Summary of Class Discussion of Mar 5, Mar 10 on Temporal Constraint Networks

Content: Section 12.2, 12.3, 12.4, 12.5

Speaker: *Chao Chen*
Scribe: *Yaling Zheng*
Corrector: *Cate Anderson*

Mar 13, 2003

Introduction

In the last two classes, Chen covered quantitative temporal networks touching on the following topics*:*

- *Simple Temporal Problem (STP),*
- *Temporal CSP (TCSP),*
- *Algorithms to solve or reinforce consistency in STP & TCSP including*
  - *Floyd-Warshall's algorithm(F-W),*
  - *Numerical Path Consistent (NPC-1, NPC-3),*
  - *Meta-CSP and*
  - *Algorithm Upper Lower Tightening(ULT),*
  - *comparison of NPC and ULT.*

1. Quantitative Temporal Networks

Chen discussed the difference between quantitative temporal networks and qualitative networks. Quantitative temporal networks explicitly deal with numbers. Consider time points as the variables and represent constraints as a set of intervals {[a_1, a_2], [b_1, b_2], …., [m_1, m_2] }, while qualitative networks deal with the relation between time points (in point algebra) or intervals (in interval algebra), consider intervals variables and represent constraints by relations such as { b, = , m , o, d , s , f , bi, mi , oi , di , si , fi } or {> , = , <}.

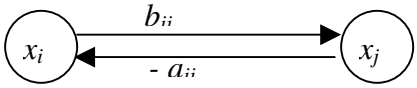
2. Simple Temporal Problem (*STP*)

A Simple Temporal Problem (STP) is a special class of Temporal Constraint Satisfaction Problem (TCSP) in which every constraint specifies a single interval. The advantage of STP is that is can be processed in polynomial time.

In an STP network each edge e_{ij}: $i \rightarrow j$ is labeled by a single interval [a_i , b_{ij}] which can be represented by the constraints ($a_{ij} \leq x_j - x_i \leq b_{ij}$). This can alternatively be expressed by the pair of inequalities $(x_j - x_i \leq b_i) \wedge (x_i - x_j \leq -a_i)$.

Solution to *STP*

A method of solving the *STP* was given as follows: transform the STP network into a directional cyclic graph. Each variable in the STP network is represented by a vertex in the directional cyclic graph and each constraint $a_{ij} \le x_j - x_i \le b_{ij}$ in the STP network is represented by two edges between vertex x_i and x_j as follows:



Once the constraint graph has been transformed into a directed cyclic graph, Floyd-Warshall's algorithm (All Pairs Shortest Path) can be run. The complexity of algorithm F-W is $O(n^3)$ where n is the number of variables.

Based on theorem 12.2.5, if any pair of nodes has a negative or zero cycle length, the network is inconsistent and the *STP* has no solution. When all cycle lengths have positive lengths, the network is consistent. Theorem 12.2.7 indicates that any consistent STP is globally consistent, meaning that every consistent assignment of values to a set of variables can be extended backtrack free to a solution and as such is decomposable.

3. Temporal *CSP* (*TCSP*)

A Temporal Constraint Satisfaction Problem (TCSP) has a set of variables, each variable representing a time point with continuous domains. Constraints are disjoint sets of intervals, which can be represented as the disjunction of inequalities. For example, if the variable x_i had the domain $d_i = \{I_1, \ldots, I_k\} = \{[a_1, b_1], \ldots, [a_k, b_k]\}$, this could be represented as $(a_1 \le x_i \le b_1) \vee \ldots \vee (a_k \le x_i \le b_k)$.

Unary constraints restrict the domain of x_i and can be expressed as $a_i \le x_i \le b_i$, while binary constraints restrict the permissible values for the distance $x_j - x_i$ and can be respresented as $(a_{ij} \le x_j - x_i \le b_{ij})$. As in a the general CSP, the tuple $x = (a_1, \ldots, a_n)$ is a solution if $x_1 = a_1, x_2 = a_2, \ldots, x_n = a_n$ does not violate any constraints.

Inference and constraint propagation in TCSPs

To support inference over TCSPs, three operations over the variables are given: union, intersection and composition. These are defined as follows.

In a *STP*, if $T = [a, b]$ and $S = [c, d]$

Union: $T \cup S = \{[a, b], [c, d]\}$

Composition: interval addition

$T \otimes S = [a + c, b + d]$

Intersection: interval intersection

$T \cap S = [max(a , c) , min(b , d)]$

In a *TCSP* , if $T = \{ I_1,..., I_l \}$ and $S = \{J_1,..., J_m\}$

Union: (set union) $T \cup S = \{I_1,..., I_l , J_1,..., J_m \}$

Intersection: (set intersection) denoted as $T \oplus S$

$T \oplus S$ = {all z, such that $z \in T$ and $z \in S$}

Composition: (this is the cross product, the final size of T ⊗ S is the product of size T and size S)

If T = {(a, b), (c, d)} and S = {(e, f), (j, k)}

T⊗ S = {(a+e, b+f), (a+j, b+k), (c+e, d+f), (c+j, d+k)}

Brute Force method of solving a *TCSP*

One method of solving a *TCSP* is to generate all the STPs of the *TCSP,* verify them for consistency and keep the consistent (and minimal) ones. This is possible because a *TCSP* $\equiv \cup_{all}$ *STP*s. After solving each *STP* separately, the solutions are combined to form the solution for the general *TCSP*.

Dr. Choueiry emphasized theorem 12.2.11: Minimal *TCSP* $\equiv \cup_{all}$ Minimal *STPs*, which is very important since each *STP* can first be changed to a minimal *STP* using F-W algorithm and then the union of all minimal STPs is the solution for the TCSP.

Complexity of this solution is $O(n^3k^e)$, wherein k is the maximum number of labels and e is the number of edges.

The question of *what the parameter k signified* was raised, and it was clarified that k is not the domain size, but is the maximum number of intervals of a constraint.

Better alternative method of solving a *TCSP: Meta-CSP*

A better method of modeling the *TCSP* is as a Meta-*CSP* and using backtracking search to solve it.

The **Model**: The constraints are treated as in the former *TCSP* as variables with the intervals in the label of the former *TCSP* as the variables' domain. The constraint of this new *CSP* is one global constraint. We call this new *CSP* as Meta-*CSP* in which,

- The root node is the starting node (dummy node)
- The First level corresponds to a *STP* with an edge e_1 (variable)
- Each node at this level corresponds to e_1 labeled with each of its intervals i_1,
- The next level corresponds to (an *STP* with) two edges e_1 and e_2
- At a given level i, we have an *STP* with i edges…

The question that *whether a Meta-CSP is the same as a Dual graph* was raised. Dr. *Choueiry* and *Chen* informed us that they are totally different. In the dual translation, the constraints of the original problem become the variables in the new representation. However, in the Meta-CSP, the domain of this new c-variables is exactly the set of tuples

that satisfy the original constraint and there is a binary constraint between c-variables *iff* the original constraints share some variables.

Search: Run a *BT* search on the meta-*CSP*, assign intervals to edges as long as no negative cycle of current *STP* (using *F-W* algorithm) is discovered. If partial assignment cannot be extended, backtrack; otherwise add one more edge; if all the variables in this Meta-CSP are assigned a value and the network is consistent then a solution is found.

Advantages: The question was raised about the advantage of Meta-*CSP* method, Dr. *Choueiry* said if a *TCSP* is transformed to a Meta-*CSP* and then solved with *BT* it will allow us to prune inconsistencies early, thus saving much work.

4. Path Consistency (*PC*) on a Quantitative Network

The definition of path consistency (12.2.12) is given as: A temporal constraint T_{ij} is path consistent **iff** $T_{ij} \subseteq \oplus_{\forall k} (T_{ik} \otimes T_{kj})$ for all $k \neq i$ and $k \neq j$. A temporal network is path-consistent **iff** all its constraints are path-consistent.

F-W* algorithm solving *TCSP
First, the *TCSP* problem is decomposed into many *STP*s, the number of which is the product of interval numbers of each label. These *STP*s can be solved by Floyd-Warshall's algorithm, which imposes path-consistency on an *STP*.

Algorithm NPC* –1 & *NPC*-3&*DPC
NPC is the abbreviation of Numerical Path Consistent.

In algorithm *NPC*-1 and *NPC*-3, temporal constraints T_{ij} is revised by
$T_{ij} \oplus_{\forall k} (T_{ik} \otimes T_{kj})$ for all $k \neq i$ and $k \neq j$. The difference between the two algorithms lies in the termination conditions. In NPC-1, the termination condition is that the network no longer changed while the termination condition of *NPC*-3 is the constraint queue to be checked is null. *NPC*-3 is more efficient since it only handles those constraints which may be affected by the filtering.

Algorithm Directional Path Consistency(*DPC*) is more efficient than *NPC*-3 although it is a weaker algorithm. Given a variable ordering $d=(x_1,\ldots,x_n)$, *DPC* guarantee a directional path-consistent network along *d,* not full path-consistency.

The complexity of *NPC*-1 is $O(n^3R^3)$, while the complexity of *NPC*-3 is $O(n^3R)$. *Here R is the range of the network, which is the maximum range over all constraints. For example,* if all constraints are {[1, 5], [7, 9], [10, 18]}, which can be contracted to {[1, 18]} and thus the range of the network is 18−1, that is,17.

Lin Xu talked about the *R*, he thinks this *R* should be an adjusted *R*. for example, if the range is [1, 1.8769], it would be better to transform this range as an integer range [10000, 18769]. This consideration is based on his empirical experience. Integer steps are used in the actual algorithm and this kind of non-integer range should be transformed to an integer range.

Fragmentation problem: When tightening a interval $[a, b]$ to $[a_1, a_2]$, $[a_3, a_4]$ …, $[a_m, b]$, although the domain is tightened, the increased number of intervals will causes the composition operation to grow exponentially, causing the complexity of *PC* to grow exponentially. Fortunately, an algorithm *ULT*, which overcomes this kind of fragmentation, is put forward.

Algorithm Upper Lower Tightening (*ULT*) *ULT* can be summarized as follows: First, relax the *TCSP* to a *STP* by summarizing all subintervals on an edge into one interval [minimum start point of all subintervals, maximum end point of all subintervals]. Then use *F-W* algorithm, tighten the constraints of the *STP*. In the end we get a new *TCSP* by intersecting the resulting consistent *STP* with the original *TCSP*. Repeat the above steps until no change or inconsistency is detected. Thus we can arrive the tightest *TCSP* while avoiding fragmentation problems that were encountered when using the *NPC* algorithm.

In addition, based on theorem 12.2.18, algorithm *ULT* terminates in $O(n^3ek+e^2k^2)$ steps where n is the number of variables, e is the number of edges, and k is the number of intervals in each constraints. In contrast to path-consistency algorithm, which cannot guarantee terminate in polynomial times, algorithm *ULT* is guaranteed to converge in a polynomial number of iterations, even if the interval boundary is not rational. So we can conclude algorithm *ULT* is better than *NPC*-1 and *NPC*-3.

Network-Based Algorithms
Since temporal networks allow only unary or binary constraints, we can only exploit structures having induced-width of 1 or 2, or tree-decompositions whose separators size is 1 or 2.

However, applying a general tree-decomposition algorithm is not natural since it requires enforcing constraints of large scopes, also, the idea of cycle-cutset cannot be employed beneficially in quantitative temporal problems because the special structure of *TCSP*s that makes back-tracking used in the solution of *TCSP*s instantiates arcs, rather than variables.

5. Translation between representations *PA* and *IA* to *TCSP*

We have discussed many algorithms for solving *TCSPs* based on Point Algebra. *PA* can be viewed as a special case of *TCSP,* which lacks metric information. Some of interval algebra can be translated into point algebra, for example,
$x_i < x_j \Rightarrow x_j - x_i \in (0, \infty)$

However, (*IA*) cannot always be translated into *PA* (binary *TCSP*). Thus strategies solving *IA* are left for further study.