

STCSP - Structured Temporal Constraint Satisfaction Problems¹

Mira Balaban
Information Systems Program
Department of Industrial & Management
Engineering
Ben-Gurion University
Beer-Sheva, ISRAEL
mira@cs.bgu.ac.il

Tzachi Rosen
Department of Mathematics and
Computer Science
Ben-Gurion University
Beer-Sheva, ISRAEL
tzachi@cs.bgu.ac.il

Abstract

Temporal Constraint Satisfaction Problems (TCSP) is a well known approach for representing and processing temporal knowledge. Important properties of the knowledge can be inferred by computing the *minimal networks* of TCSPs. *Consistency* and *feasible values* are immediately obtained; computing *solutions* can be assisted. Yet, in general, computing the minimal network of a disjunctive TCSP is intractable.

The minimal network approach requires computation of the full network in order to answer a query. In this paper we characterize TCSPs for which subsets of the minimal network can be computed without having to compute the whole network. The partial computation is enabled by decomposition of the problem into a tree of sub-problems that share at most pairs of time points. Such decompositions are termed *sim/2-tree decompositions*. For TCSPs that have sim/2-tree decompositions, minimal constraints of input propositions can be computed by independent computations of the minimal networks of the sub-problems at most twice. It is also shown that the sim/2-tree characterization is a minimal set of conditions. The sim/2-tree decomposition extends former results about decomposition of a TCSP into bi-connected components. An algorithm for identifying a sim/2-tree decomposition of a TCSP is provided as well. Finally, the sim/2-tree decomposition is generalized in an inductive manner, that enables components of a decomposition to be further decomposed. For that purpose a model of Structured Temporal Constraint Satisfaction Problems ($STCSP^{(n)}$, $0 \leq n$), where $STCSP^{(0)}$ is simply TCSP, $STCSP^{(1)}$ is a set of $STCSP^{(0)}$ s, and in general, $STCSP^{(n)}$ for $1 \leq n$, is a set of $STCSP^{(n-1)}$ s, is introduced.

¹ This work was supported in part by the Paul Ivanir Center for Robotics and Production Management at Ben-Gurion University of the Negev.

1 Introduction

Temporal processing is central to many areas in Computer Science, like Signal Processing, Database Management, Real Time Systems and Artificial Intelligence. In AI, research in temporal reasoning began primarily with the works of Allen, Hayes, and McDermott [Allen83, Allen84, AH85, McDermott82]. Allen studied *qualitative temporal constraints*, where time intervals are related by relationships like being before, intersecting or after in time. His work was followed by an intensive study of qualitative and metric constraint systems. The work of Allen, Hayes and McDermott initiated research into common sense reasoning about time (for example [BTK91] and [Shoham86]), which is essential for typical AI tasks like planning. In general, it is commonly agreed that temporal aspects are central to the design of intelligent systems, since they operate, plan and reason about a world that changes with time. Research on temporal databases has also flourished in recent years. Major subjects for research are data modeling ([BK96, CW83, MTM91, WD92), query languages ([Ariav86]), implementation ([JMR90]), and management ([Dean89]).

Intelligent systems that process temporal information deal with issues like the temporal consistency of the information, and provide services for reasoning about time. One important task of such systems has to do with solving temporal constraints. The data for that task. consists of a set of temporal events, such as "my last trip to Jerusalem" or "your birthday party". Each event is associated with a time slice with a beginning point and an ending point (not necessarily disjoint). The timing of the events is constrained by a set of temporal propositions on the beginning and ending points. The propositions may specify qualitative relations between time intervals [Allen83, LR92, VanBeek92, GS93], or quantitative relations between time points [DMP91] or both [Meiri96, JB97, KL91]. They may be given implicitly by some kind of cause-effect rules [BCS92, Shoham86], or explicitly [Allen83, DM87].

Dechter, Meiri and Pearl, in [DMP91], propose to use constraint network formalisms for implementing quantitative temporal constraint systems. Their approach, called *Temporal Constraint Satisfaction Problems (TCSP)*, is demonstrated in Example 1.

Example 1: A TCSP.

A bus line and a train line act as a connected line. The bus goes from point A to point B, leaving its station every 105 minutes, starting at 6 a.m. and operating until 8 p.m.. The train goes from point B to point C, leaving its station every half an hour, around the clock. Depending on the traffic, the bus drive takes between 110 to 115 minutes. The train drive always takes 100 minutes.

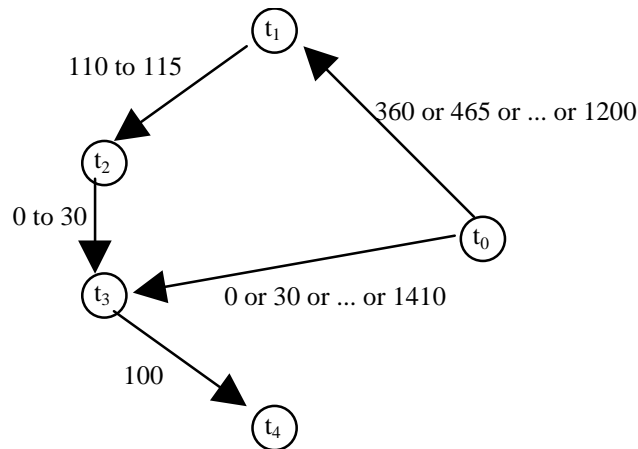
A TCSP modeling of this problem suggests methods for testing consistency of the given information and reasoning about its temporal content. For example, a TCSP system can answer questions such as:

- Is the information temporally consistent, i.e., temporally possible?
- When should a person leave point A in order to arrive at point C no later than 3 p.m.?
- For a person that arrives point A at 11 a.m., how much time it can take to reach point C?
- What is the minimal time of waiting to a train at point B?
- What are the possible arrival times at point C for people starting at point A?

In order to model the problem as a TCSP we need to single out its temporal events, their delimiting time points, and the temporal constraints imposed between such time points. The first event spans the time that passes between midnight and a departure of a bus from point A. This event can take either 6 hours or 7 hours and 45 minutes, or 9 and a half hours, and so on until 20 hours. We denote by t_0 the beginning time point of this event, i.e., midnight, and by t_1 its ending point. The time point t_1 happens to be the beginning of the event that describes any bus drive from point A to point B. The time interval of this event is between 110 minutes to 115 minutes. Its ending point, denoted t_2 , is also the beginning of a waiting event at point B, which can take between 0 to 30 minutes. The ending time point of this event, denoted t_3 , is the beginning of a train drive event from point B to point C, that takes 100 minutes. Additional information, given in the problem description, delimits the time interval between midnight (t_0)

and a train departure from point B (t_3), to be either 0 minutes or 30 minutes or 1 hour, and so on until 23 hours and 30 minutes.

Temporal constraint systems can be graphically described by representing time points as nodes, and temporal events as arcs, labeled by temporal constraints. The following figure captures the bus-train problem:



□

Quantitative temporal constraint problems usually include different kinds of temporal propositions. *Simple quantitative propositions* (*simple propositions*, in short) express constraints such as "the time interval between t_1 to t_2 is between 110 to 115 minutes". *Disjunctive quantitative propositions* (*disjunctive propositions*, in short) consist of collections of simple propositions such as "the time interval between t_0 to t_1 is either 360 or 465 or 1200 minutes". *Comparative quantitative propositions* such as "at least 60 minutes between t_i to t_j " specify only the least or upper bound of a time intervals between points. In the TCSP approach such propositions are modeled as quantitative propositions where the missing interval bound is taken either as the greatest time point (denoted ∞) or the least time point (denoted $-\infty$), respectively. *Comparative qualitative propositions* such as " t_i occurs before t_j " or " t_i occurs no later than t_j " are modeled in the TCSP approach as comparative quantitative propositions with numerical bounds that express the qualitative relations. For example, " t_i occurs no later than t_j " is understood as "at least 0 minutes between t_i and t_j ". *Absolute propositions*, such as " t_i occurs either at 6.00am or at 7.00am", are modeled as quantitative propositions with respect to a time point that stands for the first time point in the problem domain. In Example 1, midnight (t_0) is taken as such time point. In sum, all propositions in the problem specification are modeled as quantitative propositions, simple or disjunctive.

Dechter et. al., in [DMP91], introduce the concept of a *minimal network* (or a *minimal problem*) as a means for answering questions about TCSPs. Intuitively, a *minimal problem* of a TCSP is the equivalent reduced TCSP. Finding the minimal problem of a TCSP with simple propositions is tractable. Several such algorithms, all cubic in the number of time points, are given in the literature ([DMP91]). The algorithms are based on Floyd's "all shortest paths" algorithm. They compute the minimal allowed distances between all time points of a problem, starting with the graph of the maximal possible distances. In general, finding minimal networks of TCSPs is NP-hard. The standard process, for a problem P , consists of finding the minimal networks of the simple problems induced from P , and combining the results into a single disjunctive problem. The number of the simple problems of P can be exponential in the number of arcs (n^2 , for a problem with n time points).

Since finding the minimal network for general disjunctive TCSPs is intractable, researchers proposed efficient approximation algorithms [SD97]. Another approach involves structure-based decomposition of a disjunctive TCSP into smaller component problems, such that the problem can be solved, or interesting questions can be answered, by computing the minimal problems of the components, while avoiding the computation of the minimal network of the whole problem. We term this approach *minimization-by-parts*. In that case, the time complexity of the overall process may reduce to the time complexity of computing the minimal networks of the components. Dechter et al., in [DMP91], suggest structure decomposition by

articulation points. For general Constraint Satisfaction Problems (CSPs), various decomposition techniques have been studied (e.g., [DP88], [DP88a], [Freuder85], [GJC94]).

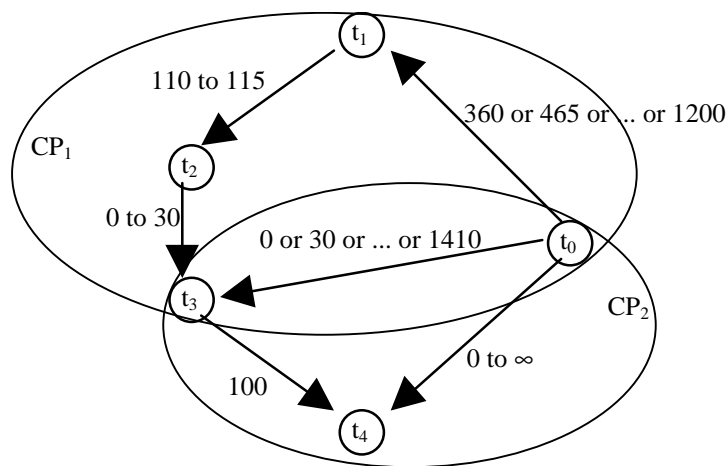
The minimal network approach of [DMP91] is a global method. If a query is focused only on a sub-problem that is characterized by a subset of the time points, it is still necessary, in general, to compute the whole minimal network and then project it on the sub-problem. The main question addressed in this paper is as follows: If we are given a TCSP and a characterization of sub-problems of interest (i.e., TCSPs defined on subsets of the time points), under what conditions can we compute the restriction of the minimal network to these sub-problems, in a way that is more efficient than the brute-force approach. In [DMP91] it was shown that this is indeed possible if the problem graph has a decomposition into bi-connected components.

A special case of the problem raised in the previous paragraph is when only original propositions are of interest, namely we are interested only in the minimal constraints of those relationships that are stated in the input TCSP, and are not interested in inferring the minimal constraints between time points that are not explicitly constrained (i.e., are universally constrained). We demonstrate this point with respect to the problem of Example 1.

Example 2: Suppose that we wish to answer the following query about the problem of Example 1:

"What are the possible arrival times at point C for people starting at point A?"

The minimal constraint between time points t_0 and t_4 provides the exact answer. So we add this constraint as an explicit input proposition, and try to compute the minimal constraints of the input propositions alone. We notice that the problem can be naturally decomposed into two sub-problems: CP_1 -- the bus drive, that includes time points t_0, t_1, t_2, t_3 , and the propositions among them, and CP_2 -- the train drive, that includes time points t_0, t_3, t_4 , and the propositions among them. The graphical description of the decomposition is as follows:



As we show later in this paper, indeed we can answer the query by computing the minimal networks of the two components alone. Moreover, we can even further decompose CP_1 and obtain its minimal constraints without computing its full minimal network.

□

Decomposition of TCSPs poses three major difficulties: *Termination*, *fragmentation*, and *composition*.

1. A termination problem might arise if two components share more than a single time point. In that case, the composition of the minimal problems of the components may be not straightforward, since the computation of the minimal network of one component might affect its neighbor components, and require repeated computations. Such iterations may propagate forward and backward until all neighbor components agree on their intersection propositions. The process may loop forever as well.
2. Fragmentation occurs when a single interval of a simple proposition is broken into several intervals during the computation of the minimal network of a non-simple problem.

Consequently, repeated computations might yield problems of growing complexity. Fragmentation of time intervals is studied in [DMP91], [PB91], and [SD97].

3. A composition problem arises when separate computations of the minimal networks of the components of a TCSP decomposition, do not yield minimal constraints of the overall problem. That is, it is possible that the minimal networks of the components agree on their shared propositions, thereby avoiding the termination problem, and still the combination of the minimal networks is not a projection of the overall minimal network on the components. Clearly, a decomposition that cannot guarantee composition is not interesting. We note that a composition problem might arise even in tree structured decompositions.

In this paper we study the *composition* and the *termination* problems of TCSP decomposition. We characterize TCSP decompositions that enable to compute the minimal constraints of input propositions by separate computations of the minimal networks of their components, with at most two repeated computations per component. That is, the characterization guarantees composition and termination. Its conditions are set on the structure of the TCSP decomposition, and can be syntactically verified. We show that the conditions form a minimal set of conditions. For problems that satisfy the conditions we provide an algorithm for computing the minimal networks of the components. If there is no fragmentation, the complexity of computing minimal constraints of input propositions is reduced to the complexity of computing minimal networks of the components. In any case, the complexity does not increase, and the benefit of partitions by articulation points are preserved. Finally we generalize the characterization of TCSP decompositions in an inductive manner, that enables components of a decomposition to be further decomposed. We do it by introducing a model of Structured Temporal Constraint Satisfaction Problems ($STCSP^{(n)}$, $0 \leq n$). $STCSP^{(0)}$ is simply TCSP, $STCSP^{(1)}$ is a set of $STCSP^{(0)}$ s, and in general, $STCSP^{(n)}$ for $1 \leq n$, is a set of $STCSP^{(n-1)}$ s.

The main contribution of this work is in extending the result about decomposition of a TCSP into bi-connected components, to a tree of sub-problems that share at most pairs of variables, and in recognizing that extensions to trees that share more variables may become inherently more complex. The paper also provides an algorithm for identifying a decomposition of a TCSP into a tree of sub-problems sharing at most pairs of variables. The origin of this work is in the M.Sc. thesis of Rosen [Rosen95].

Section 2 provides the relevant concepts and results for TCSPs, and introduces the new concept of *graph-minimization* which is computing the projection of the minimal network on the input propositions. TCSP decompositions are presented and studied, in section 3, including the conditions for *graph-minimization in parts*, i.e., computing the projection of the minimal network on the input propositions, by separately computing the graph-minimal problems of the components. Section 3 also includes the algorithm for deriving a TCSP decomposition that satisfies the conditions for graph-minimization in parts. Section 4 introduces the $STCSP^{(n)}$ model. Section 5 is the conclusion.

2 Preliminaries

2.1 TCSP -- Definitions, Concepts, and Properties

A TCSP is a set of propositions that temporally constrains a collection of binary events. A *binary event* is an ordered pair of *time points*, called *beginning* and *ending*. The propositions constrain the temporal distance between the points of the events. The task is to suggest a set of events that satisfy the propositions. Each such set comprises a solution. The problem is inconsistent if it has no solution.

Definition 2.1 -- TCSP:

Let D be a domain of time points. We take D as the set of real numbers. For $a \leq b$, an interval $[a, b]$ over D is the set $\{d \mid a \leq d \leq b\}$ in D . If $a = b$, $[a, b]$ is denoted $[a]$.

A *TCSP* is a pair $\langle X, \Psi \rangle$, where

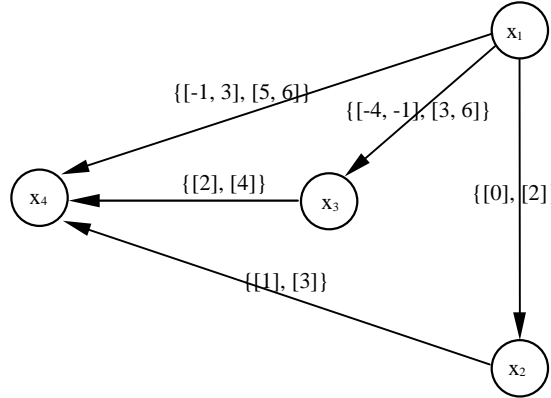
1. X is a set of *variables* $\{X_1, \dots, X_n\}$, each identified with a distinguished beginning or ending point of an event in the problem domain.

2. Ψ is a set of *binary propositions*. A binary proposition is a *binary constraint*, i.e., a set of *intervals* over D , that constrains the temporal distance between two variables, to belong to one of the intervals. A proposition that sets the binary constraint C on the variables X_i and X_j is denoted C_{ij} .

□

A TCSP can be described graphically by representing the variables as vertices, and the temporal propositions as labeled edges. C_{ij} is described by a directed edge from X_i to X_j , labeled C . Example 3 shows a TCSP with four variables X_1, X_2, X_3 and X_4 . The propositions in the problem are: $\{[0], [2]\}_{1,2}, \{-4, -1], [3, 6]\}_{1,3}, \{-1, 3], [5, 6]\}_{1,4}, \{[1], [3]\}_{2,4}, \{[2], [4]\}_{3,4}$.

Example 3: A TCSP.



□

Definition 2.2 -- Semantics of TCSP: Given a TCSP $P = \langle X, \Psi \rangle$.

1. A proposition C_{ij} in Ψ is satisfied by an assignment $\{X_1 = t_1, \dots, X_n = t_n\}$, if $t_j - t_i \in C$.
2. A solution is a tuple (t_1, \dots, t_n) such that the assignment $\{X_1 = t_1, \dots, X_n = t_n\}$ satisfies every proposition in Ψ . The value t_i in a solution $(t_1, \dots, t_i, \dots, t_n)$ is a *feasible value* of X_i .
3. A TCSP is *consistent* if it has a solution.
4. A time point t is a *feasible value of a proposition* C_{ij} in Ψ , if there exists a solution such that $t_j - t_i = t$.

□

The assignment $\{X_1 = 0, X_2 = 2, X_3 = 3, X_4 = 5\}$ satisfies all propositions of Example 3, and 0, 2, 3 and 5 are feasible values of X_1, X_2, X_3 and X_4 , respectively. Hence, the tuple $(0, 2, 3, 5)$ is a solution of the problem, and the problem is consistent. The time points 2, 3, 5, 3 and 2 are feasible values of the propositions $\{[0], [2]\}_{1,2}, \{-4, -1], [3, 6]\}_{1,3}, \{-1, 3], [5, 6]\}_{1,4}, \{[1], [3]\}_{2,4}, \{[2], [4]\}_{3,4}$, respectively.

A *temporal constraint* is any set of time intervals; it can be empty, finite or infinite. The time intervals are non-empty and closed. Each time interval stands for a *time slice*, i.e., a "continuous" portion of the set of time points. The temporal constraint $\{[3, 5], [4, 6], [6, 7]\}$, denotes a single time slice as well. A temporal constraint, in general, denotes several (disjoint) time slices. A constraint that denotes a single time slice is called *simple* and a proposition with a simple constraint is a simple proposition. A *simple TCSP* is a problem with simple propositions alone. Otherwise it is non-simple. It is well known ([DMP91]) that deciding the consistency of non-simple TCSPs is NP-complete. For a simple TCSP with n variables, consistency can be decided in $O(n^3)$. A proposition with an empty constraint is not satisfiable.

A *simple case* of a TCSP P is obtained from P by dropping in every proposition of P all intervals but one. A *union of problems* is a problem on the same set of variables, whose propositions are the union of the corresponding propositions in the input problems. A non-simple TCSP can be viewed as the union of its simple cases. Two problems with a common set of variables are *equivalent*, if they have the same set of solutions.

A TCSP is expected to provide the following services:

1. Find whether the problem is consistent.
2. Find the feasible values of the propositions and of the variables.
3. Find a single solution or all solutions.

The notion of a *minimal problem* of a TCSP, which intuitively is the equivalent reduced TCSP, is introduced in [DMP91], as a means for fulfilling the first two tasks. As for the third service, for a simple minimal problem, a solution can be computed in $O(n^2)$; for a non-simple minimal problem, finding a solution might be intractable. The minimal problem can be viewed as a compact way for storing all solutions. The definition of a minimal problem requires a means for comparison of TCSPs:

Definition 2.3 -- Comparable and Equivalent TCSPs, the Tighter than relation, and Complementation of TCSPs:

TCSPs P_1 and P_2 are *comparable* if they have the same set of variables, and their propositions are defined on the same pairs of variables. Given two TCSPs P_1 and P_2 (not necessarily comparable), P_1 and P_2 are *equivalent* ($P_1 \equiv P_2$), if they have the same set of solutions. Given two comparable TCSPs P_1 and P_2 , the proposition C^1_{ij} of P_1 is *tighter than the proposition* C^2_{ij} of P_2 ($C^1_{ij} \subseteq C^2_{ij}$) if $C^1 \subseteq C^2$. P_1 is *tighter than* P_2 ($P_1 \subseteq P_2$), if for all $1 \leq i, j \leq n$, $C^1_{ij} \subseteq C^2_{ij}$. The *complemented problem* of a TCSP P , **complement**(P), is a TCSP that is obtained from P by complementing all "missing" propositions C_{ij} in P (missing edges in the graph) as the *universal propositions* $\{[-\infty, +\infty]\}_{ij}$.

□

Definition 2.4 -- Minimal TCSP:

The *minimal problem* of a TCSP P , denoted **min**(P), is the tightest problem among all equivalent and comparable problems of **complement**(P). A problem P is *minimal* if $P = \mathbf{min}(P)$.

□

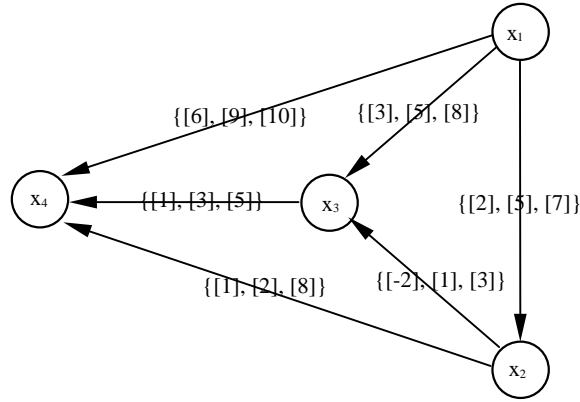
Note that the minimal problem always exists since equivalent TCSPs are closed under intersection, and is unique by its definition.

2.2 Solving TCSPs

In a minimal problem, testing consistency is immediate, since it amounts to testing whether any proposition is empty [DMP91] (if a proposition C_{ij} is empty, then no assignment of values to the variables X_i and X_j can satisfy the binary constraint C which is the empty interval). Feasible values of propositions are already given in the propositions (since if a proposition allows a non-feasible value, then that value can be removed in contradiction to the minimality of the problem). Feasible values for variables can be obtained by first assigning the time point 0 to an arbitrary variable X_k . The feasible values of the variable X_i ($1 \leq i \leq n$) are the allowed values of C_{ki} .

A solution for a simple minimal TCSP can be computed in a backtrack free manner, by anchoring a variable at a time point, and extending the solution, point by point, such that a new point is always consistent with points already in the solution. This is possible since a simple minimal TCSP has the property that every partial solution can be extended into a full solution. Hence, the solution can be obtained in time $O(n^2)$ [DMP91]. If we can find a solution in that way, we say that the problem is *backtrack free* (Also, *k-consistent, for every k*, in the general CSP terminology). Non-simple minimal problems are not necessarily backtrack free, as demonstrated in Example 4. Consequently, finding a solution might lead to exhaustive search over all simple cases. Yet, the minimal problem might help in filtering out some impossible combinations.

Example 4: A non-simple minimal problem that is not backtrack free.



It can be checked directly that every value allowed by any proposition is feasible. Hence, the problem is minimal. Yet, the partial solution (0, 2, 3) assigned to X_1 , X_2 and X_3 , respectively, cannot be extended to a full solution. Therefore, the problem is not backtrack free.

□

The minimal problem of a TCSP P is the tightest problem among all equivalent and comparable problems of **complement**(P). Hence, a problem is minimal if and only if its propositions allow only feasible values. Such propositions are called *minimal propositions*. Proposition 1 provides a criterion for minimality. An equivalent criterion, based on the notion of a distance graph, is provided in [DMP91].

Proposition 1: In a simple problem, all propositions are minimal iff every proposition C_{ij} is tighter than or equal to all indirect propositions between X_i and X_j .

Proof: In the Appendix.

□

In simple problems, since a proposition consists of a single interval, the All-shortest-path-algorithm of Floyd [AHU75], can be adapted for computing the tightest propositions between all pairs of time points. This process is given in Algorithm 1 (a similar algorithm, applied to the distance graph, is given in [DMP91]). Note that since the process does not divide intervals, the minimal problem of a simple problem is also simple. In a non-simple problem P , the minimal problem can be constructed from the minimal problems of the simple cases of P .

Algorithm 1: Min-of-Simple

input: A simple TCSP P .

output: $\min(P)$

time: $O(n^3)$, for a problem with n variables.

method: 1. Complement P .

2. For every time variable x_i ,
For every proposition C_{jk} ,

$$C_{jk} \leftarrow C_{jk} \cap (C_{ji} \oplus C_{ik})$$

where \oplus stands for proposition composition, defined as follows:

1. For intervals $[a, b]$, $[c, d]$: $[a, b] \oplus [c, d] = [a+b, c+d]$.
2. For propositions C_{ij} , C_{jk} : $C_{ij} \oplus C_{jk} = \{ [a, b] \oplus [c, d] \mid [a, b] \in C_{ij}, [c, d] \in C_{jk} \}_{jk}$

□

2.3 A Graph Based Minimal Problem

The minimal problem provides explicit minimal constraints between all time points, including points that are originally only implicitly constrained. This is done by first explicitly complementing the given problem with universal propositions. For instance, in Example 3, the process starts with complementing the missing constraint between X_2 and X_3 with the universal proposition $\{[-\infty, +\infty]\}_{2,3}$. Computing such minimal constraints may be redundant since they might be not relevant for the user, and not necessary for testing consistency. On the user part, it may be the case that points that are not directly constrained in the problem, are not directly related in the problem domain, and a user might never enquire about the minimal constraint between them. As for testing consistency, minimal constraints between points that are not directly constrained in the problem are irrelevant since they reflect only the minimality of indirect constraints between the points. Hence, it makes sense to look for a relaxed version of the minimal problem, where constraints that are not explicitly given in the original problem are not computed. Such a relaxed version is sufficient for testing consistency, since the missing constraints are, originally, universal. Yet, for finding a solution and feasible values for variables, the regular techniques, using the minimal problem, should be employed. We term the relaxed version "the graph-based minimal problem of P ", and provide a semantical characterization for it.

Definition 2.5: The *graph-based minimal problem of P* , denoted **graph-min**(P), is the projection of **min**(P) on the explicit propositions (original arcs of the problem). A problem P is *graph-minimal* if $P = \mathbf{graph-min}(P)$.

□

The following claim provides a semantic characterization of **graph-min**(P). It shows that within the restricted set of equivalent and comparable problems of P (see Definition 2.3), **graph-min**(P) plays the role of a minimal problem.

Proposition 2: **graph-min**(P) is the tightest problem among all equivalent and comparable problems of P .

Proof: Let $P^`$ be the tightest problem among all equivalent and comparable problems of P . We show that $P^` = \mathbf{graph-min}(P)$. Let $C_{ij}^` \in P^`$ and $C_{ij}^m \in \mathbf{graph-min}(P)$.

⊆ Since $\mathbf{min}(P) \equiv P \equiv P^`$, **min**(P) is tighter than **complement**($P^`$). Hence, $C_{ij}^m \subseteq C_{ij}^`$.

⇒ Since **graph-min**(P) $\equiv P$ and also comparable, $P^`$ is tighter than **graph-min**(P). Hence,

$$C_{ij}^` \subseteq C_{ij}^m.$$

□

Henceforth, we use the term *minimization* as a shorthand for *computing the minimal network (or problem)*, and the term *graph-minimization* as a shorthand for *computing the graph-based minimal network (or problem)*.

3 Sim/2-Tree Decomposition of TCSPs

In this chapter we focus on computing in parts the graph-based minimal problem of a TCSP. In doing that we expect to have a double gain: First, we avoid computing the full minimal network, and second, we compute it by parts. We characterize a decomposition of TCSP into clusters, for which we can prove that graph-minimization can be achieved by at most two repeated graph-minimizations of each cluster. That is, the decomposition, called *sim/2-tree decomposition*, handles the *termination* and the *composition* difficulties in achieving graph-minimization by parts. Moreover, we show that it presents a minimal set of requirements for achieving graph-minimization by parts, i.e., overcoming the termination and composition problems. More concretely, each requirement is justified by presenting a counter example of a decomposition of a non-graph-minimal TCSP with graph-minimal clusters.

Definition 3.1 -- Decomposition of a TCSP, decomposition graph, intersection-width of a decomposition:

A *decomposition of a TCSP* is a partition of its variables into clusters, such that the sub-problems (TCSPs) obtained by the projection of the original TCSP on the cluster variables, cover the whole problem (every proposition is included in some cluster problem).

cluster problems are graph-minimal, i.e., $CP_i = \mathbf{graph-min}(CP_i)$, $1 \leq i \leq n$. The decomposition is *locally-minimal* if all of its cluster problems are minimal.

□

Our first goal in this chapter is to characterize decompositions for which local-graph-minimality guarantees the graph-minimality of the problem. We show that sim/2-tree decompositions satisfy this characterization since they enjoy the property of *local-expansion* of a solution from a cluster problem to its neighbors in the decomposition graph. Consequently, given a sim/2-tree decomposition of a TCSP P , $\mathbf{graph-min}(P)$ can be obtained from independent graph-minimizations of the cluster problems. Then, we suggest an algorithm for achieving graph-minimization of the cluster problems of a sim/2-tree decompositions, with at most two repetitions of graph-minimization per cluster problem. The meaning of these results is that for sim/2-tree decompositions we have a way of computing graph-minimality in parts.

Our second goal in this chapter is to show that the three structure characteristics in the definition of sim/2-tree decompositions are essential in the sense that if any condition does not hold, then it may not be possible to compute $\mathbf{graph-min}(P)$ through local graph-minimizations, or even through full minimizations of the cluster problems. That is, for non sim/2-tree decompositions, graph-minimization or full minimization of the cluster problems may not be sufficient for achieving graph minimization. Furthermore, the process of repeated graph-minimizations in such problems may be (practically) unbound.

Proposition 3: In a locally-graph-minimal sim/2-tree decomposition $\{CP_i\}_{i=1, n}$, of a TCSP P , every solution for a cluster problem CP_i can be expanded to a solution of CP_i and any neighbor cluster problem of CP_i .

Proof: Let S_i be a solution of CP_i . Let CP_j be a cluster problem that shares variables with CP_i .

First assume that P is simple. S_i can be expanded into a solution S_{ij} for $\{CP_i, CP_j\}$, for the following reasons:

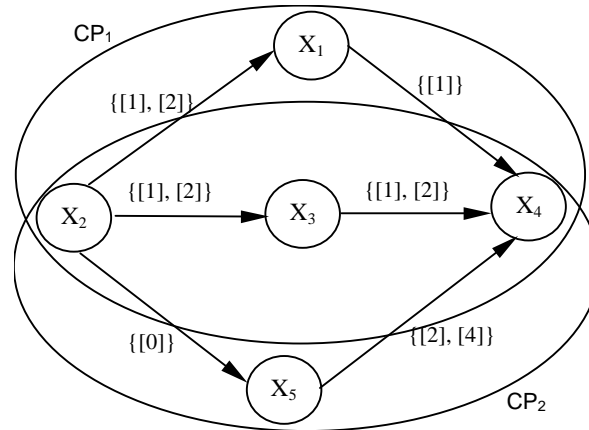
1. The values of the shared variables form a solution of $CP_i \cap CP_j$.
2. Since CP_j is graph-minimal, and $CP_i \cap CP_j$ is fully complemented, the $CP_i \cap CP_j$ is a sub-problem of $\mathbf{min}(CP_j)$. Hence, the solution of $CP_i \cap CP_j$ is a partial solution for CP_j .
3. Since CP_j is simple, $\mathbf{min}(CP_j)$ is backtrack free. Hence, the solution of $CP_i \cap CP_j$ can be expanded into a full solution for $\mathbf{min}(CP_j)$, which is also a solution to CP_j .

If P is a non-simple problem, and the decomposition has intersection-width ≤ 2 , CP_i and CP_j share at most a single binary (non-simple) explicit proposition. Every value in that proposition is feasible in CP_j , due to its minimality. Hence, the solution S_i of CP_i can be expanded into a solution of $\{CP_i, CP_j\}$.

□

Note: Proposition 3 does not hold for decompositions of non-simple problems with intersection-width greater than 2, or for decompositions with non-complete cluster intersections. Examples 6 and 7 are counter examples:

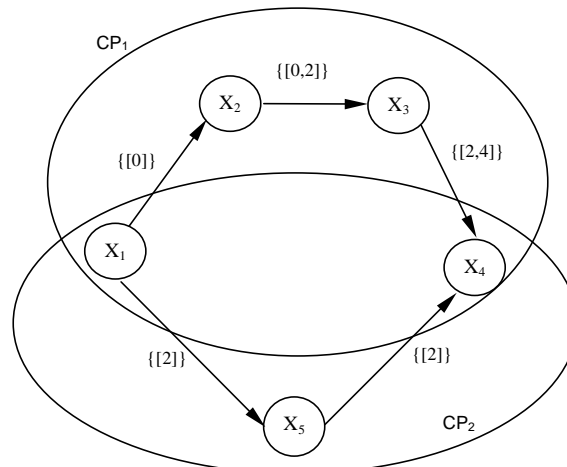
Example 6: A decomposition of a non-simple problem of intersection-width greater than two, with graph-minimal cluster problems, whose solutions cannot be expanded.



It is not hard to see that the cluster problems are graph-minimal, and yet, the partial solution $\{X_1=0, X_2=-2, X_3=0, X_4=1\}$ of CP_1 cannot be extended into a full solution, and the problem is not graph-minimal.

□

Example 7: A decomposition of a simple TCSP P that is not graph-minimal, into 2 graph-minimal cluster problems, with cluster-intersection that is not complemented:



The problem lies in the missing arc, on which $\min(CP_1)$ and $\min(CP_2)$ disagree. Indeed, $CP_1 \cup CP_2$ is not a sub-problem of $\min(P)$, and there are solutions to CP_1 that cannot be extended to solve the whole problem. For example, the solution $\{X_1=0, X_2=0, X_3=2, X_4=6\}$ of CP_1 cannot be expanded to a solution of CP_1 and CP_2 together.

□

This example also clarifies why achieving full minimality of the cluster problems does not provide any extra strength: The composition of the minimal problems of two neighbor cluster problems depends only on the "interface" between the clusters. Hence, it is sufficient to guarantee that cluster intersections are sub-problems of the minimal problems of the clusters. Graph-minimality with complete cluster-intersection is sufficient to guarantee this property. Full minimality does not add anything.

The desired property of TCSPs, i.e., being graph-minimizable by parts, is achieved when having a locally-graph-minimal decomposition for the problem means that it is already graph minimal. The following theorems state that $\text{sim}/2$ -tree decompositions have this property, and that having a locally-graph-minimal $\text{sim}/2$ -tree decomposition is a minimal sufficient condition for graph-minimality. That is, the existence of a decomposition that violates one of the conditions in this characterization cannot guarantee graph-minimality.

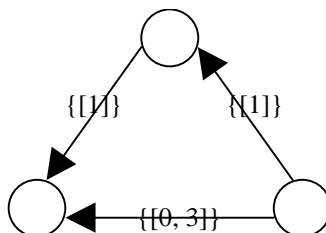
Theorem 1: If a TCSP has a locally-graph-minimal sim/2-tree decomposition then it is graph-minimal.

Proof: Let $\{CP_i\}_{i=1 \dots n}$ be a sim/2-tree decomposition of a TCSP P , such that $CP_i = \mathbf{graph-min}(CP_i)$ ($1 \leq i \leq n$). To show that P is already graph-minimal, it is sufficient to show that every value of a binary proposition is a feasible value of P . Let r be such a value in a cluster problem CP_i . Since CP_i is graph-minimal, r participates in a solution S_i of CP_i . By proposition 3, S_i can be expanded to any neighbor cluster problem of CP_i . Since the decomposition is tree structured, the process can be inductively repeated without cycles, until a solution for P is obtained. \square

Theorem 2: The existence of a locally-graph-minimal sim/2-tree decomposition is a minimal sufficient condition for graph-minimality of the problem.

Proof: Example 6 demonstrates a locally-graph-minimal (actually, local-minimal) tree decomposition of a non-simple problem, that is not graph-minimal. The reason lies in the intersection-width which is 3. The existence of 2 disjunctive propositions that are shared by both cluster problems enables the existence of independent solutions to the cluster problems, that satisfy different simple cases that do not agree on the combination of disjuncts from the intersection propositions. In example 6, the solution $\{X_1=0, X_2=-2, X_3=0, X_4=1\}$ to CP_1 satisfies only the simple case $\{-2\}_{1,2}, [2]_{2,3}, [1]_{3,4}, [1]_{1,4}$, but the partial simple case $\{2\}_{2,3}, [1]_{3,4}$ is not satisfied by any solution of CP_2 . Consequently, the value 2, allowed by the disjunctive propositions $C_{2,3}$ and $C_{3,4}$ is not feasible, and the problem is not graph-minimal.

Violation of the local-graph-minimal or the tree structure requirement in the condition, clearly turns it insufficient for guaranteeing graph-minimality. For example, the following TCSP, which is not graph-minimal, has a trivial locally-graph-minimal non tree decomposition, given by its binary propositions, with intersection-width = 1:



The decomposition is: $\{C_{12}, C_{23}, C_{13}\}$.

Violation of the complete cluster intersection requirements is demonstrated in Example 7. The TCSP in that example is simple and not graph-minimal, and yet the decomposition consists of 2 graph-minimal cluster problems. As explained over there, the missing arc in the intersection prevents true composition of the minimal problems of the clusters. Adding the missing arc, and tightening its proposition until it becomes minimal for both clusters, removes some of the otherwise possible solutions of CP_1 . \square

Note: The sim/2-tree decomposition requires that if the problem is not simple, then the intersection-width of the decomposition is at most two. This pre-requisite for achieving graph-minimization by parts is essential, as long as the problem has a single non-simple proposition. That is, even if the intersection of two cluster problems forms a simple problem, it must be restricted by the intersection-width = 2 condition, in order to maintain the graph-minimization by parts result. Example 6 proves this observation: Taking the problem over the integers, the intersection of the two cluster problems is simple, the two cluster problems are minimal, but the problem is still not graph-minimal.

Proposition 3 suggests an algorithm for computing $\mathbf{graph-min}(P)$ by parts, where the graph-minimization of each cluster problem is repeated at most twice. The idea is to start graph-minimizing the cluster problems of the leaves of the decomposition graph towards the root, and back. On the way up, from the leaves to the root, graph-minimizations of child cluster problems affect the graph-minimization of their parent cluster problems. The upward direction

ends with a graph-minimal problem for the root cluster problem. On the way down, from the root to the leaves, the graph-minimization of a parent cluster problem affects only the graph-minimizations of its child cluster problems, but not its own parent cluster problem. Consequently, the downward direction eventually leads to graph-minimal cluster problems for the whole problem.

Algorithm 2: Transformation of a sim/2-tree decomposition of a TCSP into a locally-graph-minimal sim/2-tree decomposition of the same TCSP.

input: A sim/2-tree decomposition of a TCSP P .

output: A locally-graph-minimal sim/2-tree decomposition of P .

method: Let CP be some cluster problem of P . Consider the decomposition graph rooted at CP .

1. Graph-minimize the cluster problems children before parents.
2. Graph-minimize the cluster problems parents before children.

□

Theorem 3: Algorithm 2 computes $\text{graph-min}(P)$ by parts, and the graph-minimization of each cluster problem is repeated at most twice .

Proof: In the Appendix.

□

3.1 Derivation of a sim/2-tree Decomposition of a TCSP

In this section we present an algorithm for derivation of a sim/2-tree decomposition. The transformation, first, obtains a tree decomposition, and then takes care of the intersection-width. Both steps are obtained by taking the union of cluster problems when necessary. A cycle of cluster problems is turned into a thread by breaking it into two equal length paths, and combining matched cluster problems. The intersection-width, for non-simple problems, is handled by combining cluster problems that share more than two points. At worst, the process ends up with a single cluster problem, which is the original TCSP. In any case, the process does not break the structure of the original decomposition. Finally, the cluster intersections are complemented with the universal propositions.

The "decomposition \Rightarrow sim/2-tree decomposition" transformation is extended, to start with an arbitrary TCSP, by taking each arc as a cluster problem of the initial composition. This initial decomposition is turned into a sim/2-tree decomposition using the process just described. The overall transformation, "TCSP \Rightarrow sim/2-tree decomposition" is summarized in Algorithm 3:

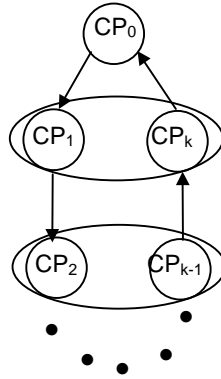
Algorithm 3: Building a sim/2- tree decomposition for a TCSP.

input: A TCSP P .

output: A sim/2-tree decomposition of P .

method: Let $P = \langle X, \Psi \rangle$ where $X = \{X_1, \dots, X_n\}$ and $\Psi = \{C_{ij}\}_{i,j=1,n}$, be a TCSP. The transformation "TCSP \Rightarrow sim/2-tree decomposition" is in three steps, as follows:

1. **TCSP \Rightarrow initial decomposition:** The initial decomposition, denoted CP' , is $\{C_{ij}\}_{i,j=1,n}$.
2. **Initial decomposition \Rightarrow tree decomposition:** CP' is transformed into a tree decomposition CP'' by traveling along the decomposition graph of CP' in a depth first manner, and turning cycles into threads. The exact method follows:
Let CP be some cluster problem in the given decomposition of P . Consider the decomposition graph of P , rooted at CP .
 1. Perform a depth first traverse on the decomposition graph.
 2. If a cycle CP_0, CP_1, \dots, CP_K is observed, with CP_0 being the highest cluster problem in the traverse, replace all pairs CP_i, CP_{k+i+1} , $1 \leq i \leq \lfloor k/2 \rfloor$, by their union.



3. Repeat the traverse.

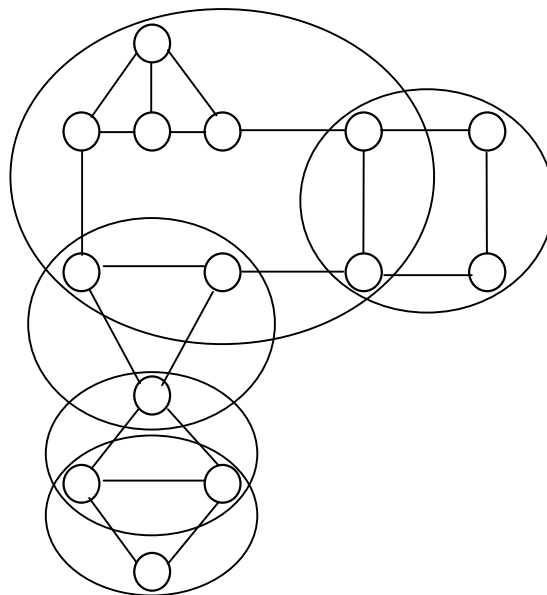
This step is non-deterministic, as it depends on the selection of the root cluster problem, and on the traverse.

3. If P is not a simple problem: **Tree decomposition** \Rightarrow **tree decomposition with intersection-width ≤ 2** : CP'' is transformed into a decomposition CP''' by repeated combinations of all parent-child cluster problem pairs that share more than two points.

4. **Tree decomposition with intersection-width ≤ 2 (if the problem is not simple)** \Rightarrow **sim/2-tree decomposition**: Cluster intersections in the resulting decomposition are complemented with the universal propositions.

The following example demonstrates a sim/2-tree-STCSP that is obtained from its underlying TCSP by the above four step process. The exact constraints are not listed since the transformation depends only on the structure of the given TCSP. The propositions are represented by unlabeled arcs. The components that result from the process are signed by circles.

Example 8: A sim/2-tree decomposition for a TCSP,



□

3.2 Related work.

In [DMP91] it is shown that if the graph of a TCSP can be decomposed into bi-connected components, i.e., into a tree of components in which every two components share at most a

single time point, then the problem can be solved, and its minimal network can be found, by computing the minimal networks of each component independently. The sim/2-tree decomposition generalizes the bi-connected decomposition since it enables neighbor components that share two time points. Our results extend the results concerning the possibility of computing the restriction of the overall minimal network to the component problems without having first to compute the whole network. Yet, the sim/2-tree decomposition does not enable to compute minimal constraints between time points that reside in different components more efficiently. Similarly, in the general case, it is not true that every solution to a component problem in a sim/2-tree decomposition can be extended into a full solution. The reason is that the sharing of two time points between component problems means also stronger dependency between the component problems.

Algorithm 3 presents a method for decomposing a TCSP into a tree of sub-TCSPs for which our previous method, presented in Algorithm 2, can be used to derive an overall graph-minimization from independent graph-minimizations of the sub-problems. This approach reminds the tree clustering technique of [DP88a], for general CSPs, which is based on decomposition of a problem into a tree of cluster sub-problems, finding independent solutions to the cluster problems, and combining these solutions into an overall solution. We cannot use the tree clustering method of [DP88a] since they create join trees by elimination of redundant (equality) arcs from the decomposition graph of the problem. For example, if the decomposition graph includes three clusters: $CP1 = \{A, B, C\}$, $CP2 = \{A, B, D\}$, $CP3 = \{A, B, E\}$, then the structure graph includes the cycle $CP1--AB--CP2--AB--CP3--AB--CP1$, from which any arc can be eliminated, to form a legal join tree for the problem. But our method for computing graph minimization by parts may not be valid, since it is based on the assumption that cluster problems that are not connected have no common variables. Although it may be the case that our results can be extended to include join trees of decomposition graphs, this is not proved yet. In addition, we have shown that just achieving a tree shaped structure cannot guarantee graph-minimization by parts (Example 6). In fact, the characterization of intersection-width = 2 as a condition for graph-minimization by parts, is rather surprising.

4 Structured Temporal Constraint Satisfaction Problems (STCSP⁽ⁿ⁾ $n \geq 0$)

In this chapter we study a generalization of the TCSP model, where binary propositions are replaced by compound propositions. A compound proposition is inductively defined as a set of binary or compound propositions that temporally constrains a set of time points. The generalized model is termed Structured Temporal Constraint Satisfaction Problems (STCSP⁽ⁿ⁾, $0 \leq n$). STCSP⁽⁰⁾ is simply TCSP, i.e., a set of binary propositions, STCSP⁽¹⁾ is a set of STCSP⁽⁰⁾s, i.e., a set of compound propositions of degree 0. In general, STCSP⁽ⁿ⁾ for $1 \leq n$ is a set of STCSP⁽ⁿ⁻¹⁾s, i.e., a set of compound propositions of degree n-1.

The STCSP model leads to an inductive generalization of the sim/2-tree approach for graph-minimization by parts. The computational value lies in the possible save in the need to compute a large amount of practically irrelevant propositions. If a sim/2-tree decomposition of a TCSP has large clusters, then graph-minimization by parts is of little value, since in order to graph-minimize the large clusters we first need to minimize them. However, if a large cluster problem can be graph-minimized by parts using an internal sim/2-tree decomposition, whose clusters can, in turn, be graph-minimized by parts using further internal sim/2-tree decomposition, etc. then we can gain a meaningful save in the number of computed propositions.

The representational advantage of the STCSP model lies in the possibility of capturing the inherent structure of real problems, that can be used as an initial problem decomposition. Our intuition is that natural component problems are relatively densely connected, while most time points in different components are not directly connected. Hence, it is possible that an intersection-width of 2 appears quite naturally in real structured problems. Similarly, natural hierarchies yield, quite frequently, tree structures. Consequently, the STCSP generalization can exploit the natural hierarchy as a good initial decomposition. We start with a definition of the level 1 STCSP model -- STCSP⁽¹⁾.

4.1 The STCSP⁽¹⁾ representation model

Definition 4.1: An STCSP⁽¹⁾ is defined in analogy with a TCSP.

	TCSP	STCSP
Constraint	Binary constraint: A set of intervals	Compound constraint: A set of sets of intervals
Proposition	Binary proposition: A triplet $\langle \{X_i, X_j\}, C, \Lambda \rangle$, where $\{X_i, X_j\}$ - a pair of time points C - a binary constraint Λ - an association of C with the pair (X_i, X_j) .	Compound proposition: A triplet $\langle X, \Delta, \Lambda \rangle$, where X - a set of time points Δ - a compound constraint Λ - a partial mapping $X \times X \rightarrow \Delta$
Problem	A set of binary propositions □	A set of compound propositions

The definition shows that the TCSP model is a special case of the STCSP⁽¹⁾ model, with compound constraints that are singletons, and compound propositions of the form $\langle \{X_i, X_j\}, C, \Lambda \rangle$, where $\Lambda(X_i, X_j) = C$. and C is a set of time intervals.

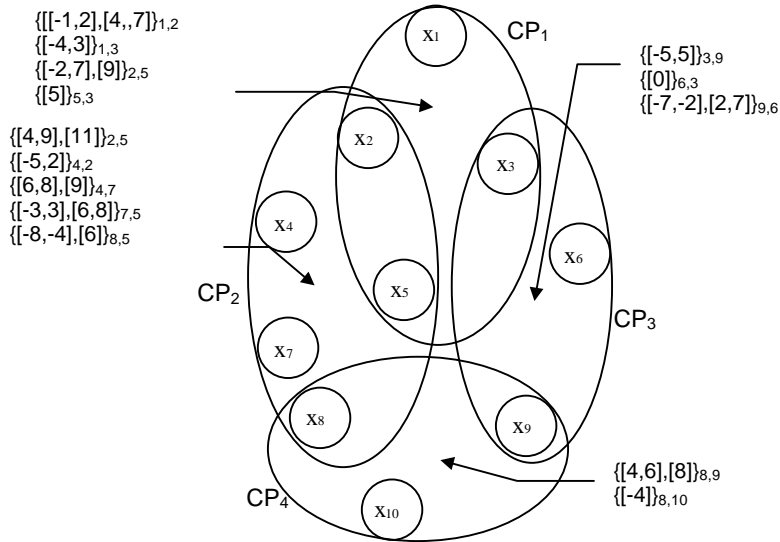
The relevant properties of STCSP⁽¹⁾s, namely, *solution*, *consistency*, and *feasible values for variables* and for *propositions*, are defined analogously to their definitions for TCSPs (see Definition 2.2). That is, a solution is an assignment to the variables that satisfies all propositions, consistency means existence of a solution, and a value is feasible for a variable if it participates in some solution. A feasible value for a proposition $P_i = \langle X_i, \Delta_i, \Lambda_i \rangle$ is a tuple of feasible values for the binary propositions in P_i .

Likewise, the notions of *comparability*, *equivalence*, *tightness*, *complementation*, *minimality*, and *graph-minimality*, are direct generalizations of their TCSP versions. The only difference is that in the STCSP⁽¹⁾ case there is an extra layer. Therefore, each syntactical notion is defined recursively, both for the STCSP⁽¹⁾ and for the STCSP⁽⁰⁾ levels. Two STCSP⁽¹⁾s are *comparable* if they have the same set of variables, their propositions are defined on the same subsets of variables, and corresponding propositions are comparable TCSPs. Two STCSP⁽¹⁾s are *equivalent* if they have the same set of solutions. An STCSP⁽¹⁾ is *tighter than* another comparable STCSP⁽¹⁾ if there is a tightness relationship between their corresponding propositions. The *complemented problem* of an STCSP⁽¹⁾ is a problem obtained by complementing all given propositions, and adding universal compound propositions for all subsets of variables that are not originally constrained.

The *minimal problem* of an STCSP⁽¹⁾ P is the tightest problem among all equivalent comparable problems of **complement**(P). The *graph-minimal* problem of P is the recursive projection of **min**(P) on the explicitly given propositions. That is, **graph-min**(P) consists of the graph-minimal problems of the compound propositions of P. An STCSP⁽¹⁾ P is *minimal* if $P = \mathbf{min}(P)$, and *graph-minimal* if $P = \mathbf{graph-min}(P)$. It is important to note that the compound propositions of **min**(P) are minimal TCSPs, and the compound propositions of **graph-min**(P) are graph-minimal TCSPs. Yet, the propositions of **min**(P) are not necessarily minimal propositions, since they might have simple cases that are created by interval combinations that are not satisfied by any solution of P (a similar case arises in Example 6). Values allowed by such simple cases can be non-feasible.

We can view an STCSP⁽¹⁾ as a *labeled hyper-graph* (V, E) , where the set of time points comprises the set of nodes V, and compound propositions are labeled arcs in E. An arc is the set of time points of a compound proposition, and its label is the compound constraint, together with the Λ mapping. Example 9 shows an STCSP⁽¹⁾ hyper-graph with 4 compound propositions and overall 10 time points. For simplicity, an arc label $\Lambda: X \times X \rightarrow \Delta$ is viewed and denoted as a set of binary propositions, i.e., if for $C \in \Delta$, $\Lambda(X_i, X_j) = C$, then C_{ij} is in the arc label.

Example 9: An STCSP as a hyper-graph.



□

Compound propositions can disagree on the binary constraints set on shared variables. In Example 9, the constraints set on X_2 and X_5 in CP_1 and CP_2 disagree. However, since a solution of an $STCSP^{(1)}$ requires that all constraints are satisfied, binary constraints on shared variables can be intersected without losing any solution. Henceforth we assume that binary propositions on shared variables are themselves shared (otherwise they are intersected). This assumption enables us to draw the arc labels in an $STCSP^{(1)}$ hyper-graph using labeled binary arcs. For the $STCSP^{(1)}$ of Example 9, the result is exactly the TCSP decomposition from Example 5.

4.1.1 The induced TCSP of an $STCSP^{(1)}$

An $STCSP^{(1)}$ can be "downgraded" into an $STCSP^{(0)}$ (i.e., a TCSP) by "ignoring the arc partitions". Similarly, an $STCSP^{(0)}$ can be "upgraded" into an $STCSP^{(1)}$ by imposing a decomposition on its graph. The downgrading process yields, for an $STCSP^{(1)}$, its underlying *induced* TCSP, and an upgrading process is given, for example, by the " $TCSP \Rightarrow$ sim/2-tree decomposition" transformation in Algorithm 3. In this section we show that an $STCSP^{(1)}$ and its induced problem have the same properties. The importance of this observation lies in the fact that a good method to compute the properties of an $STCSP^{(1)}$ (e.g., graph-minimization by parts) provides also a good method to compute the properties of its induced TCSP, and vice versa.

Definition 4.2: The *induced TCSP* of an $STCSP^{(1)}$ $P = \{ \langle X_i, \nabla_i, \nabla_i \rangle \}_{i=1,m}$, denoted **induced**(P), is the problem $\langle \bigcap X_i, \{C_{ij}\} \rangle$, where for every C_{ij} in **induced**(P) there exists a compound proposition $\langle X^k, \nabla^k, \nabla^k \rangle$, such that $X_i, X_j \in X^k$ and for some $C \in \nabla^k$, $\nabla^k(X_i, X_j) = C$. □

induced(P) is well defined, since by the above assumption, for every pair $X_i, X_j \in \bigcap X_i$, there is at most a single C such that for some $\nabla^k, C \in \nabla^k$ and $\nabla^k(X_i, X_j) = C$.

Proposition 4: Let P be an $STCSP^{(1)}$.

1. The *solutions, consistency, and feasible values for variables* for P are exactly those of **induced**(P).
2. A *feasible value for a compound proposition* in P is a tuple of feasible values for binary propositions in **induced**(P) (but not every tuple of feasible values for binary propositions in **induced**(P) is feasible for a compound proposition in P).
3. **graph-min**(P) is the tightest problem among all equivalent and comparable problems of P .

Proof: Parts (1) and (2) are immediate from the definitions. Part (3) is proved as in Proposition 2.

□

The following proposition shows that the relevant properties of an $STCSP^{(1)}$ are equal to those of its induced TCSP. The proposition is used later on in proving that the properties of an $STCSP^{(1)}$ can be generalized to the $STCSP^{(n)}$ level.

Proposition 5: For an $STCSP^{(1)}$ P , $\text{induced}(\text{graph-min}(P)) = \text{graph-min}(\text{induced}(P))$.

Proof: The TCSPs obtained in both sides of the equation are equivalent since an $STCSP^{(1)}$ and its induced TCSP have the same set of solutions, and graph minimization preserves equivalence in both models. To show that the problems are equal we argue that their (binary) propositions are minimal.

Graph minimization for a TCSP yields a TCSP whose (binary) propositions are minimal. Graph minimization for an $STCSP^{(1)}$ yields an $STCSP^{(1)}$ whose compound propositions are graph-minimal TCSPs. Hence the binary propositions within the compound propositions of a graph-minimal $STCSP^{(1)}$ are minimal. Consequently, in the TCSPs obtained in both sides of the equation, all binary propositions are minimal.

□

Conclusion: For an $STCSP^{(1)}$ P , if P is graph-minimal then $\text{induced}(P)$ is also graph-minimal.

Proof: $\text{induced}(P) = \text{induced}(\text{graph-min}(P)) = \text{graph-min}(\text{induced}(P))$.

□

This conclusion is essential for proving the correctness of the graph-minimization by parts for the generalized $STCSP^{(n)}$ model, below.

The $STCSP^{(1)}$ model is an abstraction of TCSP decompositions as a representation model. All decomposition related notions can be adapted to the $STCSP^{(1)}$ context. In analogy to the *decomposition graph* and the *intersection-width of a decomposition* notions, we have in the $STCSP^{(1)}$ context the notions of the *structure graph* and the *intersection-width of the $STCSP^{(1)}$* , respectively. The notions *sim/2-tree $STCSP^{(1)}$* and *locally-graph-minimal* and *locally-minimal $STCSP^{(1)}$* are defined similarly. Consequently, we obtain the same results for $STCSP^{(1)}$ s. In particular, Theorem 1 states that a locally-graph-minimal sim/2-tree $STCSP^{(1)}$ is graph-minimal. Theorem 2 states that this is a minimal sufficient condition for graph-minimality of $STCSP^{(1)}$ s. Algorithm 2 transforms a sim/2-tree $STCSP^{(1)}$ into a locally-graph-minimal sim/2-tree $STCSP^{(1)}$, and Algorithm 3 transforms a TCSP into a sim/2-tree $STCSP^{(1)}$.

In the next section, the $STCSP^{(1)}$ model is inductively generalized into the $STCSP^{(n)}$ model., and the above results are extended to the generalized model as well.

4.2 The $STCSP^{(n)}$ generalization

The $STCSP^{(n)}$ model is an inductive generalization of the $STCSP^{(1)}$ model.

Definition 4.3 -- $STCSP^{(n)}$: Let X be a set of time variables. We denote a TCSP over X by $TCSP_x$.

1. $STCSP_x^{(0)} = TCSP_x = \{ C_{ij} \}_{x_i, x_j \in X}$
2. $STCSP_x^{(n)} = \{ STCSP_Y^{(n-1)} \}_{Y \subseteq X} \quad (n \geq 1)$.

All properties of $STCSP^{(n)}$ s, relations between $STCSP^{(n)}$ s, and operations on $STCSP^{(n)}$ s, are defined exactly as their $STCSP^{(1)}$ version. This includes the properties of *solution*, *consistency*, *feasible values for variables and for propositions*; the relations of *comparability*, *equivalence*, and *tightness*; and the characterizations of *minimality*, *graph-minimality*, and *induced problem*. The induced problem have the same properties, as characterized in Propositions 4 and 5.

In order to obtain graph-minimization by parts at the n-th level, we first generalize also the notions of *structure graph*, *intersection-width*, *sim/2-tree*, *locally-graph-minimal*, and *locally-minimal*. The structure graph is defined as for $STCSP^{(1)}$ s, based on the variable intersection of compound propositions. The intersection-width counts the number of shared

variables between propositions. In the definition of a sim/2-tree $STCSP^{(n)}$, the proposition intersections must be complemented with binary propositions. The graph-minimization by parts of an $STCSP^{(n)}$, is proved by downgrading an $STCSP^{(n)}$ into an $STCSP^{(n-1)}$ with the same structure. The downgrading cannot use the induced problem since the properties of the structure graph are not preserved. In particular, a tree $STCSP^{(n)}$ might have a non-tree induced $STCSP^{(n-1)}$.

We term the compound propositions of an $STCSP^{(n)}$ *level 1 components*, and inductively, the compound propositions of level i components ($1 \leq i \leq n$) are *level $i+1$ components*. The level $n-1$ components are $STCSP^{(1)}$ s, the level n components are TCSPs ($STCSP^{(0)}$ s), and the $n+1$ level components are binary propositions. The $STCSP^{(n)}$ itself is its own level 0 component. An $STCSP^{(n)}$ is downgraded into an $STCSP^{(n-1)}$ with the same structure, by replacing its level $n-1$ components by their induced TCSPs. The key property is that this downgrading procedure preserves both the sim/2-tree structure, and the locally-graph-minimal property. This property results from the following proposition:

Proposition 6: Structure preserving downgrading and upgrading of an $STCSP^{(n)}$.

1. In a graph-minimal $STCSP^{(n)}$, replacing all level $n-1$ components ($STCSP^{(1)}$) by their induced TCSPs, yields a graph-minimal $STCSP^{(n-1)}$, with the same solutions and feasible values for variables and for the underlying binary propositions.
2. In a graph-minimal $STCSP^{(n)}$, replacing each level n component (TCSP = $STCSP^{(0)}$) by an $STCSP^{(1)}$ of which it is an induced problem (i.e., imposing a decomposition) yields a graph-minimal $STCSP^{(n+1)}$, with the same solutions and feasible values for variables and for the underlying binary propositions.

Proof: Both parts are proved by induction on the $STCSP$ level n . The basis for $n = 1$, for both parts, is obtained from the conclusion that follows from Propositions 4 and 5.

For $n > 1$, the downgrading procedure yields, by the inductive hypothesis, compound propositions that are graph-minimal $STCSP^{(n-2)}$, with the same solutions and feasible values for variables and for the underlying binary propositions, as in their corresponding original propositions. Since the binary propositions were originally minimal with respect to the whole problem, and the set of solutions for the $STCSP^{(n-2)}$ compound proposition stay intact, the resulting $STCSP^{(n-1)}$ is graph-minimal.

For $n > 1$, the argumentation for the upgrading procedure is similar.

□

Based on Proposition 6, we can now prove the generalization of Theorem 1 and Algorithms 2 and 3 to $STCSP^{(n)}$ s.

Theorem 4: A locally-graph-minimal sim/2-tree $STCSP^{(n)}$ is graph-minimal.

Proof: Induction on the $STCSP$ level n . The basis for $n = 1$ is proved in Theorem 1. For $n > 1$, applying the downgrading procedure described in Proposition 6 yields, by the inductive hypothesis, a graph-minimal $STCSP^{(n-1)}$. Upgrading this $STCSP^{(n-1)}$, according to Proposition 6, leaves it graph-minimal, and yields the original problem.

□

Algorithm 4: Transformation of a sim/2-tree $STCSP^{(n)}$ into a an equivalent locally-graph-minimal sim/2-tree $STCSP^{(n)}$.

input: A sim/2-tree $STCSP^{(n)}$ P .

output: A locally-graph-minimal sim/2-tree $STCSP^{(n)}$ equivalent to P .

method:

1. $n = 0$: Compute $\min(P)$ and project it on the graph of P .
2. $n > 0$: Let CP be some compound proposition of P . Consider the structure graph rooted at CP .
 - a. **Upward graph-minimization:** Apply Algorithm 4 to all compound propositions of P , children before parents.

- b. **Downward graph-minimization:** Apply Algorithm 4 to all resulting compound propositions of P, parents before children.

□

Theorem 5: Algorithm 4 computes **graph-min**(P) by parts, and the graph-minimization of each compound proposition is repeated at most twice .

Proof: Induction on the STCSP level n. The basis for $n = 1$ is proved in Theorem 3. For $n > 1$, applying the downgrading procedure described in Proposition 6 yields, by the inductive hypothesis, a graph-minimal $STCSP^{(n-1)}$. Upgrading this $STCSP^{(n-1)}$, according to Proposition 6, leaves it graph-minimal, and yields the output problem. The graph-minimization of each compound proposition is repeated at most twice since there are performed upward and downward, just once.

□

Algorithm 5: Transformation of a TCSP into an equivalent sim/2-tree $STCSP^{(n)}$.

input: A TCSP P.

output: A sim/2-tree $STCSP^{(n)}$ equivalent to P.

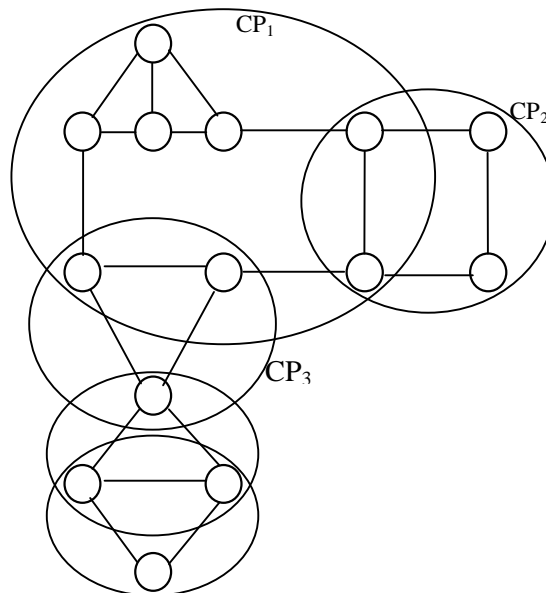
method:

1. Apply Algorithm 3 to P. Resulting an $STCSP^{(1)}$ P'.
2. Apply Algorithm 3 to each compound proposition of P'. If for a compound proposition Q the result is an $STCSP^{(1)}$ Q', with at least two compound propositions, replace Q by Q', and apply step (2) to Q'.

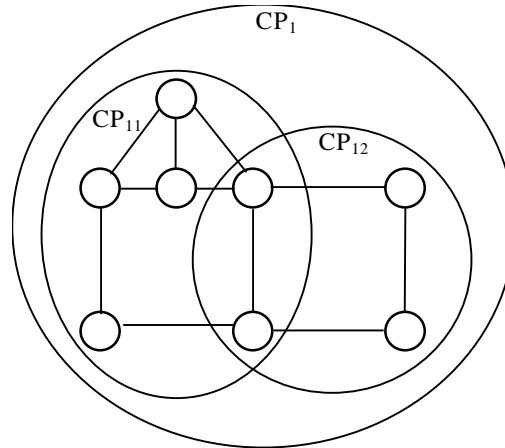
□

Example 10: TCSP \Rightarrow sim/2-tree $STCSP^{(4)}$.

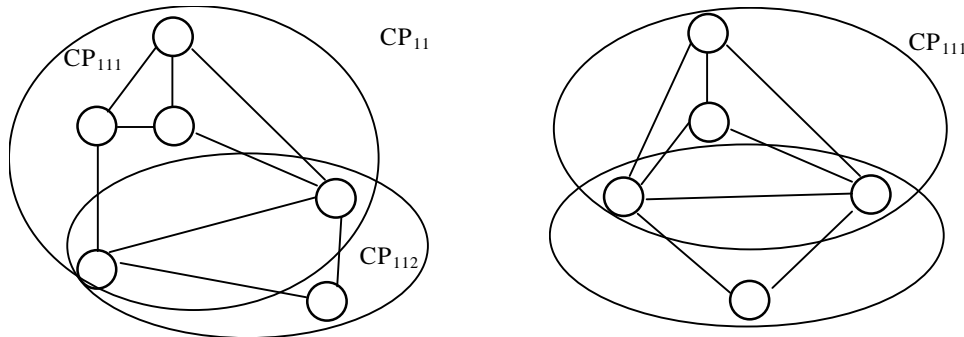
Consider the $STCSP^{(1)}$ from Example 8:



The compound propositions CP_1 and CP_2 can be further decomposed. CP_1 can be decomposed into CP_{11} and CP_{12} as follows:



CP_{11} can be further decomposed into CP_{111} and CP_{112} ; CP_{111} can still be decomposed into 2 components:



Similarly, CP_{12} and CP_2 can be decomposed once, each. All in all, the problem can be decomposed into an $STCSP^{(4)}$. Note that the induced problem of the resulting $STCSP^{(4)}$ is not tree structured since there is a cycle between CP_3 , CP_{11} , and CP_{12} .
□

5 Conclusion and Future Research

In this paper we studied the possible benefits that TCSPs might gain from an additional structuring. We characterized the sim/2-tree decomposition that enables to compute the minimal constraints of input propositions by separate computations of the minimal networks of their components. For a given sim/2-tree decomposition of a TCSP, the restriction of its minimal network to the component problems can be computed by computing the minimal networks of the components at most twice. The sim/2-tree decomposition extends the results of [DMP91] about decomposition of a TCSP into bi-connected components, into a tree of sub-problems that share at most pairs of time points. We also proved that the conditions form a minimal set of conditions. That is, if any condition in the definition of sim/2-tree decompositions is violated, it is not guaranteed that the graph minimal network can be computed by separate computations of the minimal networks of the components. An algorithm for identifying a sim/2-tree decomposition of a TCSP is provided as well. We also generalized the sim/2-tree decompositions in an inductive manner, that enables components of a decomposition to be further decomposed. For that purpose we introduced the model of $STCSP^{(n)}$ ($0 \leq n$).

If the separate computations of the minimal networks of components in a sim/2-tree decomposition does not lead to fragmentation of intervals, the complexity of computing minimal constraints of input propositions is reduced to the complexity of computing minimal

networks of the components. In any case, the complexity does not increase, and the benefit of partitions by articulation points are preserved. The sim/2-tree decomposition can also be useful in maintaining updates of a TCSP, since propagation of modifications is blocked when it reaches a component that is not changed.

We are currently studying a refined definition of the decomposition graph of a TCSP decomposition. It seems that the current definition is rather restrictive; and a somewhat weaker structuring relation is sufficient. In particular, it seems that our results can be extended to join trees of decomposition graphs as well. In that case, the tree clustering method of [DP88a] can be used for obtaining a sim/2-tree decomposition.

We believe that structure is inherent in real problems. The STCSP⁽ⁿ⁾ model strengthens the TCSP model by capturing such structuring. The bus-train example, presented in the introduction, demonstrates a real problem that its natural structuring actually satisfies the conditions for graph-minimization by parts. From the point of view of conceptual modeling, the further abstraction added in the STCSP⁽ⁿ⁾ model is helpful in all aspects of information processing, such as maintenance, organization, and inference.

Acknowledgment

We are much indebted to Rina Dechter for helpful discussions, and to Peter van Beek for providing detailed comments on an earlier draft of this paper, and pointing to possible extensions of this work. We are also much grateful to an anonymous referee for detailed comments, that helped us to clarify the ideas and the presentation of this paper.

References

- [AHU75] Aho, Hopcroft and Ullman. The Design of Computer Algorithms, Addison Wesley, (1975), 200-201.
- [Allen83] J.F. Allen, 'Maintaining knowledge about temporal intervals', Commun. ACM 26 (11), 832-843, (1983).
- [Allen84] J.F. Allen, 'Towards a general theory of action and time', Commun. Artif. Intell. 23(2), 123-154, (1984).
- [AH85] J.F. Allen and P.J. Hayes, 'A commonsense theory of time', Proc. IJCAI 1985, 528-531, 1985.
- [Ariav86] G. Ariav, 'A temporally oriented data model', ACM trans. On database systems, 11(4), 1986.
- [BTK91] F. Bacchus, J. Tenenber and J. A. Koomen, 'A non-refied temporal logic', Artif. Intell. 52, 87-108, (1991).
- [BK96] M. Balaban and Y. Kornatzky. 'A Data model for Processes Based on Relative Time' Journal of Intelligent Information Systems, 7(1), pp. 29-50, 1996.
- [BCS92] M. Boddy, J. Carciofini and B. Schrag, 'Managing disjunction for practical temporal reasoning', Proc. KR-92, 1992.
- [CW83] J. Clifford and D.S. Warren, 'Formal semantics for time in databases', ACM trans. On database systems, 8(2), 1983.
- [DM87] T.M. Dean and D.V. McDermott, 'Temporal data base management', Artif. Intell. 32, 1-55, (1987).
- [Dean89] T. Dean, 'Using temporal hierarchies to efficiently maintain large temporal data base', J. ACM 36 (4), 687-718, (1989).
- [DMP91] R. Dechter and I. Meiri and J. Pearl, 'Temporal constraint network', Artif. Intel. 49, 61-95, (1991).
- [DP88] R. Dechter and J. Pearl, 'Network based heuristics for constraint satisfaction problems', Artif. Intell. 34,(1), 1-38, (1988).
- [DP88a] R. Dechter and J. Pearl, 'Tree clustering for constraint networks', Artif. Intell. 38, 353-366, (1988).
- [Freuder85] E.C. Freuder, 'A sufficient condition of backtrack-bounded search', J. ACM 32, 755-761, (1985).
- [GJC94] M. Gyssens, P.G. Jeavons and D.A Choen, 'Decomposition constraint satisfaction problems using data base techniques', Artif. Intell. 66, 57-89, (1994).

- [GS93] M.C. Golumbic and R. Shamir, 'Complexity and Algorithms for Reasoning about Time: A Graph-Theoretic Approach', JACM 40, 1108-1133, 1993.
- [JMR90] C.S. Jensen, L. Mark and N. Roussopoulos, 'Incremental implementation model for relational databases with transaction time', IEEE trans. On knowledge and data engineering, 1990.
- [JB96] P. Jonsson and C. Backstrom, 'A Linear-Programming Approach to Temporal Reasoning', proc. AAI-96, 1235-1240, 1996.
- [KL91] H. Kautz and P.B. Ladkin, 'Integrating metric and qualitative temporal reasoning', proc. AAI-91, 241-246, 1991.
- [LR92] P.B. Ladkin and A. Reinefeld, 'Effective solution of qualitative interval constraint problem', Artif. Intell. 57, 105-124, (1992).
- [McDermott82] D. McDermott, 'A temporal logic for reasoning about processes and plans', Cognitive Science 6, 101-155, 1982.
- [Meiri96] I. Meiri, 'Combining qualitative and quantitative constraints in temporal reasoning', Artif. Intell. 87 (1-2), 343-385, 1996.
- [MTM91] J. Moyne, T. Teorey and L. McAfee, 'Time sequence ordering extensions to the entity-relationship model and their application to automated manufacturing process', Data & knowledge engineering 6, 421-443, 1991.
- [PB91] M. Poesio and R.J. Brachman, 'Metric Constraints for Maintaining Appointments: Dates and Repeated Activities', AAI-91, 253-259, 1991.
- [Rosen95] T. Rosen. 'Structured Temporal Constraints Satisfaction Problems', M.Sc. Dissertation, Ben-Gurion University of the Negev, Beer Sheva, Israel 1995.
- [SD97] E. Schwalb and R. Dechter, 'Processing Disjunctions in Temporal Constraint Networks', Artif. Intell. 93 (1-2), 29-41, 1997.
- [Shoham86] Y. Shoham, 'Reasoning about change: time and causation from the stand point of artificial intelligent', Ph.D. dissertation, Yale Univ., 1986.
- [VanBeek92] P. Van Beek, 'Reasoning about qualitative temporal information', Artif. Intell. 58, 297-326, 1992.
- [WD92] G.T.J. Wu and U. Dayal, 'A uniform model for temporal object-oriented databases', proc. IEEE data engineering, 584-593, 1992.

APPENDIX: Proofs

Proposition 1: In a simple problem, all propositions are minimal iff every proposition C_{ij} is tighter than or equal to all indirect propositions between X_i and X_j .

Proof:

\Rightarrow If C_{ij} is minimal, then by its definition it is tighter than or equal to all indirect propositions between X_i and X_j .

\Leftarrow We prove, by induction on the size (number of variables) of the problem, that the problem is progressively solvable. As a result, all of its propositions are minimal.

For a problem of size 0 the claim is vacuously true. Assume that the claim holds for problems with n variables, and prove that this implies the claim for problems with $n+1$ variables. Let P be a size $n+1$ problem ($n \geq 0$), and let $X_1, X_2, \dots, X_n, X_{n+1}$ be any ordering of its variables. Let P' be the projection of P on the variables X_1, X_2, \dots, X_n . By the assumption every proposition C_{ij} of P is at least tighter than or equal to all non-indirect propositions between X_i and X_j in P . The assumption holds also for P' , being a projection of P . Since P' is of size n , P' is progressively solvable, by the inductive hypothesis. It remains to show that every solution of P' can be extended into a solution for P . Since P' is defined by an arbitrary ordering of the variables, this means that P is progressively solvable.

Let $S' = (s_1, s_2, \dots, s_n)$ be a solution for P' . Denote $C_{i,n+1} = \{ [v_i, u_i] \}_{i,n+1}$, $i=1, \dots, n$, the proposition between X_i and X_{n+1} . We need to select a value s_{n+1} for X_{n+1} , that extends the solution S' into a solution S for P . That is, for every $i=1, \dots, n$, s_{n+1} must satisfy the constraints

$s_i+v_i \leq s_{n+1} \leq s_i+u_i$. Let l be the index for i such that s_i+v_i is maximal among all s_i+v_i , and k be the index for i such that s_k+u_k is minimal among all s_i+u_i . We select s_{n+1} such that $s_l+v_l \leq s_{n+1} \leq s_k+u_k$. Since S is a solution for P , $s_k-s_l \in C_{l,k}$. By the assumption, $C_{l,k} \subseteq C_{l,n+1} \oplus C_{n+1,k}$. Hence, $s_k-s_l \in \{ [v_i, u_i] \}_{l,n+1} \oplus \{ [-u_i, -v_i] \}_{n+1,k}$. Hence there are d_l, d_k , such that $v_l \leq d_l \leq u_k$ and $v_k \leq d_k \leq u_k$, and $s_k-s_l = d_l-d_k$. That is, $s_k+d_k = s_l+d_l$. Choosing $s_{n+1} = s_k+d_k = s_l+d_l$, we get $s_l+v_l \leq s_{n+1} \leq s_k+u_k$. \square

Theorem 3: Algorithm 2 computes **graph-min**(P) by parts, and the graph-minimization of each cluster problem is repeated at most twice.

Proof: Let $\{P_i\}_{i=1..n}$ be the given sim/2-tree decomposition of P . Assume that CP is the selected root cluster, and k is the depth of the decomposition tree rooted at CP . The algorithm includes the following steps:

Step 0: $\{P_1^0, \dots, P_n^0\} = \{P_1, \dots, P_n\}$

Step 1: Graph-minimize all depth k P_i^0 . Intersect the shared binary propositions.

Step 2: Graph-minimize all depth $k-1$ P_i^1 . Intersect the shared binary propositions.

Step $k+1$: Graph-minimize CP (depth 0). Intersect the shared binary propositions.

Step $k+2$: Graph-minimize all depth 1 P_i^{k+1} . Intersect the shared binary propositions

...

Step $2k+1$: Graph-minimize all depth k P_i^{2k} . Intersect the shared binary propositions.

Note that the steps preserve equivalence since no feasible value of a proposition is lost, as the omitted values are not even locally consistent. We have:

1. Equivalence is preserved.
2. Termination is obtained in $2k+1$ steps, with at most two repetitions of graph-minimization for each cluster problem..

Hence, we have to show that the process results **graph-min**(P_i) $1 \leq i \leq n$. We prove it by induction on the depth of the decomposition tree.

Basis : $k = 0$ (A single cluster)

I.S. : $k > 0$.

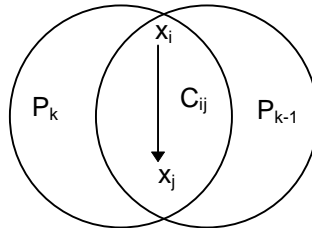
Step 1 graph-minimizes all depth k leaf cluster problems, and performs intersection only between depth k to depth $k-1$ cluster problems. Now, consider the process, starting from **Step 2** until **Step $2k$** . This process operates on a depth $k-1$ tree, and by the inductive hypothesis, achieves **graph-min**(P_i) for all of its cluster problems. It is left to show that after **Step $2k+1$** the following holds:

1. The depth i nodes, for $i \leq k-1$, are left intact.
2. The depth k nodes are graph-minimized.

Both parts are achieved, once we show that the intersection that follows the graph-minimization (of the depth k nodes) does not cause any change. If no intersection is changed, then the depth k nodes are left graph-minimal, and all other nodes are left intact (and graph-minimal, as well).

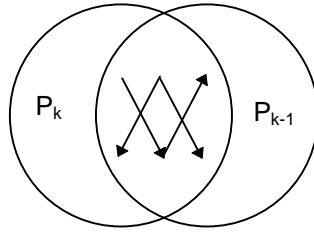
Consider two intersecting nodes P_k, P_{k-1} in depth $k, k-1$, respectively. We have one of two cases:

1. The problem is of at most width 2. We have the following situation before **Step $2k+1$** :



C_{ij} includes only feasible values for both P_k , and P_{k-1} , since it is at most tighter than the constraint between X_i and X_j following **Step 1**. All other propositions in P_k have not changed since **Step 1**, when P_k was minimized. Consequently, the minimization of P_k at **Step $2k+1$** does not affect the shared binary proposition.

2. The problem is simple. We have the following situation before **Step 2k+1**:



Every solution of $P_k \cap P_{k-1}$ is a partial solution of P_k and of P_{k-1} , since the binary propositions in $P_k \cap P_{k-1}$ are at most tighter than their corresponding propositions following **Step 1**. All other propositions in P_k have not changed since **Step 1**, when P_k was graph-minimized. Consequently, the graph-minimization of P_k at **Step 2k+1** does not affect the shared binary propositions.

□