretical value is given by *4,8*ln*(N) - *4,38*, the constant *4,38* is supposed to change. In Figure 7, the experimental limit value is given by *4,8*ln*(N) - *7,38*.

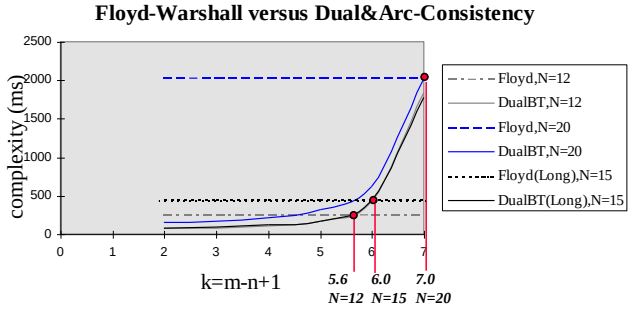


Figure 6. Comparison of the complexities of Floyd-Warshall and Dual&AC, depending on k.

The curves for N=12 and N=15 are closely mingled. This result confirms the fact that the behavior of the method is very distantly dependent of N. The value of N only influences the position of the curves and consequently the value of V ; but the behavior is always the same, as also shown in the following figure.

The Figure 7 shows the evolution of the logarithm value of the ratio (*Floyd-Warshall-Complexity/Dual&AC-Complexity*). It aims to find a criteria for choosing the more convenient method to minimize a STP, according to the properties of the STP itself: given a STP G=(N,M), one should minimize G by Dual&AC if f-ratio(M-N+1)>0, and by Floyd-Warshall otherwise.

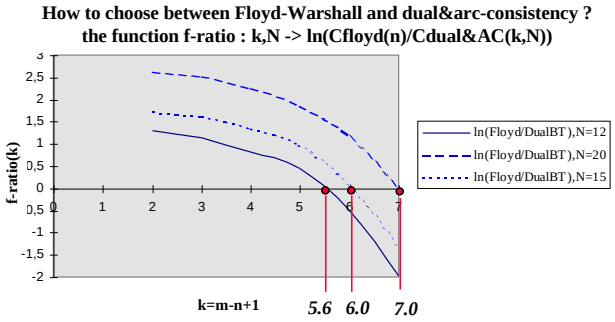


Figure 7. Experimental curves of the function f-ratio.

Once more, the results globally confirm the theoretical ones. The theoretical curves of f-ratio have a decreasing and almost linear trend. Practically the global trend is more precise. There is a minimal cost of problem's building that we cannot prevent. We find that for problems with a small k, this cost should be taken into account and could influence the final complexity.

In the previous study, there are as many basic calculations as possible cycles. Each basic calculation creates a linear nary constraint in the dual CSP only if the temporal network actually contains the potential cycle. Then, one can avoid some useless operations in the AllCycles procedure, as proposed by the Read and Tarjan algorithm (see section 2.3). Thus one can still improve the global efficiency as shown in the following figure.

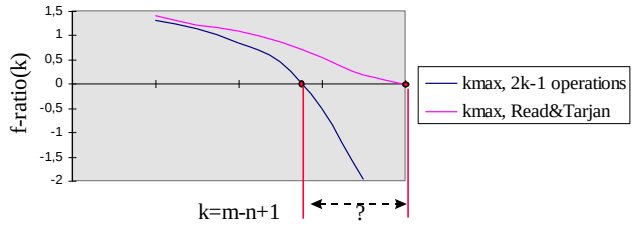


Figure 8. Pessimistic criteria for choosing between Floyd-Warshall and Dual&AC.

The optimal criteria would be the maximal admissible value of k above which Floyd Warshall is preferable to Dual&AC. As shown in the figure, the criteria k<*4,8*ln*(N)-*7,38* is not optimal. As a matter of fact, it only gives a pessimistic criteria.

4. PROPERTIES

Our approach assumes that only the constraints involved in a cycle of a STP are needed to perform the minimization. Below is the proof that this hypothesis is a valid one. As a matter of fact only the constraints involved in a cycle may be modified by any minimization procedure. As a consequence, if all the cycles of a STP are minimal and consistent, then the whole STP is minimal and consistent.

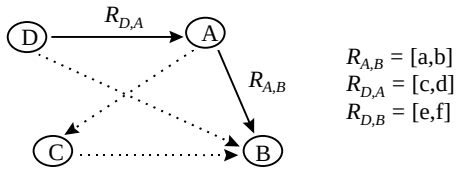
4.1 Property 1

This is a property of the minimization on a cycle of 3 nodes.

For all cycles C of 3 labeled arcs, the minimization of C doesn't change the label of its arcs if there is one or two arc(s) with universal constraint.

Proof:

Recall what the minimization consists of. We have 3 nodes A,B,C related two by two by interval constraints:



Preliminary definitions:

- The composition $\underline{o}$ of two intervals is the classical operation of composition in the domain of intervals. For all floats a,b,c,d, $[a,b] \underline{o} [c,d] = [a+c, b+d]$.
- The opposite of one interval $R = [a,b]$ is by definition $[-b,-a]$ and we note it -R.
- We note ∞ the universal constraint : $]-\infty, +\infty[$.
- The minimization consists of reducing the intervals, using the composition $\underline{o}$.
  In the example, R_{DB} will be updated by $R_{DB} \cap (R_{DA} \underline{o} R_{AB}) = [max(e, a+c), min(f, b+d)]$; R_{AB} and R_{DA} will be updated by a similar relation.

In the example, R_{DA}, R_{DB} and R_{AB} may be changed by minimization.

Now, let us consider the situation in a constraint graph, where three nodes A,B,C, are not in a cycle. What happens while minimizing ?

There is two possible cases.

If one constraint is in a cycle of 3 arcs, with two universal constraints, then it isn't modified. This is the case of R_{AB} in the figure. We have: $R_{AB} = R_{AB} \cap (R_{AC} \underline{o} R_{CB}) = R_{AB} \cap \infty = R_{AB}$.

If two constraints are in a cycle of 3 arcs, with one universal constraints, then they aren't modified. This is the case of R_{DA} and R_{AB}. We have:

$R_{DA} = R_{DA} \cap (R_{DB} \underline{o} - R_{AB})$

$R_{DA} = R_{DA} \cap ((R_{DA} \underline{o} R_{AB}) \underline{o} - R_{AB})$

$R_{DA} = R_{DA} \cap ((R_{AD} \underline{o} - R_{AB}) \underline{o} R_{DA})$, because the composition $\underline{o}$ is associative and commutative.

$R_{DA} = R_{DA}$, because of the lemma: for all intervals R_1 and R_2, $R_1 \cap ((R_2 \underline{o} -R_2) \underline{o} R_1) = R_1$.

The same reasoning can be applied to R_{AB}.

Finally we proved the property.

Lemma :

For all interval R_1 and R_2, $R_1 \cap ((R_2 \underline{o} -R_2) \underline{o} R_1) = R_1$

Proof:

Trivially, there exists a positive float, say α, such that, for all intervals R_2, $(R_2 \underline{o} -R_2) = [-\alpha,\alpha]$.

We note that, by definition, for all interval R, $R = [inf(R), sup(R)]$.

$$\begin{aligned} R_1 \cap ((R_2 \underline{o} -R_2) \underline{o} R_1) &= R_1 \cap ([-\alpha,\alpha].\underline{o} R_1) \\ &= R_1 \cap ([-\alpha+inf(R_1), \alpha+sup(R_1) \\ &= [max(inf(R_1),inf(R_1)-\alpha),min(sup(R_1),sup(R_1)+\alpha)] \\ &= [inf(R_1), sup(R_1)] \\ &= R_1. \end{aligned}$$

4.2 Property 2

In an incomplete STP, the missing constraints are replaced by universal ones, through completion. Thus we can enunciate the following property:

Every interval constraint which is not included in an expressed cycle of a STP graph G, is not changed by the minimization process. So there's no need to consider them in order to detect the inconsistencies in G.

Proof :

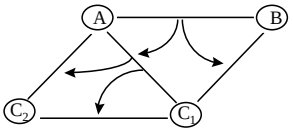


Figure 9. Minimization of a constraint in a completed STP

Let us consider two nodes A and B of a completed STP noted G. Let H be the following hypothesis.

H: The constraint *AB* is reduced by minimization of G and *AB* is not included in any cycle of non universal constraints.

Following property 1, *AB* can be reduced only when considered in a triangle of non universal constraints. Let AC_1 and BC_1 be these constraints. We now face two possibilities:

1. AC_1 and BC_1 were not universal before minimization.
2. AC_1 and/or BC_1 have been previously reduced by the minimization.

The first possibility contradicts H, because ABC_1 would have been a cycle of non universal constraints before minimization.

The second possibility implies that AC_1 and/or BC_1 have been considered in triples of non universal constraints during a previous operation of the minimization. Let us consider the case of AC_1 and let us introduce the triple (AC_1, AC_2, C_2C_1). Once more, two possibilities exist:

1. AC_2 and C_2C_1 were not universal before minimization.
2. AC_2 and/or C_2C_1 were previously reduced by minimization.

The first possibility contradicts H because of the cycle AC_2C_1B.

A similar reasoning can be performed with the second possibility. This recursive reasoning must stop because the number of node of the STP is finite. Therefore, we will end by deducing that *AB* was included in a cycle of non universal constraints before minimization.

Thus H is an inconsistent hypothesis. We conclude that any constraint that can be reduced by minimization in a non completed graph, has to be included in a cycle. The property is proved.

4.3 Property 3

Let G be a TCSP. Let B be a basis of cycles on G.
If all the cycles of G are consistent and minimum then G is consistent and minimum.

Proof:

Let assume that G is inconsistent (or not minimal), then there is a cycle which is not consistent (or not minimal) because of the property 1. This consequence contradicts the hypothesis of the property because one of the cycle is inconsistent or not minimal. So that proves the property.

Finally, the minimization of a non completed STP is only dependent on the cycles included in it. This justifies the Dual&AC approach.

5. INCREMENTAL MINIMIZATION

In our context, the session dynamically changes every time a temporal measure is done. In this case, the fusion graph only changes locally. Considering this fact, an incremental minimization would be more adapted, since it would avoid the calculation of the whole expressed cycles of the STP. We just present here an idea of what the incremental minimization should be in our context and how our method can be extended with this aim.

The incremental minimization requires some knowledge to be memorized. The temporal network stores a permanent link to its dual CSP. Every event modifying the temporal network is propagated in the corresponding dual CSP. Thus, the CSP becomes a dynamic CSP as defined in [19]. Two events are authorized in the context of medical monitoring: adding or removing a temporal constraint. Adding a constraint over-constrains the current CSP, whereas removing a constraint requires a mechanism to restore the domain of the variables linked by the removed constraint. For this reason, after adding a constraint, we attach a historic of the domains reduction to this constraint,. This can be modeled by a set of associations (variable -> old domain). Whenever a new constraint *C* is created, for each variable *v* of *C*, the domain value of *v* is added in the historic of *C*. Whenever a constraint *C* is removed, for each variable *v* of *C*, so that it remains in the CSP, the domain of *v* is set to its memorized value.

Moreover the CSP is always kept arc-consistent, hence the STP is automatically maintained minimal. As we assume that the STP should be connected, we must remove all the unlinked nodes and arcs of the STP and also the unconstrained variables of the CSP, during the retraction of one constraint.

The following figure shows what happens when adding a binary constraint in the scenario G of Figure 1.

First let us add the constraint (e3, e13), labeled D1. This doesn't create any cycle. Thus C is empty and no propagation is performed in the dual CSP.

Then we add the constraint (e13, e6), labeled D2. This adds three cycles, returned by the cyclesWithArc((e13,e6), G) function call. Then three nary constraints are created with their historic.

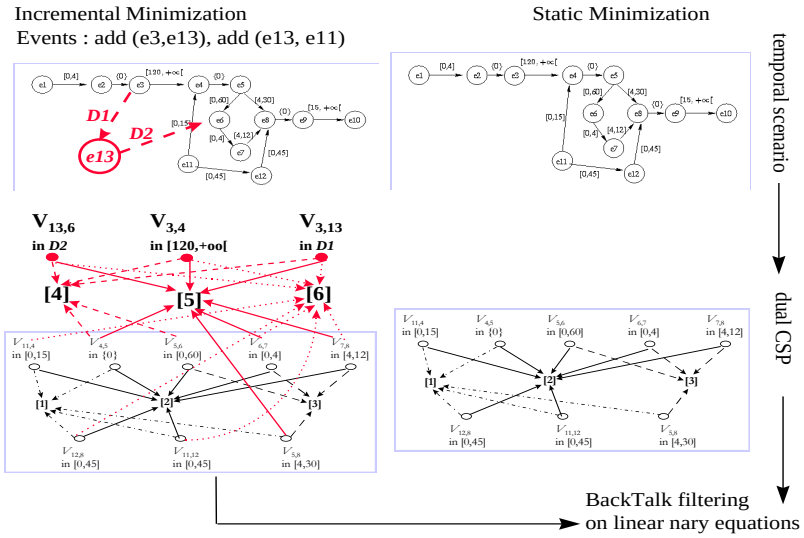


Figure 10. Incremental minimization when a new constraint is added.

As far as incremental addition is concerned, we just need to compute the new cycles including the new temporal constraint *C*, because the other ones were previously transformed in the dual CSP. The responsible procedure can be easily implemented, reusing the Read and Tarjan algorithm. Indeed the basic step consists in computing the cycles including a particular edge of the graph. Thus it is quite well adapted to the calculation of all the cycles containing *C*.

```
ActionWhileAdding (Co, G)
Comments: add Co in the STP G and propagate in the dual CSP.
Data: G=a STP, Co=a binary constraint, G'=the dual CSP of G,
G'=(X',D',C').
Begin
   AddBinaryConstraint(Co, G).
   C := CyclesWithArc(Co, G').
   if C is not empty then
   Begin
      create the variable v(Co) in X'.
      for each cycle in C, create a nary constraint in C'. (as
explained in section 2)
      Arc-consistency (G').
   End.
End.
```

```
ActionWhileRemoving(Co, G)
Comments: remove Co from the STP G and propagate in the
dual CSP.
Data: G=a STP, Co=a constraint in G, G'=the dual CSP of G.
Begin
   RemoveBinaryConstraint(Co, G).
   remove useless constraints in G.
   remove nary constraints related to v(Co) from C'.
   let be C the set of these obsolete constraints.
   remove unconstrained variables from X'.
   for each constraint c of C, for each variable v of c, remaining
in X', Begin
      oldD := old domain of v from the historic of c.
      if oldD includes current domain of v, then restore oldD as
the current domain of v.
   End.
   Arc-consistency(G').
```

End.

To perform the arc-consistency, we reuse the filtering procedures of the CSP environment BackTalk, which we have also used for the global minimization.

6. CONCLUSION

In this paper, a particular use of nary constraints was presented and applied in the context of medical scenario recognition. It was proved that the minimization of the STP is equivalent to the arc-consistency of a particular Finite-Domain CSP with linear nary equalities. This work also shows, on a real-world example, the interest of the dual reformulation of a problem using only the useful information (here the existence of a cycle). From this result, we built a method which leads to better results than the classical algorithm when the STP has a basis of cycles with a cardinality less than a given limit. The method is very distantly dependent on the number of nodes of the STP. Moreover, it offers the advantage of being easily reusable, since it only has to be connected to a CSP solver and to have a graph of numerical temporal constraints as input. Thus, this idea could be reused in various applications areas, such as Time-Map Management and environments dedicated to multimedia document authoring [20]. Indeed these problems represent temporal graphs with few cycles due to their strong semantic nature.

As further works, we would have a more general and more efficient method. As the transformation TCSP-CSP doesn't modify the domains of the temporal constraints, we can apply it on general TCSP with discontinuous domains. Then the CSP computed by our method have discontinuous real domains of variables. We plan to exploit the optimized filtering algorithm for real variables to use our method for the minimization of general TCSP.

In our context, as we assume that the session is consistent, the global minimization of the graph may be avoided and replaced by an incremental one. Thus we are currently extending it in order to perform an incremental minimization, whenever the graph locally changes. The idea is to check the truth of the relation of equivalence between the minimization and the dual-arc-consistency when we use an incremental minimization and a dynamic arc-consistency, as defined in [19].

7. REFERENCES

[1] J. Hunter. Decision support in the operating theater and intensive care : a personal view. *Artificial Intelligence in Medicine*, vol. 11, pp. 93-06, 1997.

[2] M. Dojat, A. Harf, D. Touchard, M. Laforest, F. Lemaire and L. Brochard. Evaluation of a knowledge-based system providing ventilatory management and decision for extubation. *American Journal of Respiratory and Critical Care Medicine*, vol. 153, pp. 997-1004, 1996.

[3] N. Ramaux, D. Fontaine and M. Dojat. Temporal scenario recognition for intelligent Patient Monitoring. *AIME'97, Grenoble, France*, Springer-Verlag, pp. 331-342, 1997.

[4] D. Mc Dermott. A temporal logic for reasoning about processes and plans. *Journal of Cognitive Science*, vol. 6, pp. 101-155, 1982.

[5] R. Dechter, I. Meiri and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, vol. 49, pp. 61-95, 1991.

[6] M. Ghallab & M. Alaoui. Managing efficiently temporal relation through indexed spanning trees. *Proceedings of 113th IJCAI conference, Detroit, USA*, pp. 1297-1303, 1989.

[7] C. Dousson, P. Gaborit and M. Ghallab. Situation Recognition : Representation and Algorithms. *Proceedings of IJCAI'93, Chambéry (France)*, vol. 1, pp. 166-172, September 1993.

[8] D. Fontaine & N. Ramaux. An approach by graph for the recognition of temporal scenarios. *IEEE Transactions on Systems, Man and Cybernetics*, vol. to appear, 1998.

[9] J. Forbes. The BATmobile : towards a Bayesian automated taxi. *Proceedings of IJCAI'95, Montréal, Canada*, vol. 2, pp. 1878-1875, August 20-25 1995.

[10] D.J. Musliner, E.H. Durfee and K.G. Shin. World modeling for the dynamic construction of real-time control plans. *Journal of Applied Artificial Intelligence*, vol. 74, pp. 83-127, 1995.

[11] S. Zilberstein & S. Russell. Optimal composition of real-time systems. *Journal of Applied Artificial Intelligence*, vol. 82, pp. 181-213, 1996.

[12] C. Bessière. Application of constraint Networks Filtering Techniques to Truth Maintenance Systems. *Proceedings of the IEEE Conference on Applications of AI, Orlando, Florida*, pp. 41-47, 1993.

[13] M. Vilain, H. Kautz and P. van Beek. Constraint Propagation Algorithms for Temporal Reasoning : A Revised Report, Readings in Qualitative Reasoning about Physical Systems.Morgan Kaufmann. pp. 373-381, 1989.

[14] Alan K. Mackworth. Consistency in Networks of Relations. *Artificial intelligence*, vol. 8(1), pp. 99-118, 1977.

[15] C. Bessière & J.Ch. Régin. Arc-consistency for General Constraint Networks : Preliminary Results. *Proceedings of IJCAI'97, Nagoya, Japan*, vol. 1, pp. 398-404, August 1997.

[16] P. Roy, A. Liret and F. Pachet. A Framework for Object-Oriented Constraint Satisfaction Problems. *ACM Computing Surveys, special issue on Frameworks*, vol. to appear, 1998.

[17] R. C. Read & R.E. Tarjan. Bounds on Backtrack Algorithms for Listing Cycles, Paths, and Spanning trees. *Networks*, vol. 5, pp. 237-252, 1975.

[18] W.J. Long. Reasoning about state from causation and time in a medical domain. *Proceedings of AAAI'83, Washington, USA,* AAAI Press/MIT Press, vol. 1, pp. 251-254, August 22-26 1983.

[19] C. Bessière. Arc-Consistency in Dynamic Constraint Satisfaction Problems. *Proceedings of AAAI'91, Anaheim, CA*, vol. 1, pp. 221-226, 1991.

[20] M. Jourdan, C. Roisin and L. Tardif. Visualization of constraint-based temporal scenarios in a multimedia authoring tool. Unité de recherche INRIA Rhônes-Alpes, report 3284, Montbonnot St Martin, France, october 1997.