Efficient Message Passing and Propagation of Simple Temporal Constraints: Results on Semi-Structured Networks

Hung H. Bui and **Mabry Tyson** and **Neil Yorke-Smith***
Artificial Intelligence Center, SRI International
333 Ravenswood Ave., Menlo Park, CA 94025, USA
{bui,tyson,nysmith}@AI.SRI.COM

Abstract

The familiar Simple Temporal Network (STN) is a widely used framework for reasoning about quantitative temporal constraints over variables with continuous or discrete domains. The inference tasks of determining consistency and deriving the minimal network are traditionally achieved by graph algorithms (e.g., Floyd-Warshall, Johnson) or by iteration of narrowing operators (e.g., $\triangle$STP). However, none of these existing methods exploit effectively the tree-decomposition structure of the constraint graph of an STN. Methods based on variable elimination (e.g., adaptive consistency) can exploit this structure, but have not been applied to STNs as far as they could, in part because it is unclear how to efficiently pass the 'messages' over a set of continuous domains. We first show that for an STN, these messages can be represented compactly as sub-STNs. We then present an efficient message passing scheme for computing the minimal constraints of an STN. Analysis of this algorithm, *Prop-STP,* brings formal explanation of the performance of the existing STN solvers $\triangle$STP and SR-PC. Empirical results validate the efficiency of Prop-STP, demonstrating performance comparable to $\triangle$STP, in cases where the constraint network is known to have small tree-width, such as those that arise in Hierarchical Task Network planning problems.

Introduction

Quantitative temporal constraints are essential for many real-life planning and scheduling domains (Smith *et al.* 2000). Many systems adopt a Simple Temporal Network (STN) (Dechter *et al.* 1991) to represent and reason over the temporal aspects of such problems, associating time-points with the start and end of actions, and modeling the temporal relations by binary simple temporal constraints. We present a general, efficient message passing scheme for propagation of such constraints, and evaluate its insights for semi-structured networks that arise in HTN planning systems.

The central role of STNs in deployed planning systems (Laborie and Ghallab 1995; Myers *et al.* 2002; Bresina *et al.* 2005; Castillo *et al.* 2006) makes efficient inference with STNs especially important. The two principal inference tasks, determining consistency of an STN and deriving its minimal network, can be achieved by enforcing path consistency (PC) (Dechter *et al.* 1991). The common approach is to run an All-Pair Shortest Path graph algorithm on the *distance graph* of the STN. Algorithms such as *Floyd-Warshall* (denoted F-W or PC-1) (time complexity $\Theta(N^3)$), or *Johnson* (complexity $\Theta(N^2 \log N + NM)$, where M is the number of edges in the constraint graph) can be used (Cormen *et al.* 1990).

To achieve better efficiency, significant efforts have been made to apply more sophisticated constraint propagation techniques to STNs. *Partial Path Consistency* (PPC) (Bliek and Sam-Haroud 1999) can be applied to a triangulated constraint graph rather than a complete graph and is sufficient for backtrack-free reconstruction of all solutions. The state-of-the-art $\triangle$*STP* (Xu and Choueiry 2003) is a specialized solver based on PPC and operates over triangles of the triangulated STN. If only consistency is required, but not the minimal network, *Directional Path Consistency* (DPC) can be used with time complexity $\mathcal{O}(Nw^2)$, where w is the induced tree-width along the node ordering used (Dechter 2003). Empirical comparisons on random STNs (Xu and Choueiry 2003; Shi *et al.* 2004) show that $\triangle$STP outperforms PC-1, Johnson's Algorithm ("Bellman-Ford"), and is comparable to (on dense graphs) or outperforms (on sparse graphs) DPC. Although the worst-case time complexity of $\triangle$STP, unstated in (Xu and Choueiry 2003), is not known, it can be bounded by $\mathcal{O}(N^3)$.

Despite the variety of methods, no dedicated STN solver takes full advantage of the tree-decomposition of the STN constraint graph, namely, the ability to decompose a constraint graph into a 'tree' of variable and constraint clusters (Dechter and Pearl 1989). One exception is the specialized solver SR-PC (Yorke-Smith 2005) that exploits the structure of STNs associated with plans in the Hierarchical Task Network (HTN) planning paradigm (Erol *et al.* 1994). The HTN planning process gives rise to STNs with the *sibling-restricted* (SR) property. Such an SR-STN can be decomposed into a tree of smaller sub-STNs, mirroring the shape of the hierarchical structure in the plan. SR-PC traverses this tree, invoking an STN solver at each sub-STN. While SR-PC shows strong empirical performance on SR-STN, it does not operate on general STNs.

In contrast to STNs, tree-decomposition methods are often applied to the general Constraint Satisfaction Problem (CSP). These methods, such as *variable elimination (adaptive consistency)* and *cluster-tree elimination* (CTE) (Dechter 2003), operate by decomposing a triangulated constraint graph into a tree of variable clusters and solving the sub-problem in each. Sophisticated decomposition-based

*Corresponding author

methods for the (discrete) CSP (e.g., (Jégou and Terrioux 2003)) have enjoyed success on appropriate problems (e.g., (Larrosa *et al.* 2005)); they can be seen as specializations of CTE and other generic variable elimination methods.

In this paper we apply the ideas of such tree-decomposition methods to the STN. Note that a direct application of tree-decomposition methods to the STN is non-trivial. Since the STN represents a CSP with *continuous* variables, it is not clear how to represent the 'messages', i.e., the sets of additional constraints resulting from eliminating some variables. We first show that, for an STN, these messages can be represented compactly as sub-STNs. We then present an efficient message-passing scheme, called *Prop-STP*. Like △STP, Prop-STP requires the STN to be triangulated; however, unlike △STP, Prop-STP operates over the set of maximal cliques of the triangulated constraint graph. The time complexity of Prop-STP is $\mathcal{O}(Kw^3)$ where K is the number of cliques, and w is the induced tree-width (the size of the largest clique minus 1). For STNs with known and bounded tree-width (e.g., SR-STNs), Prop-STP thus achieves linear time complexity, a substantial improvement over the use of All-Pair Shortest Path algorithms. Empirically, Prop-STP achieves the same level of performance on structured SR-STNs as the specialized solver SR-PC.

For general STNs, triangulation can be carried out efficiently by greedy methods (Kjaerulff 1990). Empirical results demonstrate that with a triangulation step, △STP outperforms earlier STN solvers (Shi *et al.* 2004; Xu and Choueiry 2003). Our analysis of Prop-STP offers insight into how to order the triangles in △STP, and also shows that △STP's complexity can be characterized in terms of the induced tree-width. Experiments with semi-structured STNs with limited tree-width indicate that Prop-STP performs at least comparably with △STP, *even when provided with no information about the STN structure*. This indicates that it is more efficient to operate on cliques rather than on triangles, as anticipated by (Choueiry and Wilson 2006).

Background and Terminology

We begin with the background necessary for the message passing scheme for STNs and the resulting Prop-STP algorithm. We are concerned with relations among a set of variables $\{x_i,\ i \in S\}$, each taking values from the domain $\mathcal{X}_i$. A relation $\mathcal{R}$ over S is simply a subset $\mathcal{R} \subseteq \prod_{i \in S} \mathcal{X}_i$. The index set S is called the *scope* of $\mathcal{R}$. We make use of two standard relational operators: projection and join. Denote the projection of a relation $\mathcal{R}$ onto the index set $V \subseteq S$ by $\pi_V \mathcal{R}$, and the join of $\mathcal{R}_1$ and $\mathcal{R}_2$ by $\mathcal{R}_1 \bowtie \mathcal{R}_2$.

Simple Temporal Networks

A STN is represented by a set of variables $\{x_i |\ i \in S\}$, representing time-points, with domain $\mathcal{X}_i = \mathbb{R}$[1]; a set of interval unary constraints $\mathcal{T}_i \subset \mathbb{R}$, where $\mathcal{T}_i = \{x_i \mid x_i \in [a_i, b_i]\}$; and a set of binary constraints $\mathcal{T}_{ij} \subset \mathbb{R}^2$, $i < j$ where $\mathcal{T}_{ij} = \{(x_i, x_j) \mid x_i - x_j \leq -a_{ij},\ x_j - x_i \leq b_{ij}\}$.[2] Generally, we assume that there are N variables, so $S = \{1, \ldots, N\}$. Given an STN $\mathcal{T}$, its *constraint graph* G is an undirected graph with vertices representing the variables; an edge links x_i and x_j iff the binary constraint $\mathcal{T}_{ij}$ exists.

We differentiate the constraints represented by $\mathcal{T}$ and the relation they imply, denoted by $sol(\mathcal{T})$ and called the *solution set* of the STN. By definition, $sol(\mathcal{T})$ is a relation with scope S that is the join of all the unary and binary constraints $\mathcal{T}_i$ and $\mathcal{T}_{ij}$. Thus every solution in $sol(\mathcal{T})$ is an assignment of values to time-points such that all constraints are satisfied. Relational operators such as join and projection can be applied on the solution set with the usual set semantics. An STN is *consistent* iff its solution set is non-empty. Two STNs are equivalent (denoted $\mathcal{T}_1 \equiv \mathcal{T}_2$) iff their solution sets are the same, while two STNs are equal (denoted $\mathcal{T}_1 = \mathcal{T}_2$) iff they contain exactly the same set of constraints.

We introduce some useful operators that operate directly on $\mathcal{T}$. Let V be a subset of the variables. The subnetwork of $\mathcal{T}$ restricted to V, denoted by $\mathcal{T}_V$, is the STN with scope V and constraints $\mathcal{T}_i, \mathcal{T}_{ij}$ for $i, j \in V$. Any solution of $\mathcal{T}$ of course will satisfy the constraints of $\mathcal{T}_V$, so $\pi_V sol(\mathcal{T}) \subseteq sol(\mathcal{T}_V)$. When $\pi_V sol(\mathcal{T}) = sol(\mathcal{T}_V)$, the STN is said to be *locally minimal* on V. We next consider two STNs with different scopes S_1, S_2 and constraint graphs G_1, G_2. The *join* of $\mathcal{T}_1$ and $\mathcal{T}_2$, denoted by $\mathcal{T} = \mathcal{T}_1 \wedge \mathcal{T}_2$, is the STN with scope $S_1 \cup S_2$ and constraint graph being the superimposition of G_1 and G_2. All constraints in G_1 (but not in G_2) and in G_2 (but not in G_1) are taken from $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively, and all constraints in both G_1 and G_2 are the pairwise intersection between constraints of $\mathcal{T}_1$ and $\mathcal{T}_2$. It is straightforward to show that $sol(\mathcal{T}_1 \wedge \mathcal{T}_2) = sol(\mathcal{T}_1) \bowtie sol(\mathcal{T}_2)$.

An STN $\mathcal{T}$ has an *equivalent* minimal network representation $\mathcal{T}^{min}$ whose constraints satisfy $\mathcal{T}_i^{min} = \pi_{\{i\}} sol(\mathcal{T})$, and $\mathcal{T}_{ij}^{min} = \pi_{\{i,j\}} sol(\mathcal{T})$ for all $i < j$. Hence, $\mathcal{T}^{min}$ is locally minimal on $\{i\}$ for all i, and on $\{i, j\}$ for all pairs (i, j). The constraint graph of $\mathcal{T}^{min}$ is thus a complete graph. Further, it has been shown that STNs are also *binary decomposable* (Dechter *et al.* 1991), i.e., for every subset of variables V, the projection $\pi_V sol(\mathcal{T})$ is expressible as a binary constraint network. Further still, the minimal network of $\pi_V sol(\mathcal{T})$ is precisely $\mathcal{T}_V^{min}$, the minimal network of $\mathcal{T}$, restricted to V. Thus for every V, $\pi_V sol(\mathcal{T}) = sol(\mathcal{T}_V^{min})$. The *minimalization* operation to compute $\mathcal{T}^{min}$ is the principal inference task for STNs.

A weaker notion, the *partial minimal network,* is denoted by $\mathcal{T}^{pmin}$ and defined by the set of constraints $\mathcal{T}_i^{pmin} = \pi_{\{i\}} sol(\mathcal{T})$, and $\mathcal{T}_{ij}^{pmin} = \pi_{\{i,j\}} sol(\mathcal{T})$ for all $i < j$ and $(i, j) \in edges(G)$. The partial minimal network thus shares the same constraint graph G with the original network, and can be obtained from the minimal network $\mathcal{T}^{min}$ by removing all binary constraints on edges that are not present in G. If the constraint graph of $\mathcal{T}$ is triangulated, given $\mathcal{T}^{pmin}$, every solution to $\mathcal{T}$ can be constructed backtrack-free.

[1] While one can consider STNs with discrete domains, we focus on the more difficult case of continuous domains. The theory in this paper can be readily specialized to the discrete case, and the algorithm we evaluate operates effectively for either case.

[2] In some representations, such as the one used by (Xu and Choueiry 2003), unary domain constraints are modelled as binary relations to a distinguished *temporal reference* time-point, denoted TR, which marks the start of time; thus, without loss of generality, all constraints may be taken to have the binary form.

Variable and Clustering-Tree Elimination

Complementary to methods that solve a CSP based on the iteration of narrowing operators, such as PC-1 and △STP (which can be seen as AC-3 operating over triangles (Xu and Choueiry 2003)), an alternative method for the general CSP is called *adaptive consistency* or *variable elimination* (Dechter 2003). Given a general (binary) CSP $\mathcal{T}$ with scope S, consider two sets of variables (called *clusters*), W and V that together cover the constraint graph G (this means $W \cup V = S$ and every edge in G belongs entirely in W or in V). If every path between W and V in G passes through $W \cap V$, we say that the two clusters are separated by their *separator* $W \cap V$, denoted by $sep(W, V)$. It is then possible to project onto W as follows:

$$\pi_W sol(\mathcal{T}) = sol(\mathcal{T}_W) \bowtie m(V, W) \tag{1}$$

where $m(V, W) = \pi_{sep(V,W)} sol(\mathcal{T}_V)$ is the *message* from V to W. This operation effectively 'eliminates' all variables in $V - W$. By eliminating the variables in a given order, we can obtain the minimal constraints. The complexity of this method thus hinges on the representation and the calculation of the messages. Because of the difficulty in representing the messages for variables with continuous domain, this idea has not been applied to STNs whose domains are inherently continuous (Dechter 2003, page 357).[3]

Cluster-Tree Elimination is a generalized variable elimination method for computing the partial minimal network $\mathcal{T}^{pmin}$ for a triangulated constraint graph (Dechter 2003). The algorithm works by decomposing the triangulated graph G into a *join-tree* (or a *junction-tree*) over a set of variables clusters $\{V_1, \ldots, V_K\}$ that cover the original graph G. The vertices $V_1, \ldots, V_K$ of a join-tree have the property that, for every tuple (i, j, k) s.t. j lies on the path between (i, k), $V_i \cap V_k \subset V_j$. Once a join-tree is constructed, messages can be passed asynchronously among the clusters along its edges. After a message has been passed in each direction on every edge of the join-tree, the resulting network can be shown to be locally minimal on every cluster V_i. Solving the local networks at each cluster then yields all the minimal constraints needed for the partial minimal network.

STN Propagation: Prop-STP

Variable Elimination for STNs

We now focus on the STN. First observe that, since every STN is binary decomposable, its projection onto an arbitrary subset of variables can be computed and represented by the minimal network. Applying this to the calculation of the message in variable elimination yields

$$m(V, W) = \pi_{sep(V,W)} \left(sol(\mathcal{T}_V)\right) = sol\left((\mathcal{T}_V)^{min}_{sep(V,W)}\right) \tag{2}$$

The message thus can be represented compactly and conveniently as the STN $(\mathcal{T}_V)^{min}_{sep(V,W)}$, which is simply the minimal network of $\mathcal{T}_V$, restricted to the separator set. We call this the *message STN*, denoted by $\mu(V, W)$. Using this compact representation of the messages, we immediately obtain an efficient variable elimination procedure. We omit proofs, referring to (Bui *et al.* 2007).

Theorem 1. *Consider an STN $\mathcal{T}$ and let V, W be two clusters of variables that cover the constraint graph G. Suppose that V and W are separated by $sep(V, W) = V \cap W$. Then the projection of $\mathcal{T}$ onto W, $\pi_W sol(\mathcal{T})$, can be represented by the solution of the STN*

$$(\mathcal{T}_W \wedge \mu(V, W)) \tag{3}$$

The above expression involves only a minimalization operation on $\mathcal{T}_V$ and a simple restriction to the separator set. Any of the STN solvers described earlier that produces the minimal network can be used for the former operation.[4] The constraint graph of the STN $(\mathcal{T}_V)^{min}_{sep(V,W)}$ is a complete graph over $sep(V, W)$ (because minimalization creates a complete constraint graph). So unless $\mathcal{T}$ is also complete over $sep(V, W)$, the operation in (3) will introduce new edges to the original constraint graph of $\mathcal{T}$.

Message Passing for STNs

Once we have a compact representation of the messages, the cluster-tree elimination algorithm can be applied immediately to compute the partial minimal network $\mathcal{T}^{pmin}$. Let us assume that we are given a join-tree $\mathcal{J}$ over a set of clusters $V_1, \ldots, V_K$ that cover the original constraint graph G. In some cases, such a join-tree can be found from the structure of G, as in the case of sibling-restricted STNs, or it can be found by first triangulating G and then extracting the set of maximal cliques of the resulting triangulated graph.

The standard message passing scheme used in cluster-tree elimination (Dechter 2003) computes a message from a cluster V_i to a neighbouring cluster V_j, using the messages V_i received from its other neighbours. Representing the message from V_i to V_j by the message STN $\mu^{i,j}$, we obtain

$$\mu^{i,j} = \left[\mathcal{T}_{V_i} \wedge \left(\bigwedge_{k \in neighbour(i), k \neq j} \mu^{k,i}\right)\right]^{min}_{sep(V_i, V_j)}$$

To compute each message requires a minimalization operation over one cluster. After computing all such messages, we must go through each cluster and minimalize it (with the neighbour messages included). Since there are $2(K-1)$ messages, the total is $2(K-1) + K = 3K - 2$ minimalization operations. At the conclusion, we obtain the minimal domain $\mathcal{T}_i^{min}$ for all i, but only $\mathcal{T}_{ij}^{min}$ for those (i, j) that belong to the same cluster. Since any edge in G must belong to one of the clusters, we obtain the partial minimal network.

An Improved Propagation Scheme

We can exploit specific properties of STN to make the message passing scheme more efficient. In the above, one minimalization operation is needed for each message. However, binary decomposability of STNs implies that all possible projections of $\mathcal{T}$ onto an arbitrary subset of its variables can

[3] Sophisticated decomposition-based methods developed for the general CSP, such as BTD (Jégou and Terrioux 2003), can be applied to the STN supposing discrete domains for the time-point variables. However, by exploiting the highly structured nature of simple temporal constraints, decomposition-based methods dedicated to the STN have inherent advantage.

[4] PPC-based methods such as △STP can also be used. However the subgraph in the separator must be complete, and the subgraph in V must be triangulated.

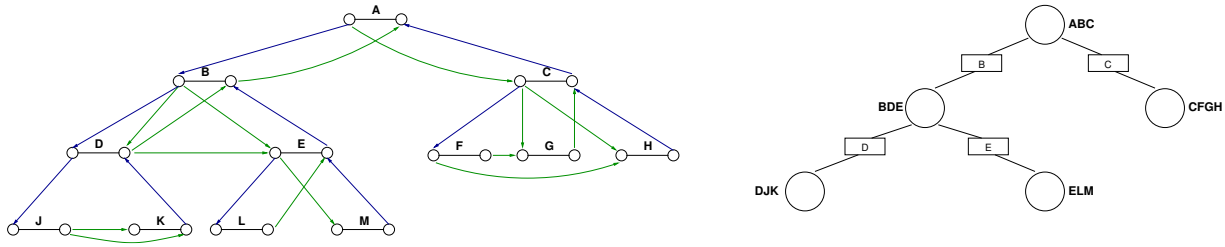

Figure 1: Left: STN; right: join-tree of clusters.

Algorithm 1 Prop-STP

1: $\mathcal{T}^0 = \mathcal{T}$
2: **for** $i = 1, \ldots, K-1$ **do** { First Pass }
3: $\quad \mathcal{T}^i = lmin(\mathcal{T}^{i-1}, V_i)$
4: **end for**
5: **for** $i = K \ldots 2K-1$ **do** { Second Pass }
6: $\quad \mathcal{T}^i = lmin(\mathcal{T}^{i-1}, V_{2K-i})$
7: **end for**
8: return $\mathcal{T}^{2K-1}$

be computed in a single minimalization operation. Thus, we can rearrange the order of message passing so that one minimalization operation can compute multiple messages. Furthermore, as we show below, it is possible to eliminate the need to store the messages all together.

First, given the join-tree $\mathcal{J}$, a cluster ordering $V_1, \ldots, V_K$ is termed *valid* if whenever V_j lies on the path from V_i to V_K then $j \geq i$. Such a valid ordering can easily be created by choosing an arbitrary cluster to be V_K and treating it as the *root* of $\mathcal{J}$. A valid cluster ordering then lists clusters that are further away from the root first, and root cluster last.

Next, we define a simple operation on STNs, termed *local minimalization*, which takes a subnetwork and replaces it with its minimal network. To be precise, given a subset of variables V, a local minimalization on V returns the STN $lmin(\mathcal{T}, V) = \mathcal{T} \wedge (\mathcal{T}_V)^{min}$. Since $(\mathcal{T}_V)^{min} \equiv \mathcal{T}_V$, it is trivial to see $lmin(\mathcal{T}, V) \equiv \mathcal{T}$ (since, if T_1 is a sub-STP of T_2, then $T_1 \wedge T_2 \equiv T_2$). Note that this operation makes the constraint subgraph in V complete.

Lemma 2. *Let $\mathcal{T}' = lmin(\mathcal{T}, V)$. If $\mathcal{T}$ is locally minimal on some $W \subseteq S$ then so is $\mathcal{T}'$.*

Lemma 3. *For any $P \subset V$ such that P separates $V - P$ and $\bar{V} = S - V$, $\mathcal{T}'$ is locally minimal on $\bar{V} \cup P$.*

Note that as long as we can find a separator P that is a proper subset of V, then $\bar{V} \cup P$ is a proper subset of S. Thus, while the original STN $\mathcal{T}$ is (of course) locally minimal on S, the new STN $\mathcal{T}'$ is locally minimal on a proper subset of S. By repeatedly applying the same kind of operation, we can obtain the minimal constraints.

We call this algorithm *Prop-STP;* pseudo-code is shown in Alg. 1. We now show that this algorithm returns all the minimal constraints of the original network.

Theorem 4. *At the end of Prop-STP, $\mathcal{T}^{2K-1}$ is locally minimal on every cluster V_k; furthermore, the subnetwork $\mathcal{T}^{2K-1}_{V_k}$ is a minimal network. As a result, $\mathcal{T}^{2K-1}$ is locally minimal on every variable i, and on every pair of variables $\{i, j\}$ for all (i, j) belonging to the same cluster.*

Example. Fig. 1 depicts an STN arising from a hierarchical planning problem. The edges labeled with letters refer to tasks; each models the duration between the start and end time-points of that task. For example, letter A refers to two time-points (variables) A_{start} and A_{end}. The other edges model precedence constraints between parent and child tasks, together with a representative sample of other temporal constraints. The STN has the sibling-restricted property, which provides a simple clustering of the variables to yield the join-tree $\mathcal{J}$ shown on the right. The separators of each cluster are depicted in the rectangular boxes. On this STN, the first pass of Alg. 1 processes the clusters in an order such as DJK, ELM, BDE, $CFGH$, ABC. Hence the cluster ABC is the root of the join-tree. The second pass processes $CFGH$, BDE, ELM, DJK. A single $lmin$ operation on BDE in the second pass effectively computes the two messages to DJK and ELM at once. At the end, we obtain the minimal subnetwork for each cluster, and thus the partial minimal network for the whole STN.

Analysis of the New Algorithm

Instantiations of Prop-STP can be implemented using different STN sub-solvers to perform the $lmin$ operation. For example, using PC-1 (Floyd-Warshall) leads to *Prop-PC1-STP*. Since there are precisely $2K - 1$ $lmin$ operations,[5] each with complexity $\mathcal{O}(w^3)$, the overall complexity of Prop-PC1-STP is $\mathcal{O}(Kw^3)$.

Since Prop-STP effectively completes the constraint graph within each cluster V_i, the resulting global constraint graph is triangulated. In practice, one can triangulate G in the initialization phase; if so, the constraint graph is already complete within each cluster. In this case, Prop-STP returns the partial minimal network $\mathcal{T}^{pmin}$ of the triangulated G. Like other algorithms that work on triangulated graphs, such as PPC and $\triangle$STP, Prop-STP does not return the full minimal network. However, $\mathcal{T}^{pmin}$ is sufficient for a triangulated graph since every solution can be constructed backtrack-free from the partial minimal network.

It is interesting to examine the behaviour of our algorithm when we use $\triangle$STP as the sub-solver at each cluster (although this is not likely to result in a performance improvement since the subgraph at each cluster is already complete). In this case, *Prop-$\triangle$-STP* will process triangles in G, one by one, but following the order imposed by our propagation scheme: all triangles within each cluster are processed until stabilized before moving on to triangles in another cluster.

[5] Improvement can be made by keeping track of which clusters have not been changed after the $lmin$ operations in the first pass, so that in the second pass we need not perform $lmin$ on them again.

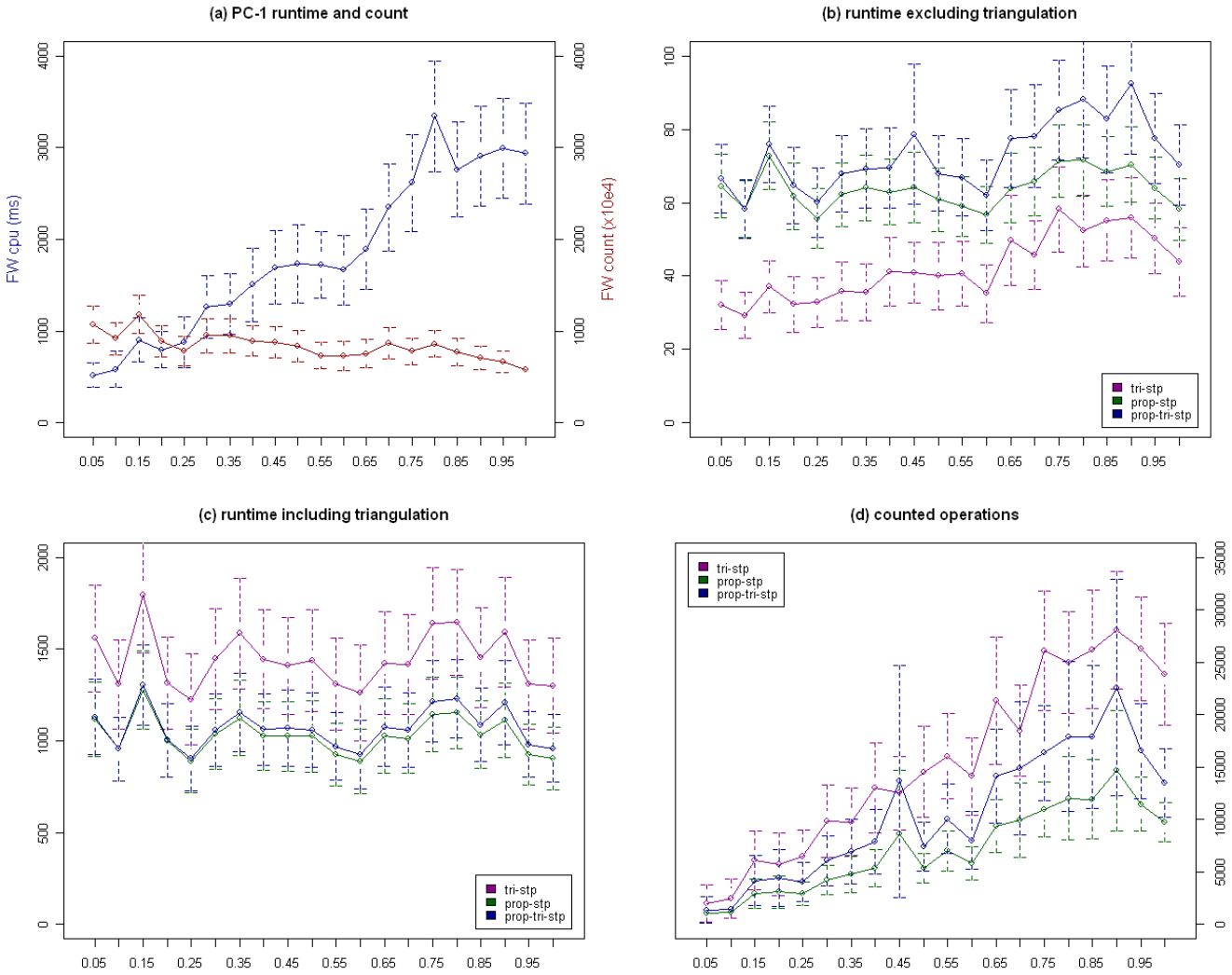

Figure 2: Landmark STNs, as consistency varies.

As noted by (Xu and Choueiry 2003), the order in which triangles are processed has a crucial effect on the performance of △STP. The improved order of triangle processing in our algorithm also agrees with the intuition of the authors of the △STP algorithm and others (Choueiry and Wilson 2006).

Experimental Results

We undertook experiments to investigate the behaviour of Prop-STP. We investigate the performance as the structure and size of the STN vary, and we compare Prop-STP against the existing STN solvers SR-PC and △STP to gain insight into their relative strengths. Since Prop-STP is designed to leverage structure in the network, in the form of a tree decomposition, we develop benchmarks based on structured STNs arising from Hierarchical Task Network (HTN) plans.

Sibling-Restricted STNs

HTN planning assumes a hierarchical flow, with high-level tasks being decomposed progressively into collections of lower-level tasks through the application of matching methods with satisfied preconditions. In a *sibling-restricted* STN, constraints may occur only between parent tasks and their children, and between sibling tasks. This restriction on what STN constraints may exist between plan elements is inherent to HTN planning models; in particular, there is no way in standard HTN representations to specify temporal constraints between tasks in different task networks (Erol *et al.* 1994). The specialized STN solver SR-PC (Yorke-Smith 2005) transverses a tree of sub-STNs that correspond to the decompositions in the HTN plan. Because the STNs thus considered are small, compared to the *global* STN corresponding to the whole plan, the overall amount of work to enforce PC is much less.

Despite its success on benchmark problems from the PASSAT plan authoring system (Myers *et al.* 2002), SR-PC has two drawbacks. First, standard HTN representations have been extended to support limited coordination between different task networks via *landmark variables* (Castillo *et al.* 2006) that allow synchronization of key events in the plan.

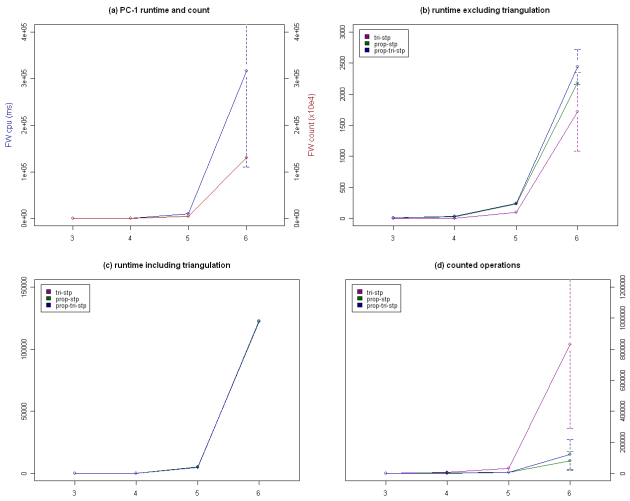

Figure 3: Landmark STNs, as depth varies.

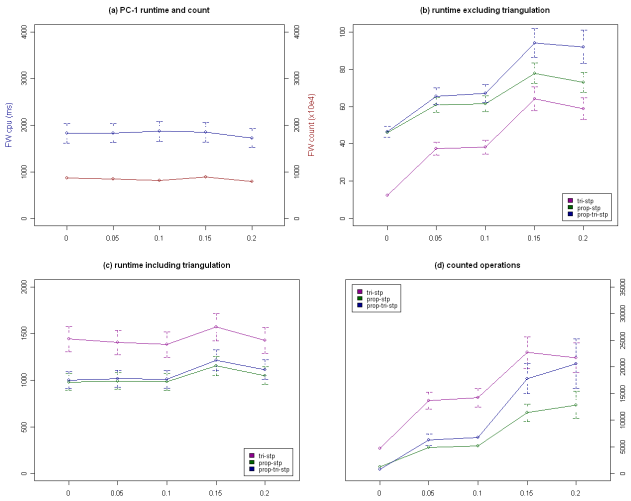

Figure 4: Landmark STNs, as landmark ratio varies.

SR-PC can accommodate, awkwardly, a small number of such landmark variables and their corresponding constraints. Second, although not exhibited in practice, SR-PC has poor and weakly characterized worst-case theoretical complexity.

The small tree-width w of an SR network means that Prop-STP will be particularly efficient for this class of STNs, provided we can find an optimal or near-optimal decomposition of the constraint graph G of the global STN into a join-tree, and a cluster ordering over this tree. Fortunately, there is a natural decomposition based on each parent task $\mathbf{T}$ and its children $\mathbf{T_i}$. Namely, we form into a cluster the start and end time-point variables of task $\mathbf{T}$ and all its $\mathbf{T_i}$, and all temporal relations between them (including those between the children). Fig. 1 illustrates this clustering. Since task networks typically comprise a handful of tasks, the size of each cluster is small. Therefore, performing minimalization on each cluster, as done in Prop-STP, is anticipated to be much more efficient than computing the minimal network for the global STN. In general, if we consider a task network represented by a balanced tree with depth d and branching factor f (hence the number of nodes is $\mathcal{O}(f^d)$), the complexity of Prop-STP is $\mathcal{O}(f^2 f^d)$ (the join-tree has $\mathcal{O}(f^{d-1})$ clusters, each with size $\mathcal{O}(f)$). We also note that an SR STN has no articulation points.Because each task in the HTN corresponds to two variables, as can be seen in Fig. 1, the constraint graph is biconnected and decomposes only via separator sets of cardinality at least two. This hampers algorithms that seek articulation points, whether explicitly such as *F-W+AP*, or implicitly, such as △STP.[6]

Experiments with Semi-Structured STNs

To validate the concept of Prop-STP, we implemented the algorithm within the PASSAT HTN plan authoring system. Our first Prop-STP implementation was written in Lisp to allow a fair comparison with the existing Lisp-based implementations of SR-PC and △STP, and to allow integration with the PASSAT system. However, as reported in the preliminary results in (Bui *et al.* 2007), this implementation of the three algorithms suffered from excessive CPU runtimes, an artifact due to the simplistic memory handling of our Lisp environment. Hence we developed a second implementation of the algorithms in Java; direct integration with PASSAT was not attempted.

The benchmark networks we employed were created by a generator that yields STNs akin to those produced by PASSAT, from random plans with a uniform tree of tasks, as described in (Yorke-Smith 2005). The generation parameters include plan size (specified by HTN depth and mean branching factor), constraint likelihood, constraint tightness, ratio of time-points that are landmarks, and STN consistency.

The experiments were conducted on a 2GHz Pentium M with 2GB RAM, using Java 1.6. A timeout of 10 minutes was enforced. Unless stated, the results average 50 runs, and error bars depict 95% confidence intervals.

The algorithms we report are PC-1 as a baseline, △STP (with and without triangulation time), and Prop-STP. △STP source code was kindly made available by its authors. When `landmarkratio` is 0, the STN is SR, and SR-PC can be applied. We verified that Prop-STP behaves as SR-PC in this case, when given the decomposition of the network; its runtime and constraint checks equal those of SR-PC.

We gave the same networks to △STP and Prop-STP, and did not provide any additional information about structure to Prop-STP. Information about the STN's construction from the HTN structure and landmark variables would enable decomposition with the addition of a minimal number of triangulation fill and cluster completion edges. Prop-STP would benefit from this information (which is not pertinent to △STP); thus, we held back Prop-STP from its full potential. This decision allows us to compare Prop-STP in the 'worst case' and assess its behaviour on STNs where the structure is not known a priori. Throughout, the same triangulation heuristics are used for all algorithms.

[6]Observe that Prop-STP can work on the untriangulated constraint graph with this clustering. To apply a generic STN solver such as △STP in this case requires triangulation of the whole constraint graph G, which in general introduces many new edges (many more than the total in all the completed $G^{(i)}$).

We experimented with two variants of Prop-STP, one using PC-1 and the other using △STP as the STN sub-solver. Recall that in Alg. 1, the subgraph at each cluster is a complete clique, and that such dense graphs are those most favourable to PC-1 (Shi *et al.* 2004).

We recorded two metrics: first, the total CPU runtime in ms for each algorithm variant,[7] and second, the counted number of constraint checks.

Consistency. Fig. 2 varies the consistency of the generated STN from 5 to 100%, for problems of mean depth 5 and mean branching factor 3.5 (approximately 500 nodes), and `landmarkratio` of 5%. Note that the y-axis scale varies between the graphs. PC-1 is the most sensitive to the consistency of the network, taking considerably longer for consistent STNs than for inconsistent but, curiously, performing fewer constraint checks for the former. There is little to distinguish between △STP, Prop-PC1-STP, and Prop-△-STP in terms of runtime when triangulation time is excluded; when included, △STP is slightly slower. In terms of counted operations, all three perform more constraint checks for consistent STNs. △STP is the most sensitive to consistency, Prop-PC1-STP the least, and Prop-△-STP blends the two, as could be expected.

Problem size. Fig. 3 varies STN size, in terms of underlying plan depth, from 3 to 6; consistency is 50%, `landmarkratio` is 5%, and mean branching factor is 4. The size of the STN grows exponentially with depth. PC-1 and △STP suffer from timeout or memory exhaustion respectively.[8] Fig. 3 includes data only for those instances for which every algorithm succeeded; we exclude data for depths 7 and 8, where often only Prop-STP succeeded. Even in the limited data presented, we see all algorithms increasingly exponentially with depth, i.e., polynomially in problem size. As in Fig. 2, we observe that the runtime metrics for △STP and Prop-STP are similar, while △STP employs more constraint checks.

Structure. Fig. 4 varies the landmark ratio from 0 to 20%, for problems of depth 5 and consistency of 50%. A low value for `landmarkratio` corresponds to a structured (more SR-like) STN; a higher value corresponds to a more unstructured STN, closer to a general random network. A value even of 10% leads to a very dense network after triangulation. Note again the y-axis scales.

While PC-1 is largely insensitive to the `landmarkratio` parameter (network consistency being more significant), △STP and Prop-STP exhibit greater runtime as structure decreases. Most clearly this is seen in runtime excluding triangulation and in counted operation: the triangulation time masks the underlying computational effort. The reason that Prop-STP does not perform significantly better than △STP lies in our giving it no information about network structure. As observed earlier, Prop-△-STP (in particular) explains the theoretical performance of △STP with optimal triangle ordering. Since we apply both methods to the same triangulated graph with the same heuristics, we should expect to observe similar performance in practice in this case.

[7] The recorded time for Prop-STP's 'no triangulation' variant still includes the time to complete each cluster.

[8] The difficulty for △STP is not the triangulation of G as much as storing the list of all triangles; in contrast, although we make Prop-STP naively work over the same triangulated graph G', only G' itself is required, not the list of all triangles.

Discussion

The picture that emerges for semi-structured STNs is that Prop-STP performs comparably with the state-of-the-art solver △STP. The empirical findings confirm that it scales polynomially with problem size. The more structured the network, the better the relative performance, and the better still if Prop-STP is informed about network structure, as in the experiments reported in (Bui *et al.* 2007) for SR-STNs. Given no structure, Prop-STP performs best on sparse networks, since a tree decomposition is more amenable in general the fewer the edges in the triangulated graph. Empirically, on our benchmark semi-structured STNs, the join-tree uncovered has average cluster size approximately equal to the $1 +$ `branchingfactor`. Approximately described, the behaviour of Prop-△-STP exhibits characteristics of the two solvers from which it is composed.

It is worth noting that our implementation of Prop-STP contains no effort to optimize the network decomposition, nor leverage reuse of, for example, triangulated clusters. The benefit of the set of smaller sub-problems that Prop-STP tackles comes at the cost of computing the decomposition into clusters and exchanging the messages between them; the number and size of the separator sets is key. A carefully engineered implementation can be expected to reduce the runtime by a constant factor. Thus the pattern for relative CPU times reported above can be expected to match the pattern for counted operations: a modest runtime advantage including triangulation time for Prop-STP over △STP provided that the network has sufficient structure (i.e., `landmarkratio` is small). This is true even though we provide no structural information to Prop-STP.

The tree-decomposition and clustering that benefits Prop-STP on sparse and structured networks is expected to hinder it on dense and unstructured networks. For completeness, we experimented on general STNs, randomly created by the generator of (Xu and Choueiry 2003). The results, not reported here, indeed suggest that the computational overhead of Prop-STP puts it at a disadvantage compared to △STP in terms of runtime. We also confirm the results in the literature that △STP is most effective for sparse networks (Shi *et al.* 2004). As in the case of structured networks, we find that Prop-PC1-STP is relatively insensitive to the constrainedness, while the behaviour of Prop-△-STP is again a blend of the two solvers from which it is composed.

Conclusion and Future Work

This paper considers a new method, *Prop-STP,* for solving Simple Temporal Networks in the case of continuous as well as discrete variable domains. In contrast to methods based on graph algorithms or on iteration of narrowing operators, our algorithm is based on an efficient message passing scheme over the join-tree of the network. The complexity of Prop-STP depends on the minimalization opera-

tor, i.e., the STN solver used to enforce path consistency on sub-problems. Thus consistency and the minimal constraints of an STN (from which solutions can be derived backtrack-free) can be determined with complexity $\mathcal{O}(Kw^3)$ or better, where K is the number of cliques and w is the induced tree-width. For STNs with known and bounded tree-width, Prop-STP thus achieves linear time complexity. The new propagation scheme provides formal explanation of the performance of the existing STN solvers $\triangle$STP and SR-PC. For $\triangle$STP, the scheme also provides an efficient triangle ordering based on the join-tree clusters.

Our motivation comes from the sibling-restricted STNs that arise in HTN planning problems. Prop-STP is well-suited to such STNs because these problems (1) have a small tree-width w, and (2) the SR structure leads to a simple way to decompose the network into a join-tree. Prop-STP generalizes the best-known solver, SR-PC, for this class of problems. It avoids the poor worst-case complexity of SR-PC, and it can accommodate landmark variables in SR-STNs. At the same time, empirical results have validated that Prop-STP retains the efficiency of SR-PC on problems that the latter can solve. Moreover, they indicate that even in the absence of any structural information, Prop-STP parallels the performance of $\triangle$STP on semi-structured STNs.

More than validation of an alternate algorithmic approach, a further contribution of this paper is insight into the behaviour of SR-PC and $\triangle$STP. Nonetheless, a carefully engineered implementation of Prop-STP can be recommended for sparse STNs (even unstructured) and structured STNs where a near-optimal tree decomposition and cluster ordering is known a priori or can be determined by automated analysis of the network.

Planning practitioners have long sought efficient temporal reasoning in planning systems. Besides exploiting structure in the temporal network underlying a plan, in a different and complementary vein, the casual structure of the plan can also be leveraged to reduce the effort of temporal propagation (Castillo *et al.* 2006). The most important direction for future work, however, is to employ Prop-STP in incremental STN solving, where time-points and constraints are added or removed incrementally; incremental STN solving arises naturally in planning. The challenge is to adapt Prop-STP, as other algorithms have been adapted, to efficiently solve the modified STN, given the result of computation from solving the previous STN.

An interesting avenue to explore is the quality of the decomposition versus the time for its computation (Jégou *et al.* 2005). Quality can be characterized not only in terms of the size of the join-tree and its clusters — according to which metrics, an optimal decomposition is simple to derive — but also in terms of the total effort, cumulative over decomposition time and subsequent solving. What if Prop-STP were to operate over a 'non-optimal' decomposition with, for instance, larger clusters than the 'optimal'?

Another interesting avenue is to consider temporal constraint networks that have preferences associated (Khatib *et al.* 2001). The challenge in applying variable elimination to STNs with preferences is to compactly represent the messages and tractably compute the minimialization operator, especially over continuous preference domains, as we have done in the case of STNs without preferences.

Acknowledgment. We gratefully thank Berthe Choueiry, Karen Myers, and Bart Peintner for discussions and insights, the reviewers for their comments, and the participants of the AAAI'07 Workshop on Spatial and Temporal Reasoning, where a preliminary report of this work was presented.

References

C. Bliek and D. Sam-Haroud. Path consistency on triangulated constraint graphs. In *Proc. of IJCAI'99*, 1999.

J. Bresina, A. K. Jónsson, P. Morris, and K. Rajan. Activity planning for the Mars exploration rovers. In *Proc. of ICAPS'05*, 2005.

H. H. Bui, M. Tyson, and N. Yorke-Smith. Efficient message passing and propagation of simple temporal constraints. In *Proc. of AAAI 2007 Workshop on Spatial and Temporal Reasoning*, 2007.

L. Castillo, J. Fdez-Olivares, and F. Palao O. García-Pérez. Efficiently handling temporal knowledge in an HTN planner. In *Proc. of ICAPS'06*, 2006.

B. Y. Choueiry and N. Wilson. Personal communication, February 2006.

T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

R. Dechter and J. Pearl. Tree clustering schemes for constraint-processing. *Artificial Intelligence*, 38(3):353–366, 1989.

R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1–3), 1991.

R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

K. Erol, J. Hendler, and D. Nau. Semantics for hierarchical task-network planning. Technical Report CS-TR-3239, University of Maryland, 1994.

P. Jégou and C. Terrioux. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, 146(1):43–75, 2003.

P. Jégou, S. Ndiaye, and C. Terrioux. Computing and exploiting tree-decompositions for solving constraint networks. In *Proc. of CP'05*, 2005.

L. Khatib, P. Morris, R. A. Morris, and F. Rossi. Temporal constraint reasoning with preferences. In *Proc. of IJCAI'01*, 2001.

U. Kjaerulff. Triangulation of graphs: Algorithms giving small total state space. Technical Report R90-09, Aalborg University, Denmark, 1990.

P. Laborie and M. Ghallab. Planning with sharable resource constraints. In *Proc. of IJCAI'95*, 1995.

J. Larrosa, E. Morancho, and D. Niso. On the practical use of variable elimination in constraint optimization problems: 'Still-life' as a case study. *J. Artificial Intelligence Research*, 23:421–440, 2005.

K. L. Myers, M. W. Tyson, M. J. Wolverton, P. A. Jarvis, T. J. Lee, and M. desJardins. PASSAT: A user-centric planning framework. In *Proc. of the Third Intl. NASA Workshop on Planning and Scheduling for Space*, 2002.

Y. Shi, A. Lal, and B. Y. Choueiry. Evaluating consistency algorithms for temporal metric constraints. In *Proc. of AAAI-04*, 2004.

D. E. Smith, J. Frank, and A. K. Jónsson. Bridging the gap between planning and scheduling. *Knowledge Eng. Review*, 15(1):47–83, 2000.

L. Xu and B. Y. Choueiry. A new efficient algorithm for solving the simple temporal problem. In *Proc. of TIME'03*, 2003.

N. Yorke-Smith. Exploiting the structure of hierarchical plans in temporal constraint propagation. In *Proc. of AAAI-05*, 2005.