

Constraint-Based Temporal Reasoning

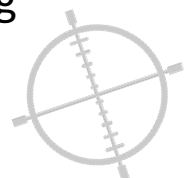
Roman Barták (Charles University in Prague)

Robert A. Morris (NASA Ames Research Center)

K. Brent Venable (Tulane University and
The Florida Institute of Human and Machine Cognition)

Scope of Tutorial

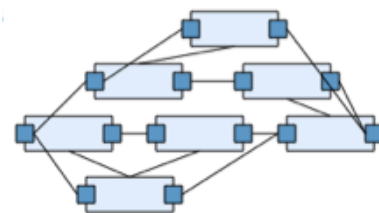
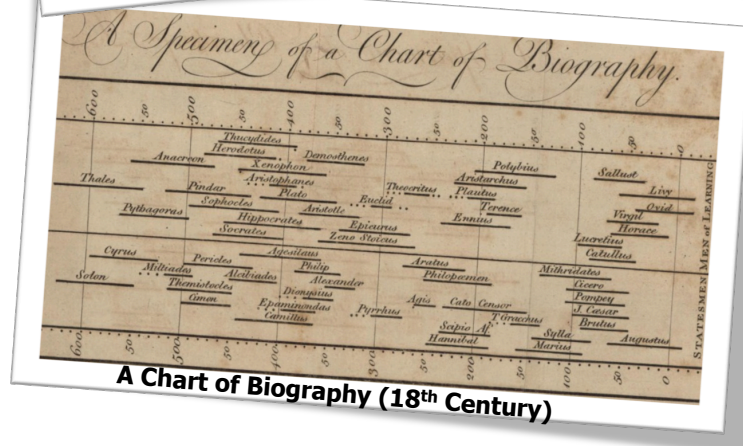
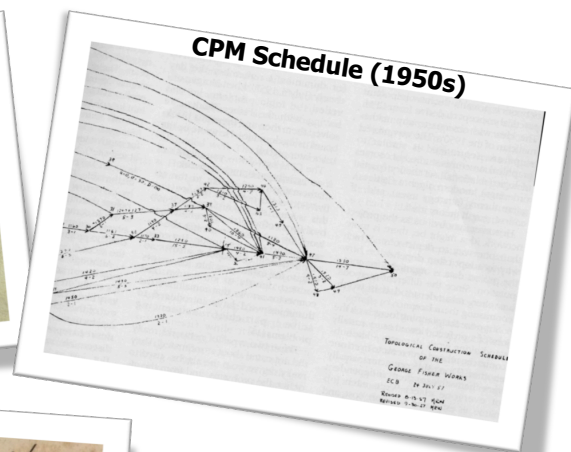
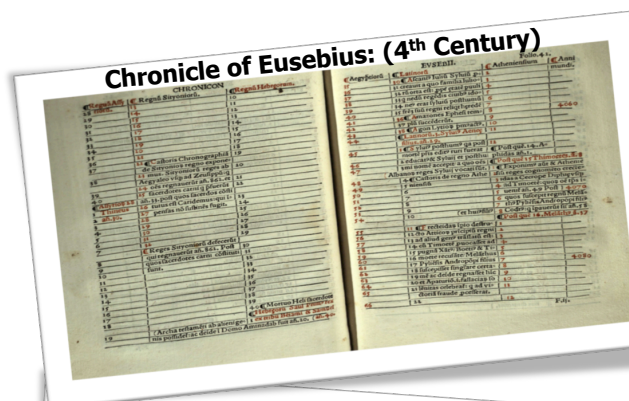
- Introduction to the varieties of graphical representations of time
 - Node elements have variables that take on time values
 - Edges represent temporal ordering or duration constraints
- Processing temporal data using constraint reasoning methods
 - Inference-based or search-based processing



- Introduction and Background (20 minutes, Morris)
- Constraint-based temporal reasoning systems (70 minutes, Barták)
- Extensions to temporal reasoning systems (60 minutes, Venable)
- Applications (30 minutes, Morris)



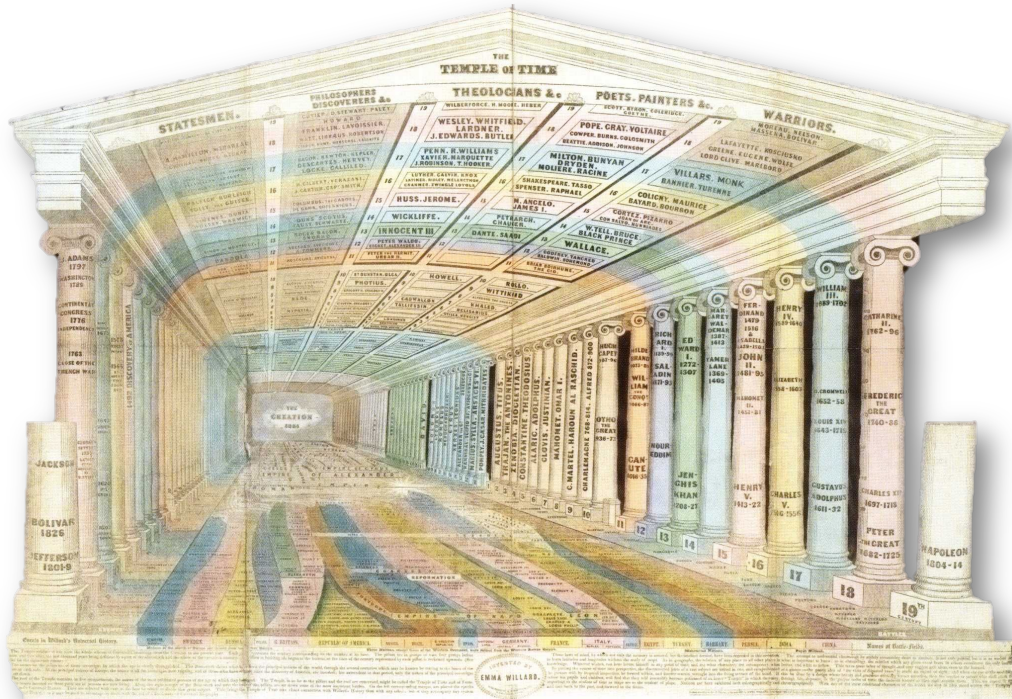
Representing Time: Chronicles, Lines and Graphs



Constraint Plan Graph (late 20th Century)

Other ways of combining chronicles and lines

The Temple of Time 19th Century



20th Century Industrial Planning and Scheduling

- Complex Production and Transportation Scheduling Problems (1910s)
 - Separation of planning (sequencing) from scheduling (assigning resources)
 - Refinements to timelines and chronicles
 - Gantt Chart
 - Timetables
- Early Mathematical Formulations (1950s):
 - Linear Programming, Critical Path Method, PERT
 - Early graphical representations were used to explain LP constraints to management (activity-on-arrow diagram)
 - DuPont: “find something to do” with UNIVAC1

- MILP and Dynamic Programming Methods for Scheduling
 - Development of hierarchy of scheduling problems
 - Numerical Methods for trajectory optimization
- Logical models
 - Time and change
 - From timeline to real line
 - Modal logics of time
 - First order representations
 - Method of Temporal arguments

- Network flows over time (Dynamic Flows)
 - *how many cars can the streets of a town serve?*
 - Solutions using shortest path algorithms
- Temporal Databases
 - Modern version of Chronicles
 - Extensions of relational theory
- Model Checking Using Temporal Logics
 - Importance of proof methods in verification

- Reasoning about Change
 - “After the stock goes above \$X, sell.”
 - State changes trigger actions.
 - “Make mortgage payment at the beginning of each month”
 - Time triggers actions.
- Time has a mathematical structure independent of the causal structure of change.
 - Discrete or continuous?
 - Linear or branching or parallel or cyclical?
 - Bounded or infinite?
 - Point or Interval fundamental?
 - Granularity
- Event Calculus (Kowalski and Sergot, 1986)
- Time map manager (McDermott 1982)
- Reasoning with multiple granularities

- Representation of constraints (graphically) as relations on variables
- Constraint processing algorithms:
 - Search-based: Variations on Backtracking
 - Inference-based: Constraint propagation methods (consistency-enforcing methods)
 - Both support a graph-based view.
- Pioneering works:
 - Montanari (1974): Networks of Constraints: Fundamental Properties and Applications to Picture Processing
 - Mackworth (1977): Consistency in Networks of Relations
 - Freuder (1978): Synthesizing Constraint Expressions

- James Allen, Maintaining Knowledge about Temporal Intervals (CACM 1983)
- R. Dechter, I. Mehri, J. Pearl, Temporal Constraint Networks (AIJ, 1991)
- *Virtually everything we talk about for the rest of this tutorial starts from one or the other of these papers.*

Temporal Reasoning

Frameworks and Algorithms

What is time?

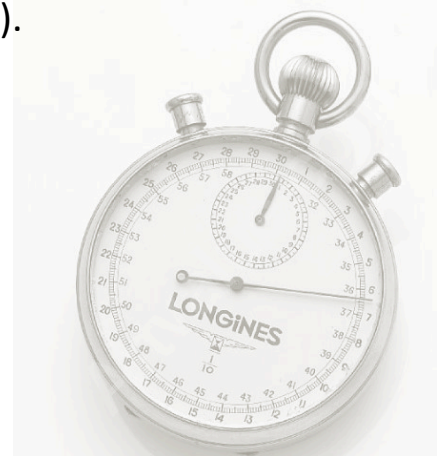
The core mathematical structure for describing time is a **set with transitive and asymmetric ordering** relation.

The set can be continuous (real numbers) or discrete (integer numbers).

The P&S system can use a **database of temporal references** with a procedure for **verifying consistency** and an **inference mechanism** (to deduce new information).

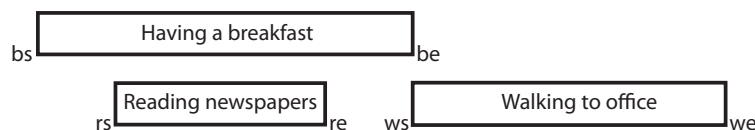
We can model time in two ways:

- **qualitative**
relative relations (A finished before B)
- **quantitative**
metric (numerical) relations (A started 23 minutes after B)

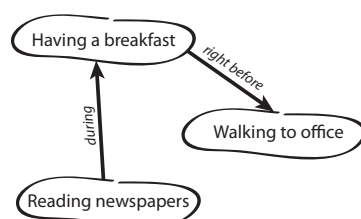


Qualitative approach

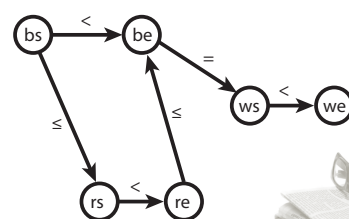
- Based on **relative temporal relations** between temporal references.
- *"I read newspapers during breakfast and after breakfast I walked to my office"*



Temporal intervals (activities)



Time points (important events)



When **modeling time** we are interested in:

- **temporal references**
(when something happened or hold)
 - **time points** (instants) when a state is changed
instant is a variable over the real numbers
 - **time periods** (intervals) when some proposition is true
interval is a pair of variables (x,y) over the real numbers, such that $x < y$
- **temporal relations** between temporal references
 - **ordering** of temporal references

Typical problems solved:

- verifying **consistency** of the temporal database
- asking **queries** (“*Did I read newspapers when entering the office?*”)
- finding **minimal networks** to deduce inevitable relations

Symbolic calculus modelling qualitative relations between instants.

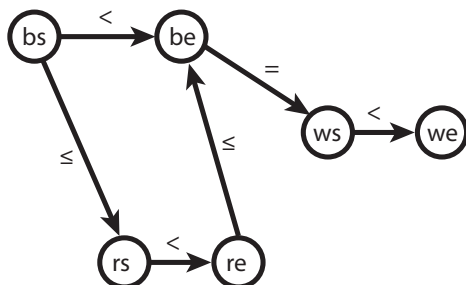
- There are three possible **primitive relations** between instants t_1 and t_2 :
 - $[t_1 < t_2]$
 - $[t_1 > t_2]$
 - $[t_1 = t_2]$
 Relations $P = \{<, =, >\}$ are called **primitive relations**.
- Partially known relation between two instants can be modelled using a set (disjunction) of primitive relations:
 - $\{\}, \{<\}, \{=\}, \{>\}, \{<,=\}, \{>,=\}, \{<,>\}, \{<,=,>\}$
- **Relation** r between temporal instants t and t' is denoted **$[t \ r \ t']$**
- Point algebra allows us to **work with relative relations** without placing the instants to particular (numeric) times.

- Let R be a set of all possible relations between two instants
 - $\{\{\}, \{<\}, \{=\}, \{>\}, \{<,=\}, \{>,=\}, \{<, >\}, \{<, =, >\}\}$
- Symbolic operations over R :
 - set operations** \cap, \cup
 - they express conjunction and disjunction of relations
 - composition operation** \bullet
 - transitive relation for a pair of connected relations
 - $[t_1 r t_2]$ and $[t_2 q t_3]$ gives $[t_1 r \bullet q t_3]$ using the table

\bullet	$<$	$=$	$>$
$<$	$<$	$<$	P
$=$	$<$	$=$	$>$
$>$	P	$>$	$>$

- The most widely used operations are \cap and \bullet , that allow combining existing and inferred relations:
 - $[t_1 r t_2]$ and $[t_1 q t_3]$ and $[t_3 s t_2]$ gives $[t_1 r \cap (q \bullet s) t_2]$

“I read newspapers during breakfast and after breakfast I walked to my office”



- Query: *“Did I read newspapers when entering the office?”*
- $[rs < we] \wedge [we < re]$

$$\begin{aligned}
 & (r_{re,be} \bullet r_{be,ws} \bullet r_{ws,we}) \cap (r_{re,we}) \\
 &= (\{<, <\} \bullet \{=\} \bullet \{<\}) \cap \{>\} \\
 &= \{<\} \cap \{>\} = \{\}
 \end{aligned}$$

\bullet	$<$	$=$	$>$
$<$	$<$	$<$	P
$=$	$<$	$=$	$>$
$>$	P	$>$	$>$

- A set of instants X together with the set of (binary) temporal relations $r_{i,j} \in R$ over these instants C forms a **PA network** (X, C) .
 - If some relation is not explicitly assumed in C then we assume universal relation P .
- The **PA network** consisting of instants and relations between them is **consistent** if it is possible to assign a real number to each instant in such a way that all the relations between instants are satisfied.

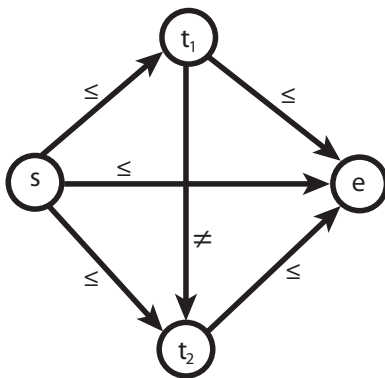
Claim:

The PA network (X, C) is consistent if and only if there exists a set of primitive relations $p_{i,j} \in r_{i,j}$ such that for any triple of such relations $p_{i,j} \in p_{i,k} \bullet p_{k,j}$ holds.

Efficient consistency checking:

To make the PA network consistent it is enough to make its transitive closure, for example using techniques of **path consistency**.

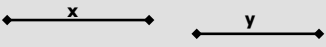
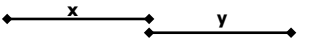
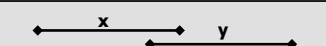
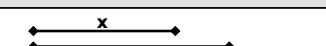
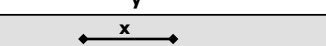
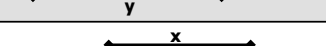
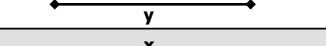
- for each k : for each i, j : do $r_{i,j} \leftarrow r_{i,j} \cap (r_{i,k} \bullet r_{k,j})$
- obtaining $\{\}$ means that the network is inconsistent



- PC verifies consistency but does not remove redundant constraints.
- Primitive constraint $p_{i,j}$ is **redundant** if there does not exist any solution where $[t_i \ p_{i,j} \ t_j]$ holds.
- **PA network is minimal** if it has no primitive constraints that are redundant.
- To make the network minimal we need 4-consistency.

Symbolic calculus modelling relations between intervals (interval is defined by a pair of instants i^- and i^+ , $[i^-, i^+]$)

- There are thirteen primitive relations:

x b efore y	$x^+ < y^-$	
x m eets y	$x^+ = y^-$	
x o verlaps y	$x^- < y^- < x^+ \wedge x^+ < y^+$	
x s tarts y	$x^- = y^- \wedge x^+ < y^+$	
x d uring y	$y^- < x^- \wedge x^+ < y^+$	
x f inishes y	$y^- < x^- \wedge x^+ = y^+$	
x e quals y	$x^- = y^- \wedge x^+ = y^+$	
bi,mi,oi,si,di,fi	symmetrical relations	

Interval algebra – consistency

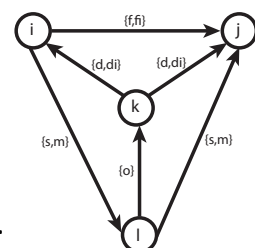
- Primitive relations can be again combined in sets (2^{13} relations).
 - Sometimes we select only a subset of possible relations that are useful for a particular application.
 - for example $\{b,m,bi,mi\}$ means no-overlaps and it is useful to model unary resources
- set operations \cap , \cup and the composition operation •
- The **IA network** is **consistent** when it is possible to assign real numbers to x_i^-, x_i^+ of each interval x_i in such a way that all the relations between intervals are satisfied.

Claim:

The IA network (X, C) is consistent if and only if there exists a set of primitive relations $p_{i,j} \in r_{i,j}$ such that for any triple of such relations $p_{i,j} \in p_{i,k} \bullet p_{k,j}$ holds.

Notes:

- Path consistency is not a complete consistency technique for interval algebra.
- Consistency-checking problem for IA networks is an NP-complete problem.
- Intervals can be converted to instants but some interval relations will not be binary relations among the instants.



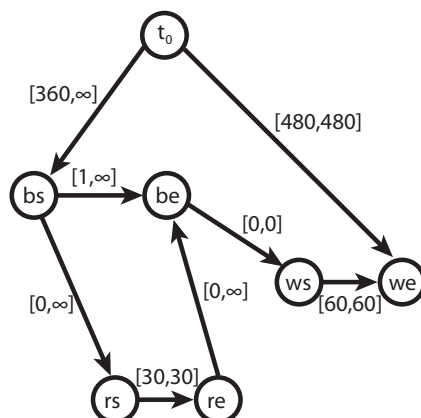
- Points in the ends of interval are not fully translatable to instants.
- "A light bulb is off and after switching the toggle, the light becomes on"*
 - Can be modelled using two intervals *on* and *off* and one interval relation *off* {m} *on*.
 - Is light on or off at the instant between the intervals?
- Qualitative algebra** uses interval and instants as first-order objects:

p before i (i after p)	$p < i^-$	
p starts i (i started-by p)	$p = i^-$	
p during i (i includes p)	$i^- < p, p < i^+$	
p finishes i (i finished-by p)	$p = i^+$	
p after i (i before p)	$i^+ < p$	

Quantitative approach

"I got up at 6 o'clock. I read newspapers for 30 minutes during the breakfast. After the breakfast I walked to my office which took me one hour. I entered the office at 8:00AM".

When did I start my breakfast?



- $360 \leq bs$, "I got up at 6 o'clock"
- $bs \leq rs$, $re \leq be$, "I read newspapers during breakfast"
- $re - rs = 30$, "I read newspapers for 30 minutes"
- $be = ws$, "after breakfast I walked to my office"
- $we - ws = 60$, "[walking] took me one hour"
- $we = 480$, "I entered the office at 8:00AM"

$$bs \leq rs = re - 30 \leq be - 30 = ws - 30 = (we - 60) - 30 = 390$$

I started my breakfast between 6:00AM and 6:30AM.



- The basic temporal primitives are again **time points**, but now the relations are numerical.
- Simple **temporal constraints** for instants t_i and t_j :
 - unary: $a_i \leq t_i \leq b_i$
 - binary: $a_{ij} \leq t_i - t_j \leq b_{ij}$,
where a_i, b_i, a_{ij}, b_{ij} are (real) constants

Notes:

- Unary relation can be converted to a binary one, if we use some fix origin reference point t_0 .
- $[a_{ij}, b_{ij}]$ denotes a constraint between instants t_i and t_j .
- It is possible to use disjunction of simple temporal constraints.

Simple Temporal Network (STN)

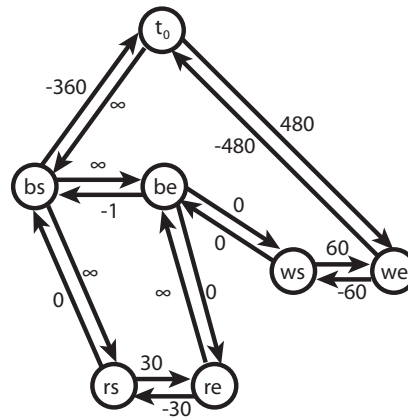
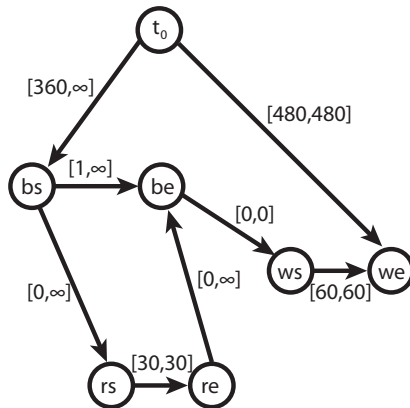
- only simple temporal constraints $r_{ij} = [a_{ij}, b_{ij}]$ are used
- **operations**:
 - composition: $r_{ij} \bullet r_{jk} = [a_{ij} + a_{jk}, b_{ij} + b_{jk}]$
 - intersection: $r_{ij} \cap r'_{ij} = [\max\{a_{ij}, a'_{ij}\}, \min\{b_{ij}, b'_{ij}\}]$
- **STN is consistent** if there is an assignment of values to instants satisfying all the temporal constraints.
- **Path consistency** is a complete technique making STN consistent (all inconsistent values are filtered out, one iteration is enough). Another option is using all-pairs minimal distance **Floyd-Warshall algorithm**.

Relations $a_{ij} \leq t_i - t_j \leq b_{ij}$ can be expressed as maximal distances between the time points:

- $t_i - t_j \leq b_{ij}$
- $t_j - t_i \leq -a_{ij}$

This gives a **distance graph**.

- Negative cycle in the distance graph means inconsistency.



• Path consistency

- finds a transitive closure of binary relations r
- one iteration is enough for STN (in general, it is iterated until any domain changes)
- works incrementally

one iteration for STN

```

PC(X, C)
  for each  $k : 1 \leq k \leq n$  do
    for each pair  $i, j : 1 \leq i < j \leq n, i \neq k, j \neq k$  do
       $r_{ij} \leftarrow r_{ij} \cap [r_{ik} \cdot r_{kj}]$ 
      if  $r_{ij} = \emptyset$  then exit(inconsistent)
    end
  end
end
  
```

general

```

PC(C)
  until stabilization of all constraints in  $C$  do
    for each  $k : 1 \leq k \leq n$  do
      for each pair  $i, j : 1 \leq i < j \leq n, i \neq k, j \neq k$  do
         $c_{ij} \leftarrow c_{ij} \cap [c_{ik} \cdot c_{kj}]$ 
        if  $c_{ij} = \emptyset$  then exit(inconsistent)
      end
    end
  end
end
  
```

• Floyd-Warshall algorithm

- finds minimal distances between all pairs of nodes
- First, the temporal network is converted into a **distance graph**
 - there is an arc from i to j with distance b_{ij}
 - there is an arc from j to i with distance $-a_{ij}$.
- STN is consistent iff there are no negative cycles in the graph, that is, $d(i, i) \geq 0$

```

Floyd-Warshall(X, E)
  for each  $i$  and  $j$  in  $X$  do
    if  $(i, j) \in E$  then  $d(i, j) \leftarrow l_{ij}$  else  $d(i, j) \leftarrow \infty$ 
     $d(i, i) \leftarrow 0$ 
  end
  for each  $i, j, k$  in  $X$  do
     $d(i, j) \leftarrow \min\{d(i, j), d(i, k) + d(k, j)\}$ 
  end
end
  
```


*“I got up at 6 o’clock. I read newspapers for 30 minutes during the breakfast. After the breakfast I walked to my office which took me one hour. I entered the office exactly at the same time as Peter who left his home at 7:00AM. **Peter is going to office either by a car, which takes him 15-20 minutes, or by a bus, which takes 40-50 minutes**”.*

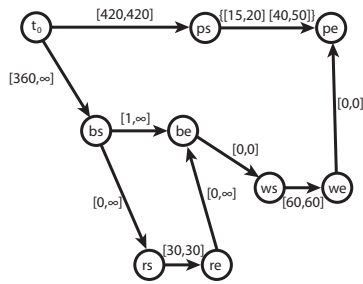
We need to express a disjunction of simple temporal constraints between the same pair of temporal points:

$$40 \leq pe-ps \leq 50 \vee 15 \leq pe-ps \leq 20$$

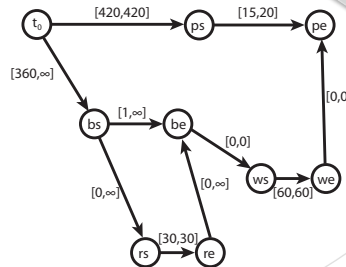


Temporal Constraint Network (TCSP)

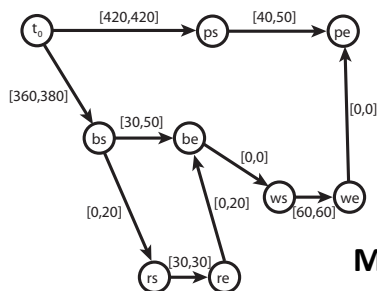
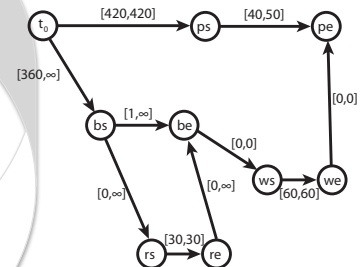
- It is possible to use **disjunctions of simple temporal constraints** over the same variable.
- Operations \bullet and \cap are being done over the sets of intervals.
- **TCSP** is **consistent** if there is an assignment of values to instants satisfying all the temporal constraints.
- Path consistency does not guarantee in general the consistency of the TCSP network!
- A straightforward **approach** (constructive disjunction):
 - decompose the temporal network into several STNs (component STNs) by choosing one disjunct for each constraint
 - solve obtained STN separately (find the minimal network)
 - combine the result with the union of the minimal intervals



Temporal constraint network



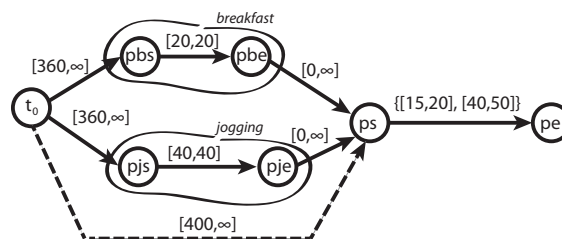
Component STNs



Minimal TCSP



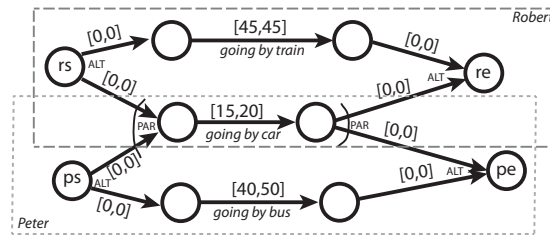
“Peter got up at 6 o’clock and before leaving home we went jogging for 40 minutes and had breakfast which took him 20 minutes. Peter is going to office either by car, which takes him 15-20 minutes, or by a bus, which takes 40-50 minutes.”



- We need to express that jogging and breakfast do not overlap in time!

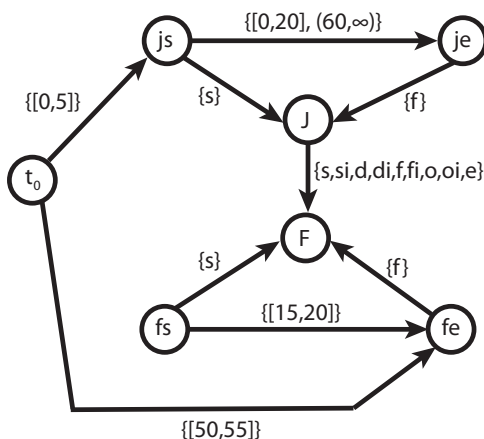
$$pbe \leq pjs \vee pje \leq pbs$$
- This is a so called a **Disjunctive Temporal Problem** (opposite to TCSP, n-ary disjunctions can be used).
- DTN can be solved similarly to TCSP – by decomposition to component STNs.

“When Peter goes by car then Robert joins him, otherwise Robert goes by train which takes him 45 minutes.”

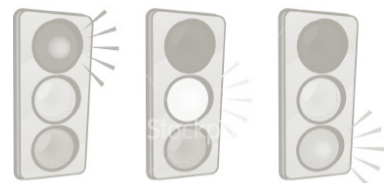


- We need to express that some points do not appear in the network by adding branching (logical) constraints.
- **Temporal Network with Alternatives** assumes parallel/alternative branching constraints in addition to temporal constraints.
 - Solution consists of selection of nodes satisfying the branching and temporal constraints.

“John and Fred work for a company that has local and main offices in Los Angeles. They usually work at the local office, in which case it takes John less than 20 minutes and Fred 15–20 minutes to get to work. Twice a week John works at the main office, in which case his commute to work takes at least 60 minutes. Today John left home between 7:00–7:05 a.m., and Fred arrived at work between 7:50–7:55 a.m. We also know that Fred and John met at a traffic light on their way to work.”



General Temporal Constraint Network combines points and intervals and supports constraints from the qualitative algebra and a from a TCSP.



	name	approach	temporal reference	temporal propositions	complexity
PA	point algebra	qualitative	time points	{<,<=,>}	tractable
IA	interval algebra	qualitative	intervals	{b,m,o,s,d,f,e,bi,mi,oi,si,di,fi}	NP-c
QA	qualitative algebra	qualitative	time points, intervals	IA, PA, interval-to-point	NP-c
STP	simple temporal problem	quantitative	time points	binary difference	tractable
TCSP	temporal CSP	quantitative	time points	binary disjunctive difference	NP-c
DTP	disjunctive temporal problem	quantitative	time points	n-ary disjunctive difference	NP-c
TNA	temporal network with alternatives	quantitative	time points	precedence, logical	NP-c
	general temporal CSP	qualitative, quantitative	time points, intervals	TCSP, QA	NP-c

Extensions of Temporal Frameworks

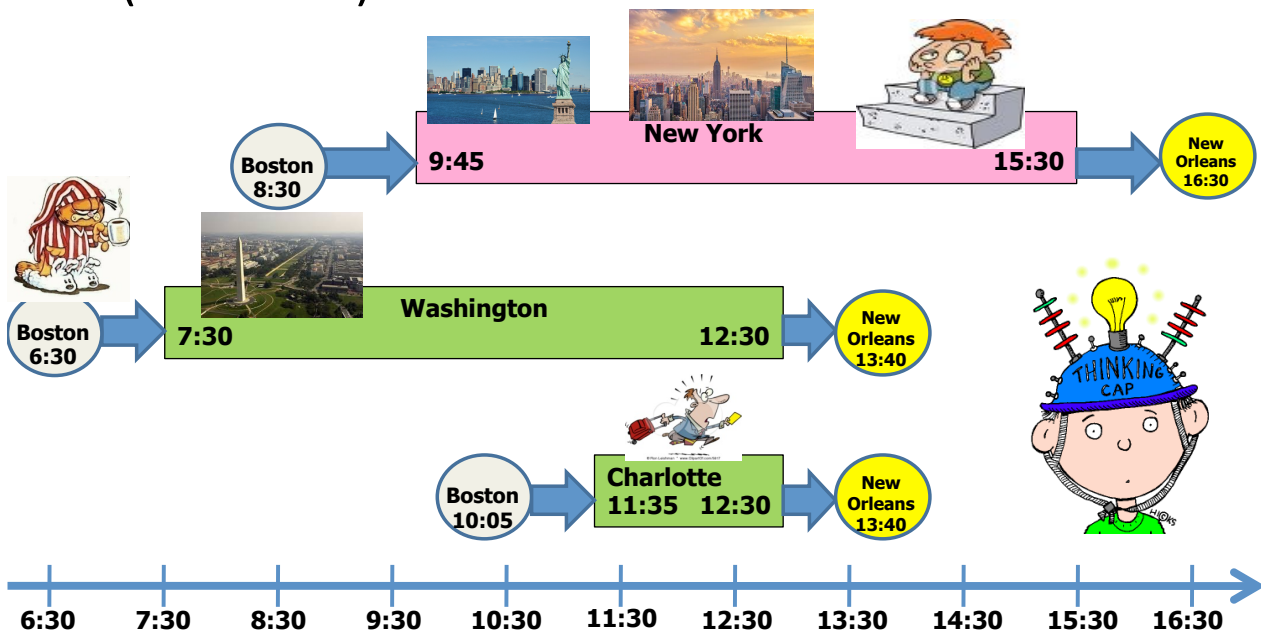
Preferences and Uncertainty



- Preferences
 - Quantitative
 - Qualitative
- Uncertainty
 - Controllability
 - Conditional temporal problems
- Preferences and Uncertainty

PREFERENCES

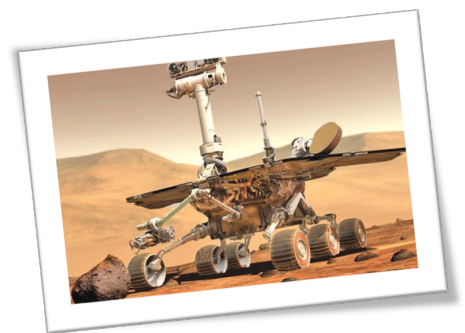
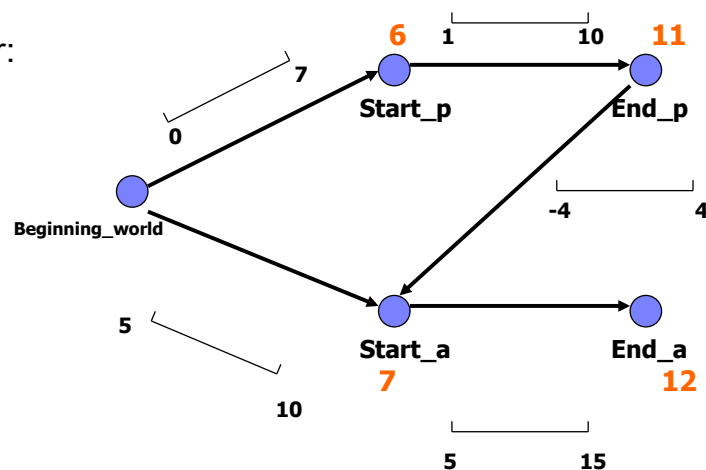
- I have to be in my office by Friday morning!
- Some flights on Thursday with the same (minimum) cost:



PREFERENCES IN QUANTITATIVE TEMPORAL FRAMEWORKS

Two activities of Mars Rover:

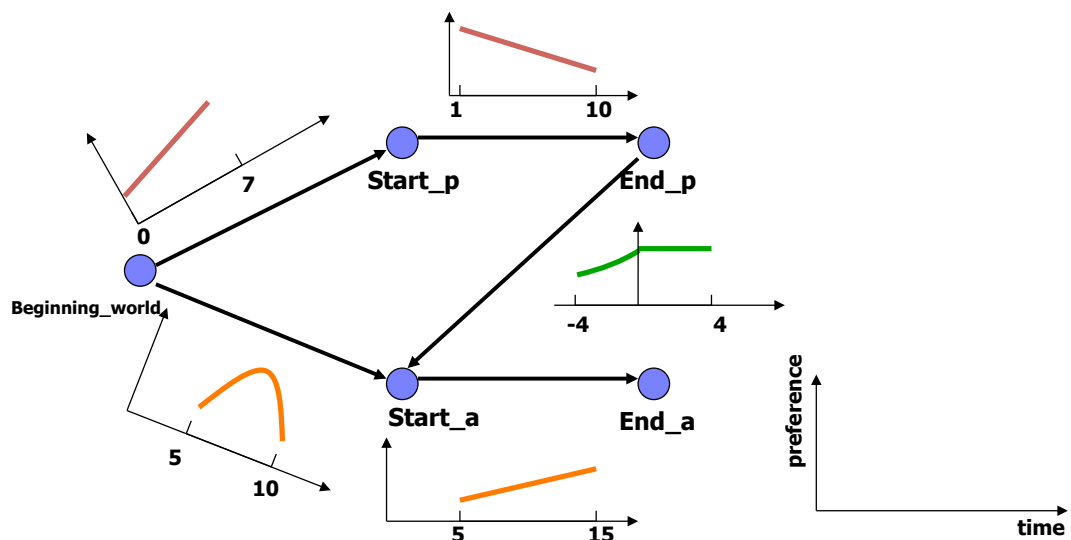
- Taking pictures:
 - $1 < \text{duration} < 10$
 - $0 < \text{start} < 7$
- Analysis:
 - $5 < \text{duration} < 15$
 - $5 < \text{start} < 10$
- Additional constraint:
 - $-4 < \text{start analysis} - \text{end pictures} < 4$
- One of the solutions



Introducing preferences

Sometimes hard constraints aren't expressive enough. We may think that:

- ▶ It's better for the **picture** to be taken **as late as possible** and as fast as possible.
- ▶ It's better if the **analysis** starts **around 7** and lasts as long as possible.
- ▶ It's ok if the two activities **overlap** but it's better if they **don't**.

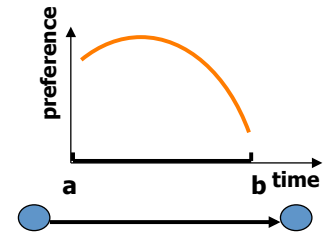


- **Simple Temporal Problem with Preferences**

- Simple Temporal Problem
 - Set of variables X_1, \dots, X_n ;
 - Constraints $T = \{I\}$, $I = [a, b]$ $a \leq b$;
 - Unary constraint T over variable X : $a \leq X \leq b$;
 - Binary constraint T over X and Y : $a \leq X - Y \leq b$;
- C-semiring $S = \langle A, +, \times, 0, 1 \rangle$
 - A set of preference values
 - $+$ compares preference values inducing the ordering on A
 - $a \leq b$ if $a + b = b$, a, b in A
 - \times composes preference values

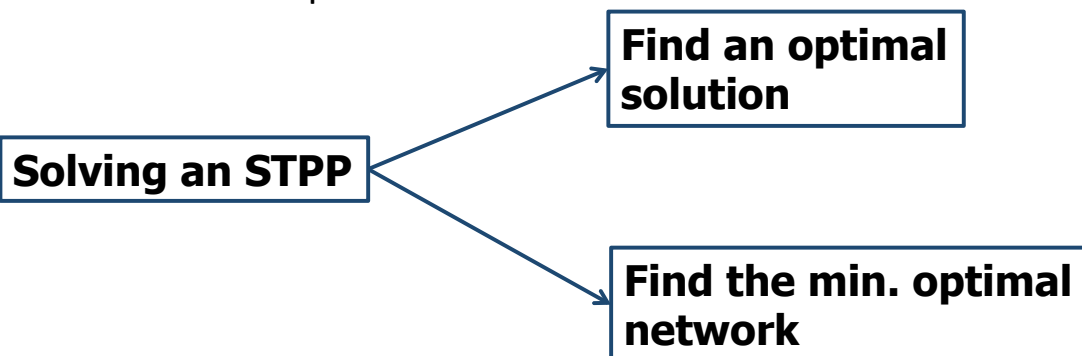
- **Simple Temporal Constraint with Preferences**

- Binary constraint
- Interval $I = [a, b]$, $a \leq b$
- Function $f: I \rightarrow A$



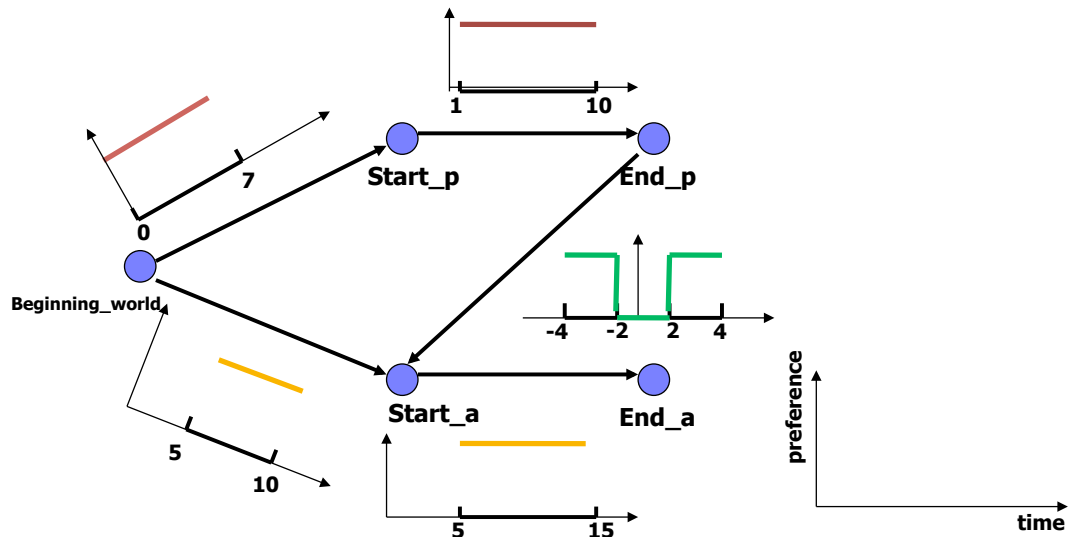
What does solving an STPP mean?

- A **solution** is a complete assignment to all the variables consistent with all the constraints.
- Every solution has a **global preference value** induced from the local preferences.



The class of STPPs is NP-hard.

Any TCSP can be reduced to an STPP



Tractability conditions for STPPs

- 1) The underlying semiring has an **idempotent** multiplicative operator (**x**).

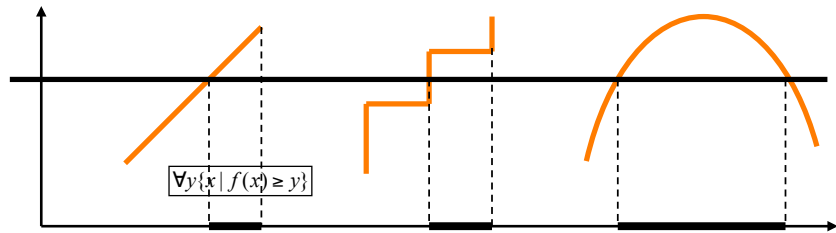
For example:

Fuzzy Semiring $\langle \{x \mid x \in [0,1]\}, \max, \min, 0, 1 \rangle$

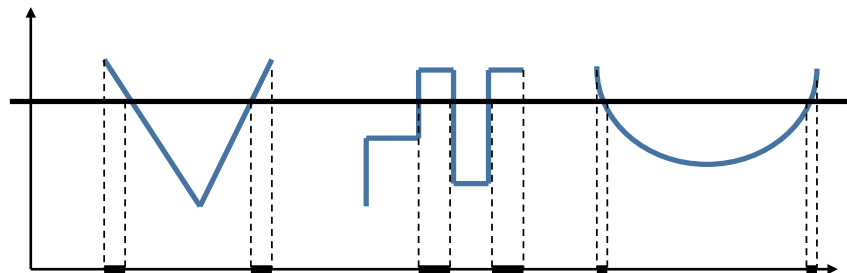
- 2) the preference functions are **semi-convex**
- 3) the set of preferences is **totally ordered**

Examples

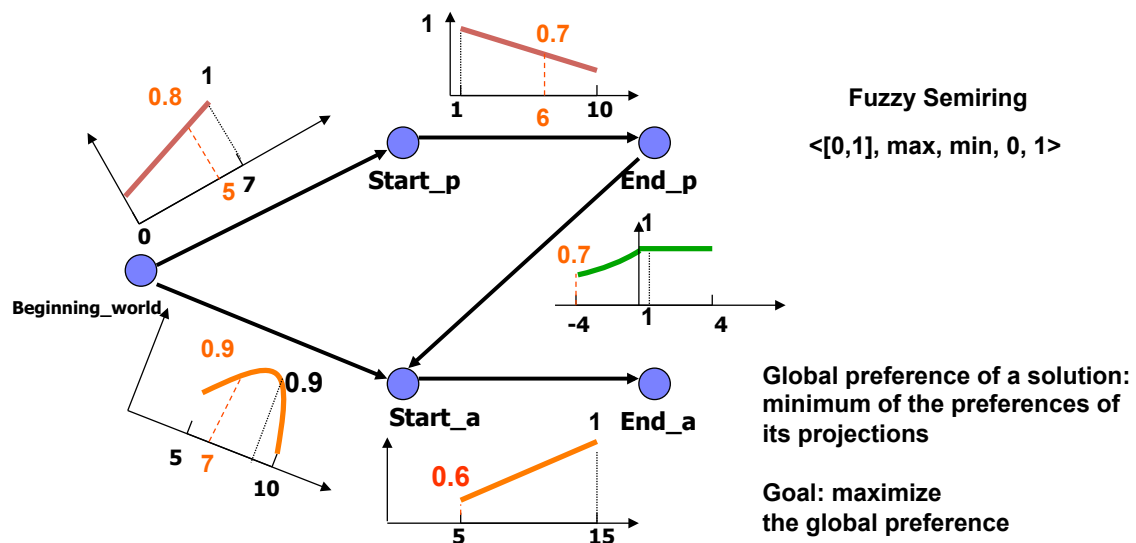
Semi-convex



Non Semi-convex



Solutions of the Rover Example



Two solutions:

Start_p = 5 End_p= 11 Start_a= 7 End_a=12 global preference =0.6

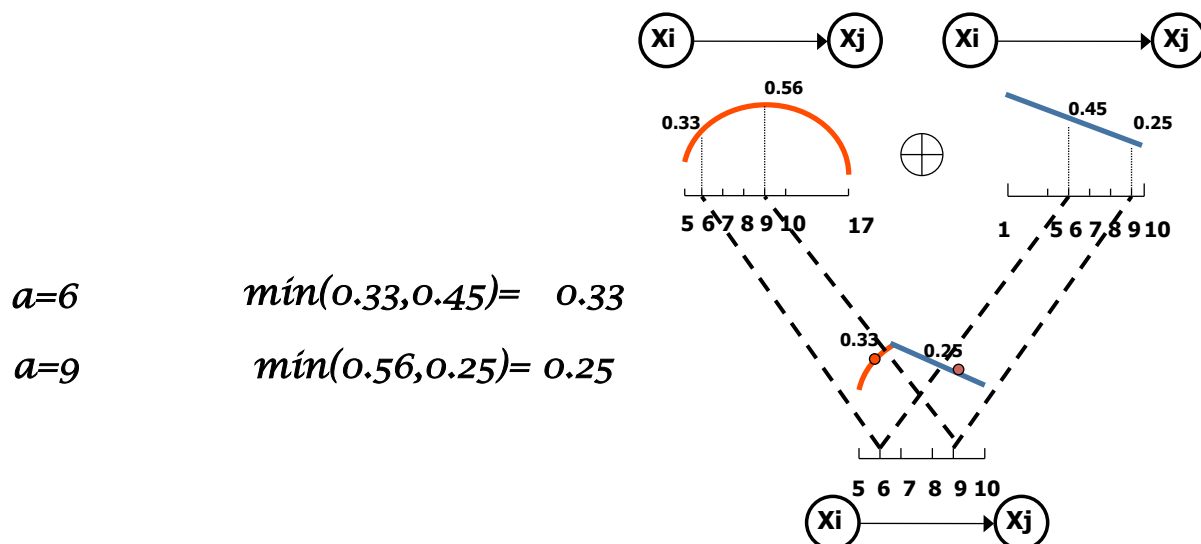
Start_p = 7 End_p= 8 Start_a= 9 End_a=24 global preference =0.9

← **BEST**

- As with hard constraints, two operations on temporal constraints with preferences:
 - Intersection
 - Composition

Intersection

Defined on two constraints on **the same variables**
For each point in the intersection of the intervals,
take the minimum preference



Composition

Defined on two constraints **sharing one variable**

New **constraint on the two not shared variables**

New Interval: all points that can be obtained by **summing a point from each** of the combined **intervals**

Preferences: for each point **maximum of** all preference associated with **decompositions**

If $a=8$

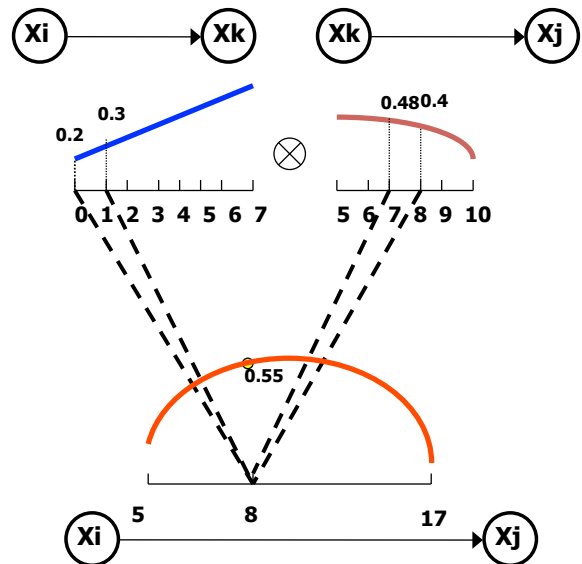
$$r_1=0 \quad r_2=8 \quad \min(0.2, 0.4)=0.2$$

$$r_1=1 \quad r_2=7 \quad \min(0.3, 0.48)=0.3$$

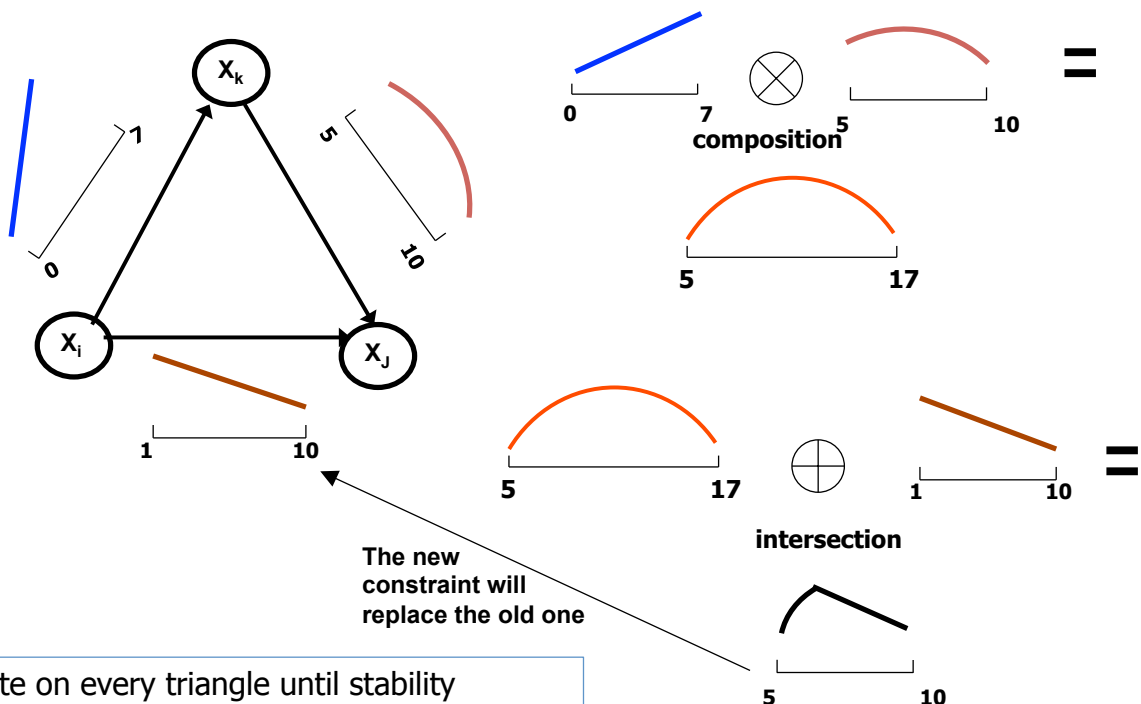
$$r_1=2 \quad r_2=6 \quad \min(0.4, 0.52)=0.4$$

$$r_1=3 \quad r_2=5 \quad \min(0.6, 0.55)=0.55$$

$$\max\{0.2, 0.3, 0.4, 0.55\}=0.55=f_1 \otimes f_2(8)$$



Path Consistency on STPPs

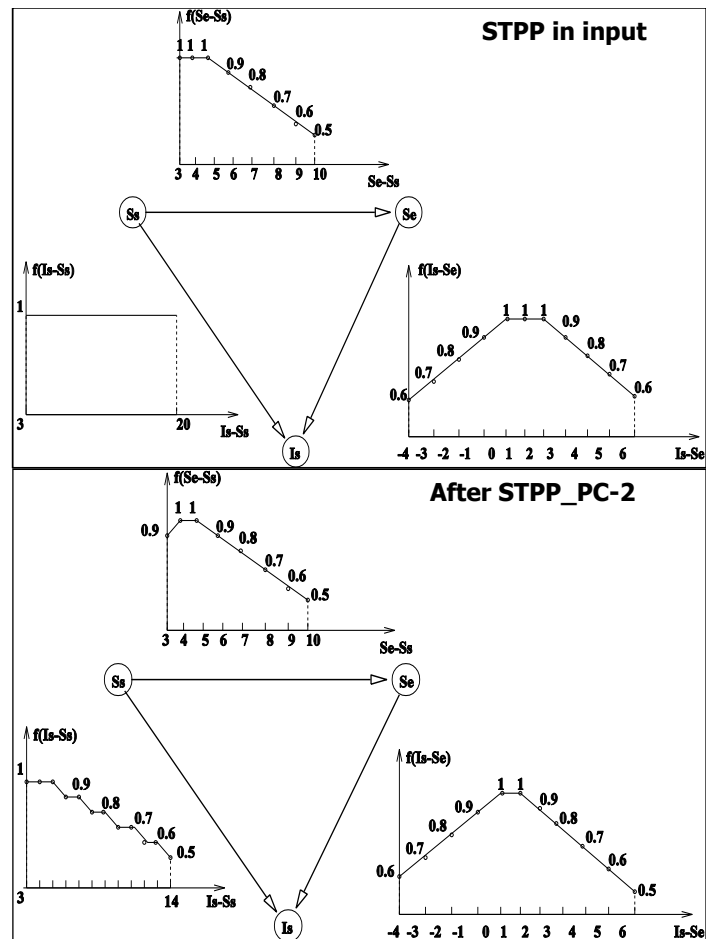


Polynomial: $O(n^3 r^3 l)$ n variable, r max range of an interval, l preference levels

Does PC help?

Given a tractable STPP, path consistency is sufficient to find an optimal solution without backtracking

- Closure of semi-convex functions under intersection and composition
- After enforcing path consistency, if no inconsistency is found, all the preference functions have the same maximum preference level M
- The subintervals mapped into M form an STP in minimal form such that an assignment is a solution of the STP iff it is an optimal solution of the STPP



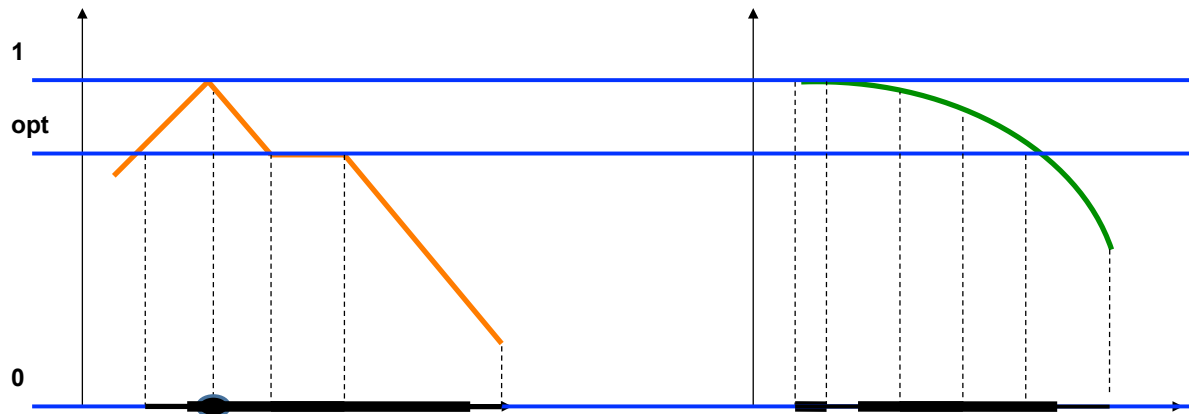
Exploiting fuzziness even more...

In fuzzy theory performing an α -cut means considering **only elements** that are mapped into a **preference greater or equal than α**

Given a tractable STPP and a **preference level y** , the intervals of **elements with preference above y** form an STP: P_y

The highest level, opt , at which STP P_{opt} is consistent is such that an assignment is a solution of P_{opt} iff it is an optimal solution of the STPP

Solving STPPs with alpha-cuts



Cut at level 1 → inconsistent (e.g. due to other constraints not shown)

Cut at level 0 → consistent

....continue until we reach the highest level opt at which cutting gives a consistent STP

Polynomial: $O(n^3l)$ n variable, l preference levels
much faster than using path consistency, less general

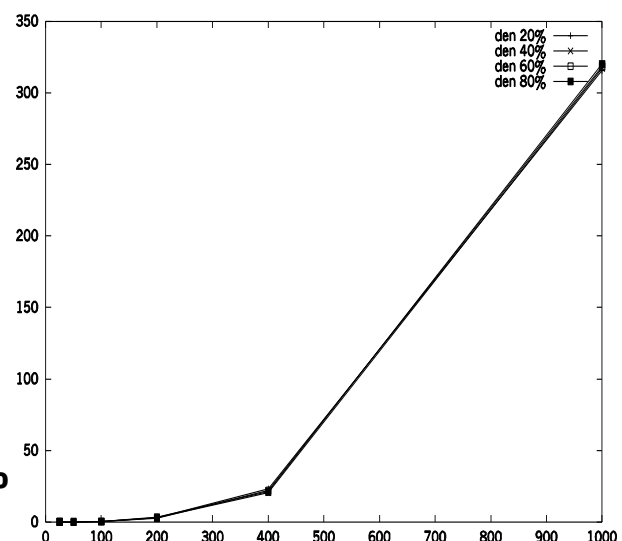
Experimental results for Chop-Solver

X-axis number of variables
Y-axis time in seconds

Fixed parameters:
Range of first solution: 100000
Max expansion: 50000
Perturbation on a: 5%
Perturbation on b: 5%
Perturbation on c: 5%

Varying:
Density 20%, 40%, 60% ,80%

Mean on 10 examples



Time to solve a problem with 40 variables, $r=100$,
 $\max=50$, $p_a=p_b=10\%$ and $p_c=5\%$

Density	Path-solver	Chop-solver
40%	1019.44 sec	0.03 sec
60%	516.24 sec	0.03 sec
80%	356.71 sec	0.03 sec

[Khatib,Morris,Morris, Venable 2003]

Mitigating the fuzzy drowning effect

In Fuzzy CSPs:

Global preference = minimum associated with any of its projections (Drowning Effect)

Fuzzy Optimal: the maximum minimum preference

Pareto Optimal: no other solution with higher preferences
on all constraints

Example: solution $S \langle f_1(S_1)=\mathbf{0.2}, f_2(S_2)=0.3, f_3(S_3)=\mathbf{0.2} \rangle$

solution $S' \langle f_1(S'_1)=0.8, f_2(S'_2)=0.9, f_3(S'_3)=\mathbf{0.2} \rangle$

Fuzzy Optimals: S, S' Pareto Optimals: S'

- Finds Pareto Optimal solution of an STPP by iterating the following 3 steps:
 1. Applying the alpha-cut solver to the problem
 2. Identifying special constraints, the weakest links (the ones that are drowning the preference of the optimal solutions)
 3. Neutralizing the weakest links (by making their preference function irrelevant)
- Polynomial Time

Disjunctive Temporal Problems with Fuzzy Preferences

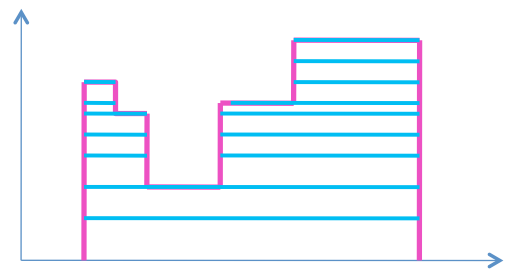
[Pollack, Peintner 04]

- Disjunctive Temporal Constraint = disjunction of STP constraints
 $(X_1 - Y_1 \in [a_1, b_1]) \vee \dots \vee (X_n - Y_n \in [a_n, b_n])$
- Disjunctive Temporal Constraint with Preferences:
 $(X_1 - Y_1 \in [a_1, b_1], f_1) \vee \dots \vee (X_n - Y_n \in [a_n, b_n], f_n), \quad f_i: [a_i, b_i] \rightarrow [0, 1]$
- Fuzzy Optimization criterion
- Algorithm
 1. For each preference level y , in increasing order, starting from 0
 2. cut the DTPP at y obtaining DTP $_y$
 3. solve DTP $_y$ obtaining an STP $_y$
 4. move up one preference and start solving DTP $_{y+1}$ using STP $_y$
- Complexity $|\text{preferences}| \times n^3 \times (\text{DTP complexity})$, n =number of variables

[Pollack, Peintner 2005]

Simple Temporal Problems with Utilities

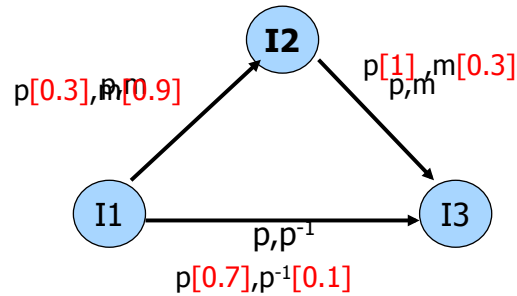
- Constraints as in STPPs (no restriction on the function shape)
- Preferences: positive integers
- Max-plus optimization criterion
 - preference combined by adding them
 - the higher the better
- Algorithms based on mapping the soft constraint into the family of hard constraints deriving from each cut
- Greedy Algorithm (not complete)
 - **Searches for a consistent** STP repeatedly trying to improve by replacing an STP constraint with one corresponding to a higher preference level
- Complete algorithm
 - Performs a complete search over the space of component STPs, using the greedy algorithm, pruning, and a divide et impera strategy
 - Complexity **exponential** (in practice few iterations needed to find a good solution)
- Other techniques
 - Reduction to a SAT problem [Sheine, Peintner, Sakallah, Pollack 2005]
 - Weighted Constraint Satisfaction [Moffitt, Pollack 2005]



Qualitative temporal problems with fuzzy preferences

[Badaloni, Giacomini 2000, 2001, 2002]

- Variables: temporal intervals/ points
- Constraints: subsets of the 13 Allen relations / of $\{<, =, >\}$
- A preference level in $[0,1]$ associated with each relation in the constraint
- $IA^{fuz}, PA^{fuz}, SA^{fuz}, SAC^{fuz}, Pac^{fuz}$
- Redefinition of the main operations (composition and intersection)
- Closure of all the algebras w.r.t. them
- Using alpha-cuts, satisfiability and computation of minimal network remain in the same complexity class
- Combination of qualitative and quantitative fuzzy constraints: [Badaloni, Giacomini and Falda in 2004]



Learning temporal preferences (1)

[Khatib, Morris, Morris, Rossi, Sperduti, Venable 2002]

It can be difficult to have precise knowledge on the preference function for each constraint.

Instead it may be easier to be able to tell how good a solution is.

Global information
some solutions + global
preference values

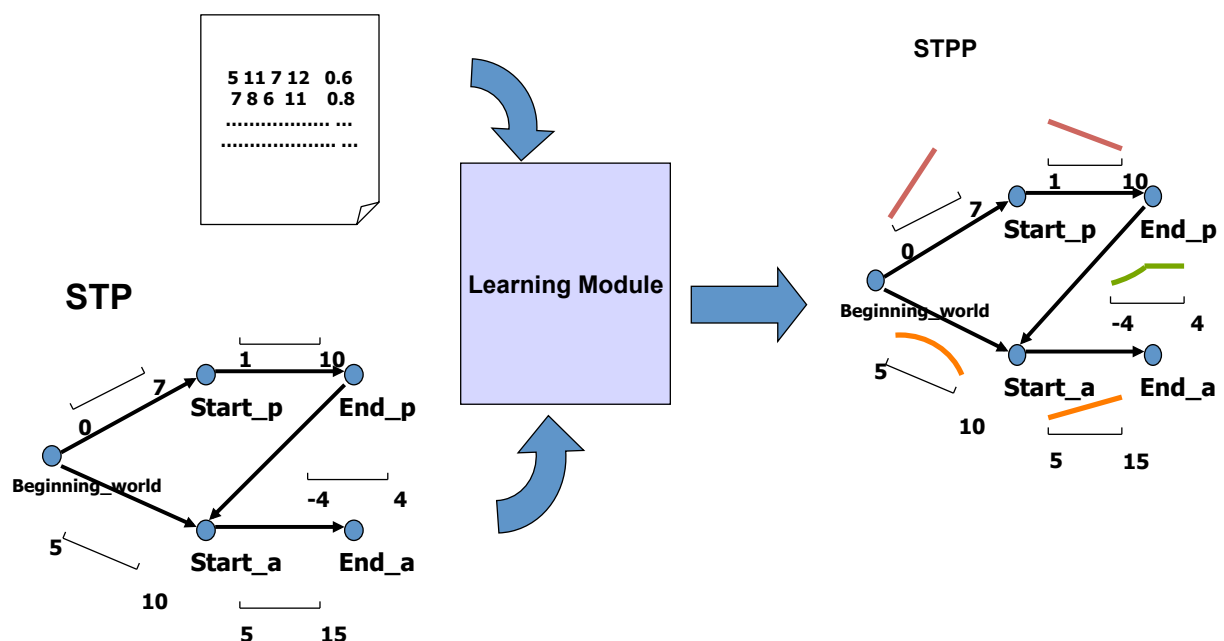


Local Information
shape of preference
functions

- **Inductive Learning:** ability of a system to induce the correct structure of a map t known only for particular inputs
- **Example:** $(x, t(x))$.
- **Computational task:** given a collection of examples (training set) return a function h that approximates t .
- **Approach:** given an error function $E(h, t)$ minimize modifying h .
- In our context :
 - $x \rightarrow$ solution
 - $t \rightarrow$ rating on solutions given by expert
 - Preference function constraint $C_i \rightarrow$ parabola $a_i x^2 + b_i x + c_i$
 - Error $E \rightarrow E(a_1, b_1, c_1, \dots, a_n, b_n, c_n)$
 - Learning technique \rightarrow gradient descent

The STPP Learning Module

Training set



Works with

- **Parabolas** $f(x)=ax^2+bx+c$ as preference functions
- **Fuzzy Semiring** $\langle [0,1], \max, \min, 0, 1 \rangle$ as underlying structure
- **Smooth** version of the **min** function

Performs

- **Incremental gradient descent** on the sum of squares error

$$E = \frac{1}{2} \sum_{s \in T} (t(s) - h(s))^2$$

- $t(s)$ Preference value of solution s in the training set
 $h(s)$ Preference value guessed for solution s from the current network

The Learning Algorithm

- 1) **Read a solution s** and its preference value **$t(s)$** from the training set
- 2) **Compute the preference** value of s , **$h(s)$** , according to the current network
- 3) **Compare $h(s)$ and $t(s)$** using the error function
- 4) **Adjust parameters a, b, c** , of each preference function of each constraint, in order to make the error smaller
- 5) **Compute the global error**; if below threshold, exit, otherwise back to 1)

➤ Delta rule:

$$\tilde{a}_i = a_i - \eta \frac{\partial E}{\partial a_i}(a_1, b_1, c_1, \dots, a_v, b_v, c_v) \quad \text{with } v = \text{number of constr.}$$

$$\tilde{b}_i = b_i - \eta \frac{\partial E}{\partial b_i}(a_1, b_1, c_1, \dots, a_v, b_v, c_v)$$

$$\tilde{c}_i = c_i - \eta \frac{\partial E}{\partial c_i}(a_1, b_1, c_1, \dots, a_v, b_v, c_v)$$

➤ Semi-convexity is maintained during all the learning process

$$\text{if } \tilde{a} < 0 \quad \text{then} \quad \tilde{a} = 0$$

Experimental results

- Varying parameters:
 - density (D)
 - maximum range of interval expansion (max).
- Fixed parameters :
 - number of variables n=25
 - range for the initial solution r=40
 - parabolas perturbations pa=10, pb=10 and pc=5.
- Displayed: absolute mean error ($0 < \text{ame} < 1$) on a test set (mean on 30 examples).
 - 357 ≤ iterations ≤ 3812
 - 2' 31" ≤ time required ≤ 8' 18"

Density _ Maximum Range	D=40	D=60	D=80	Number of examples of training and test set.
max=20	0.017	0.007	0.0077	500
max=30	0.022	0.013	0.015	600
max=40	0.016	0.012	0.0071	700

Learning STPPs: An example with maximum lateness

- Problem: **8 activities** to be scheduled in **24 hours**
- Given:
 - Duration intervals for each activity
 - Constraint graph
- Aim: Minimize the ending time of the last activity scheduled.
- Procedure:
 - 1) Solve the hard constraint problem: **900 solutions**
 - 2) Rate each solution with a function that gives higher preference to schedules that end sooner: **37 optimal solutions**
 - 3) Select **200** solutions for the **training set**, 8 optimal solutions, and **300** for the **test set**.
 - 4) Perform learning: **1545 iterations**.
- Results:
 - Absolute mean error on test set: 0.01
 - Maximum absolute error on test set: 0.04
 - Number of optimal solutions of the learned problem: 252 all rated highly by the original function.
 - Number of unseen optimal solutions recognized by the learned problem: 29.

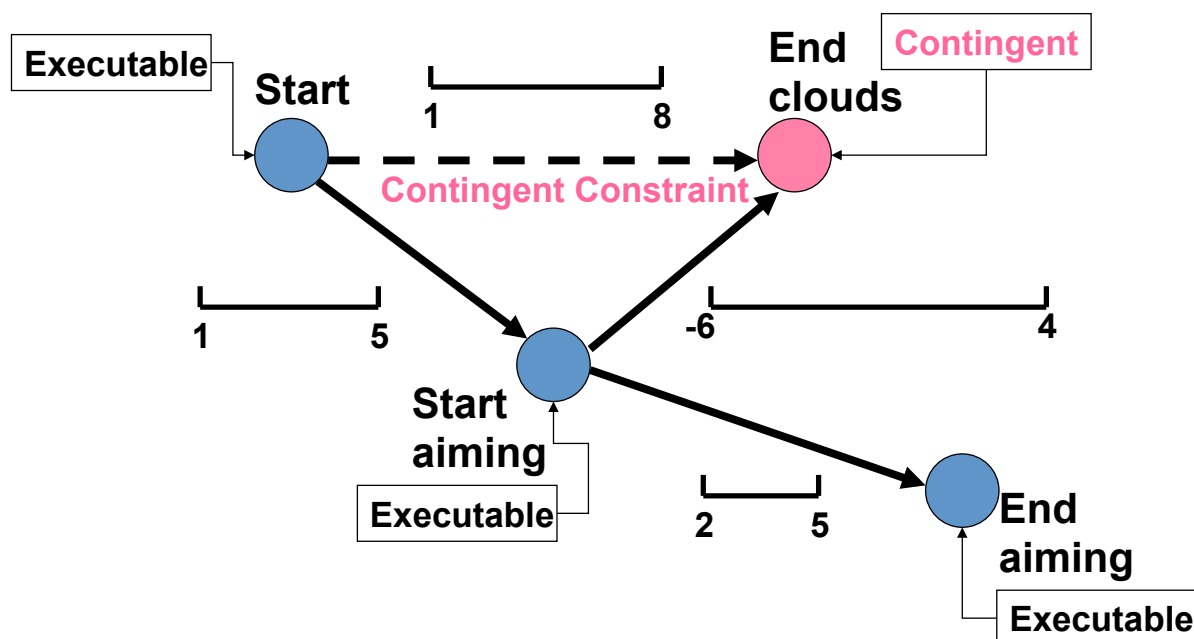
UNCERTAINTY

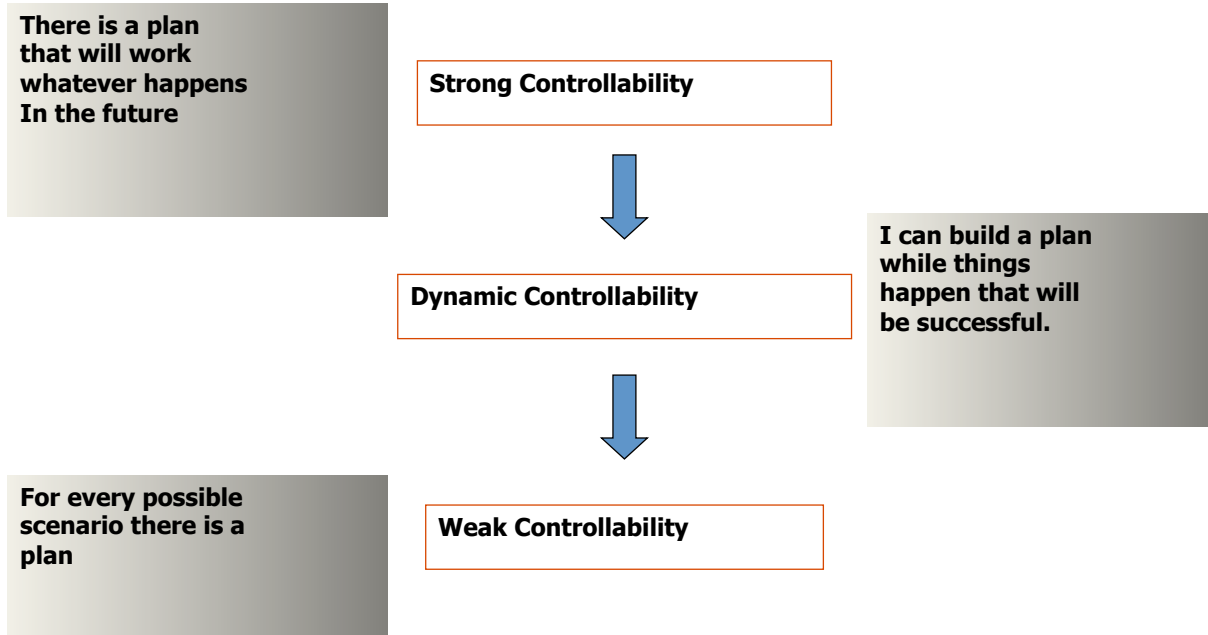
Informally, an STPU is an STP where some of the variables are not under the control of the agent, i.e. the agent cannot decide which value to assign to them.

An STPU:

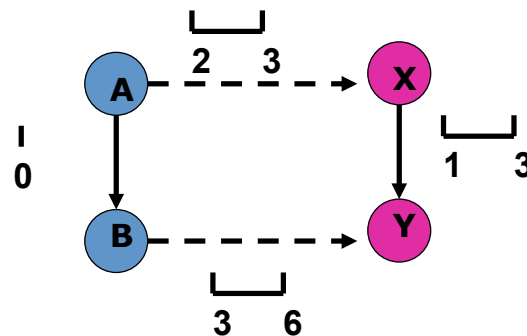
- Set of **executable timepoints** (controllable assignment);
- Set of **contingent timepoints** (uncontrollable assignment);
- Set **requirement constraints** T_{ij} :
 - Binary
 - Temporal interval $I=[a,b]$ meaning $a \leq X_j - X_i \leq b$
- Set of **contingent constraints** T_{hk} :
 - Binary: on an executable X_h and a contingent timepoint X_k
 - Temporal interval $I=[c,d]$ meaning $0 \leq X_k - X_h \leq d$

Example: satellite maneuvering





- Consider the STPU as an STP (forgetting about the distinction between contingent and executable events)
- The STPU is **pseudo-controllable** iff in the minimal network of the associated STP no interval on a contingent constraint is tightened
- Weakly controllable → pseudo-controllable
- Mostly used as a (cheap) initial test
- Not pseudo-controllable → Not weakly-controllable → Not dynamically controllable → Not strongly-controllable

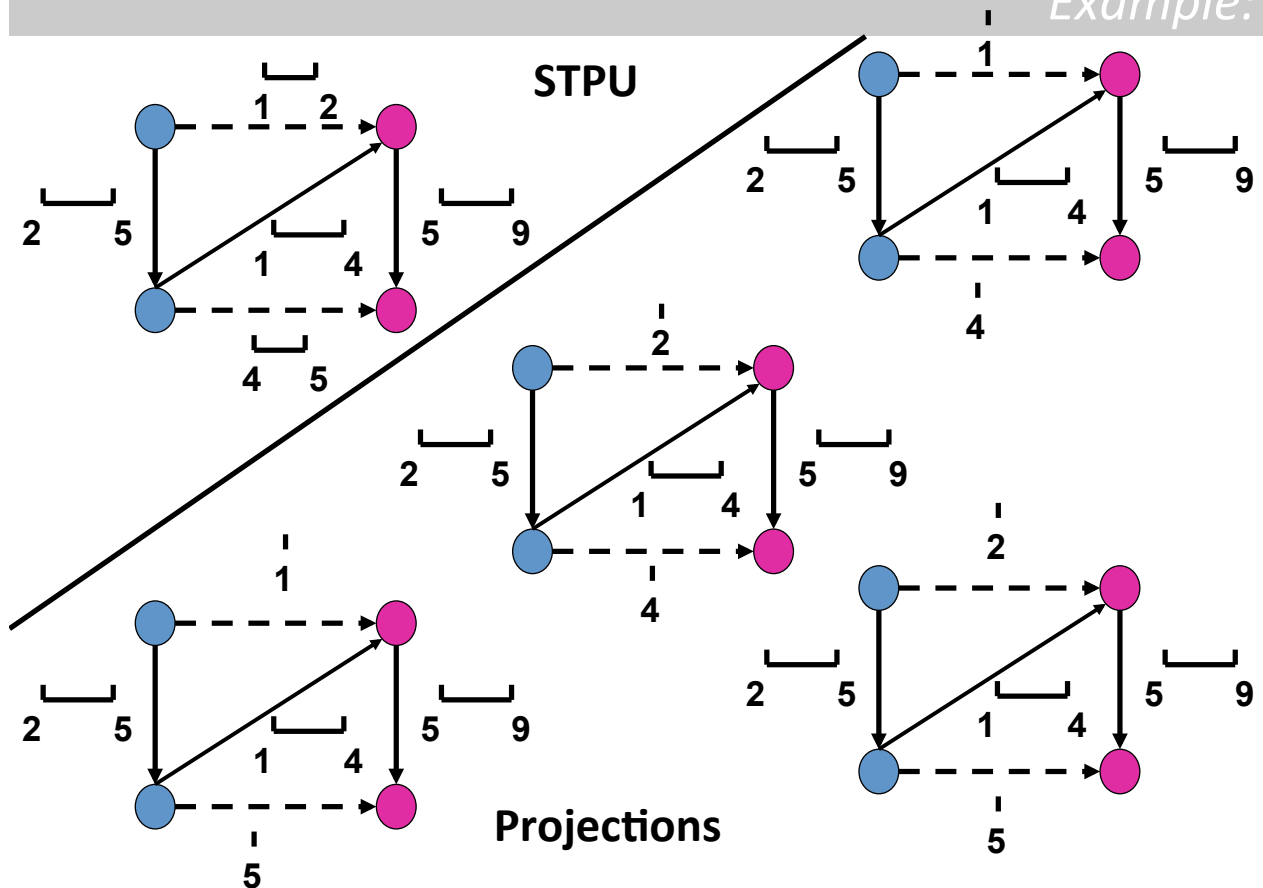


- It is pseudo-controllable
- It is not SC
- It is not WC

STPUs Definitions

- Given an STPU P
- A **control sequence** d is an assignment to the executable timepoints
- A **situation** w is a set of durations on contingent constraints (set of elements of contingent intervals)
- A **schedule** is a complete assignment to the variables of P
- A schedule is **viable** if it is consistent with all the constraints. $Sol(P)$ is the set of all viable schedules of P .
- A **projection** P_w corresponding to situation w is the **STP** obtained replacing each contingent constraint with its duration in w . $Proj(P)$ is the set of all projection of P .
- A **viable strategy** $S: Proj(P) \rightarrow Sol(P)$ maps every projection P_w into a schedule including w

Example:

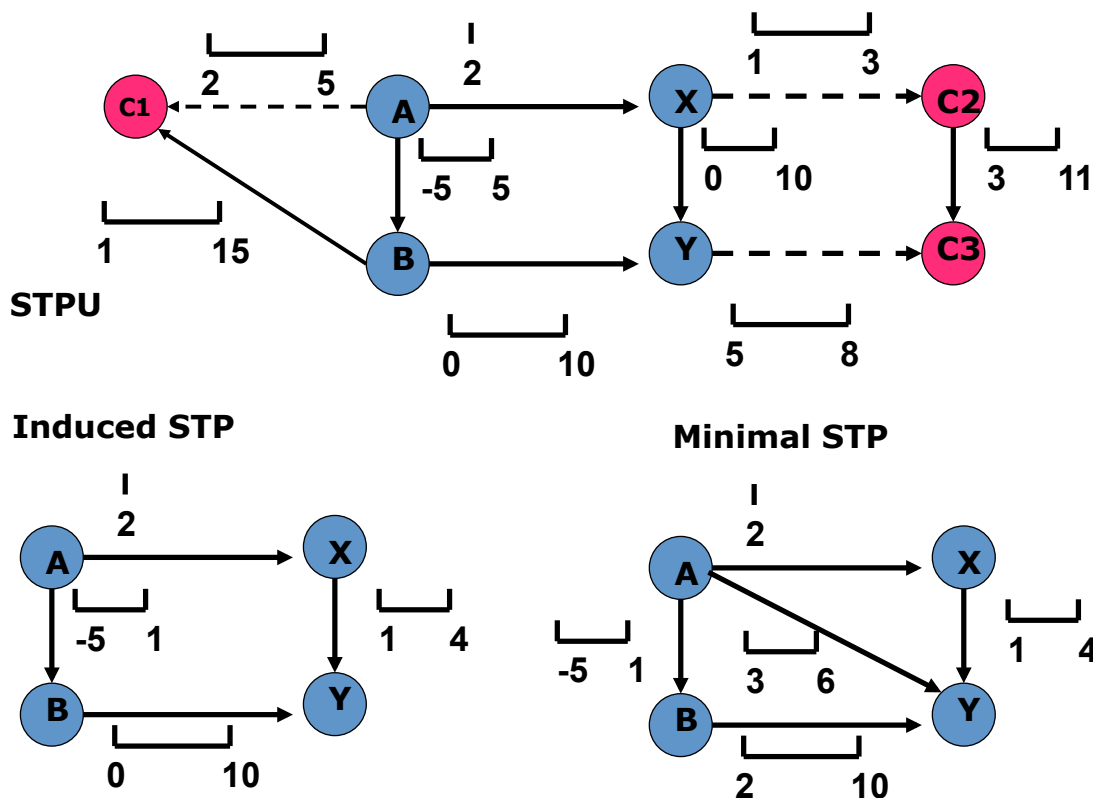


Notation

- $[S(P_w)]_x$: time assigned to executable variable x by schedule $S(P_w)$
- $[S(P_w)]_{<x}$: history of x in $S(P_w)$ is the set of durations corresponding to contingent events which have occurred before $[S(P_w)]_x$

- There must be an assignment to the controllable variables consistent with all possible outcomes of the uncontrollable variables
 - Strongly controlling a contingent event induces new (simple temporal) constraints on executable variables connected to it
 - Solving procedure:
 1. Induce all “controllability” constraints only on executable variables
 2. Remove all contingent variables and constraints involving them
 3. Solve the STP obtained on executable variables
 - **Output: minimal STP such all its solutions are strongly controlling assignments**
 - Polynomial

Example



- For every possible outcome of uncontrollable variables, there is a way to choose controllable variable that is consistent
 - No need to check all possible outcomes, **just** the ones corresponding to the **bounds of intervals**
 - Solving procedure:
 - For all possible combinations of bounds of contingent constraints:
 - 1. Fix contingent constraints to selected bounds
 - 2. Solve the STP obtained
 - Exponential

Satisfiability Modulo Theory encoding of controllability

[Cimatti, Micheli, Roveri 2012, 2013]

- Recently proposed: alternate method for strong and weak controllability based on
Satisfiability Modulo Theory (SMT)
- Fragments of first order logics where interpretations are constrained to satisfy a specific theory (e.g. linear Real arithmetic) extended with quantifiers
- SMT solver: SAT Solver + Constraint Solver

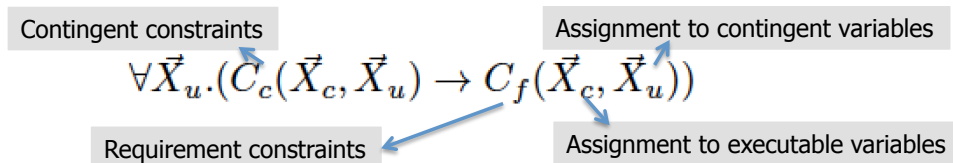
SMT encoding of controllability(2)

- Time point \rightarrow SMT variable, a Real variable in this case
- **Consistency** \rightarrow Satisfiability of an SMT formula:

$$C_f(\vec{X}_c) \doteq \bigwedge_{i=1}^{|C_f|} \bigvee_{j=1}^{D_i} (((x_{i,j} - y_{i,j}) \geq l_{i,j}) \wedge ((x_{i,j} - y_{i,j}) \leq u_{i,j}))$$

- Encoding linear in the size of the temporal problem
- Holds for DTPs, TCSPs, STPs
- Can be improved by
 - encodings which give a CNF result
 - Emphasizing mutual exclusion in TCSP constraints

- Encoding **Strong Controllability** in quantified SMT formula



- Quantifiers can be then removed to allow the application of SMT solvers that don't support them

SMT encoding of controllability(3)

- Encoding weak controllability:

$$\forall \vec{y}. \exists \vec{x}. (\Gamma(\vec{y}) \rightarrow \Psi(\vec{x}, \vec{y})) \quad \exists \vec{y}. (\Gamma(\vec{y}) \wedge \neg \exists \vec{x}. \Psi(\vec{x}, \vec{y}))$$

$$\Gamma(\vec{y}) \doteq \bigwedge_{k=1}^m (y_k \geq 0) \wedge (y_k \leq (u_k - l_k)) \quad \Psi(\vec{x}, \vec{y}) \doteq \bigwedge_{c \in C_f} c(\vec{x}, \vec{y})$$

- Encoding by refutation: non constructive
- Restriction to the class of **linear strategies**: start times of controllable activities are a linear combination of durations of uncontrollable events
- \rightarrow Encoding to SMT for the theory of quantified non-linear polynomials

- Encoding weak controllability:

$$\exists \vec{y}. (\Gamma(\vec{y}) \wedge \neg \exists \vec{x}. \Psi(\vec{x}, \vec{y}))$$

$$\Gamma(\vec{y}) \doteq \bigwedge_{k=1}^m (y_k \geq 0) \wedge (y_k \leq (u_k - l_k)) \quad \Psi(\vec{x}, \vec{y}) \doteq \bigwedge_{c \in C_f} c(\vec{x}, \vec{y})$$

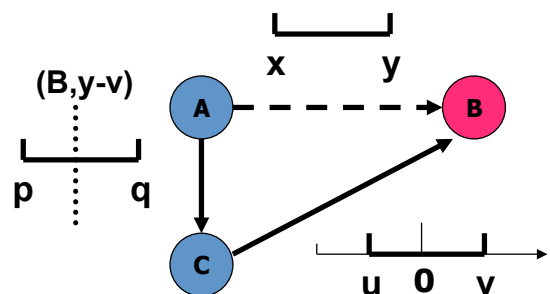
- Encoding by refutation: non constructive
- Restriction to the class of **linear strategies**: start times of controllable activities are a linear combination of durations of uncontrollable events
- \rightarrow Encoding to SMT for the theory of quantified non-linear polynomials

- It must be possible determine when to execute a controllable variable in a sequential fashion, based only on the time at which previous controllable and uncontrollable variables have been executed, without backtracking
- Solving procedure
 - Based on the concept of one event having to **wait** for another for a given time
 - **Output: STP + waits**
 - **Two fundamental operations: reductions and regressions**
 - Polynomial

- Inducing DC constraints on executables from contingent constraints and dynamic controllability requirements
- Like for SC but now the induced constraints can be ternary

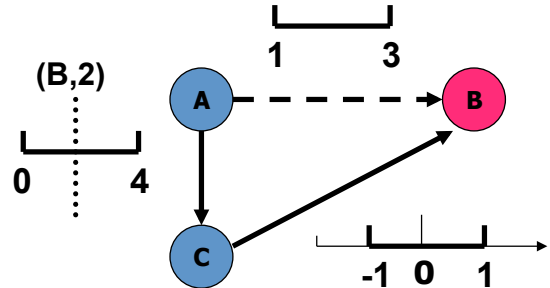
Example: Unordered reduction

- Only when $u < 0$ and $v \geq 0$
- C before or after B
- Impose a wait (B, y') , $y' = y - v$ on AC
- Wait (B, y') on AC means:
 - Either B occurs and thus C can be immediately executed or
 - C can be safely executed after $T_A + y'$ regardless if B has executed or not
- Wait \rightarrow ternary constraint on A, C and B



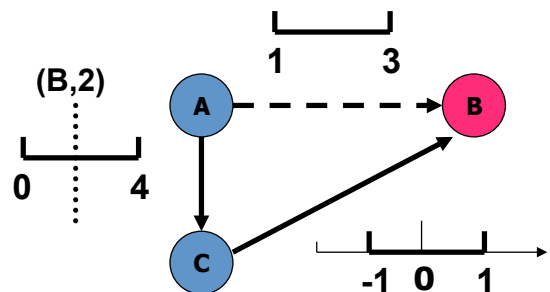
Example: Unordered reduction

- $u < 0$ and $v \geq 0$, since $u = -1$ and $v = 1$
- C before or after B
- Impose a wait (B, y') , $y' = 3 - 1 = 2$ on AC
- Assuming $T_A = 0$
- Either
- If B occurs at 1 C can be immediately executed at 1 or 2
- Otherwise C can be safely executed after $T_A + 2 = 2$ regardless if B has executed or not
- If we do not respect the wait and we execute C at 1 and then B occurs at 3 \rightarrow CB is violated



Example: Unordered reduction

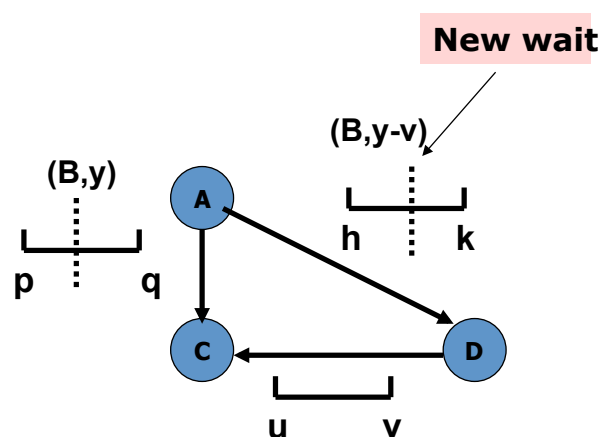
- Since $u = -1$ and $v = 1 \rightarrow$ C before or after B
- Impose a wait (B, y') , $y' = 3 - 1 = 2$ on AC
- Assuming $T_A = 0$
- Either
- If B occurs at 1 C can be immediately executed at 1 or 2
- Otherwise C can be safely executed after $T_A + 2 = 2$ regardless if B has executed or not
- If we do not respect the wait and we execute C at 1 and then B occurs at 3 \rightarrow CB constraint is violated



- A regression is a **propagation of a wait from one constraint to another**
- The regression of the wait is **from requirement constraint to requirement constraint**
- However it can be **caused by another requirement or contingent constraint**

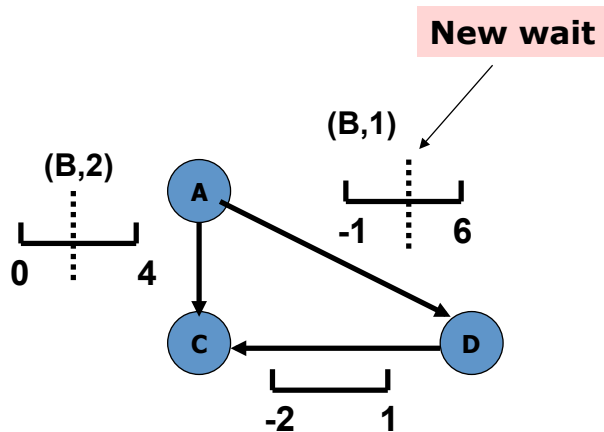
Example: Simple regression

- Given that
- C has to wait either for B to occur or for y time units to pass after A
- D has to occur at most v time units after C
- Then, it follows that
- D must wait either for B to occur or for $y-v$ time units to pass after A



Example: Simple regression

- No contingent constraints involved
- Given that
- C has to wait either for B to occur or for 2 time units to pass after A
- C can occur at most 1 after D
- Then, it follows that
- D must wait either for B to occur or for 1 time unit to pass after A

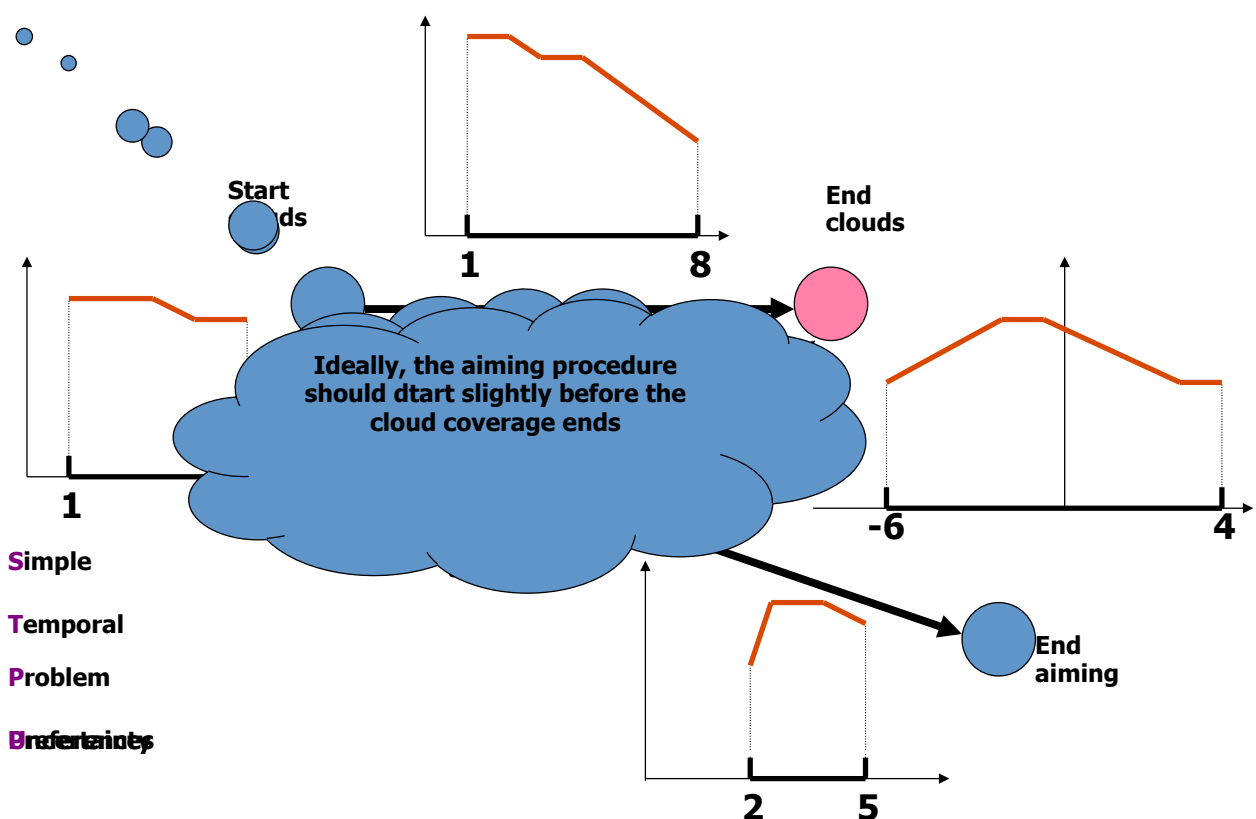


Complexity of Classical-DC

- **[Morris and Muscettola 2001]: $O(n^3r)$** complexity is **pseudo-polynomial** since it is polynomial if we assume the maximum link size is bounded
- **[Morris and Muscettola 2005]: $O(n^5)$** new algorithm truly polynomial
 - Reductions and regressions on constraint graph \rightarrow reductions on distance graph
 - Pseudo-controllability \rightarrow consistency of the AllMax projection
 - Cut-off Bound as in Bellman-Ford replacing termination dependence on domain size
- **[Morris 2006]: $O(n^4)$** Improves on previous one by targeting specific edges
- **[Hunsberger 2010]: $O(n^4)$ incremental** real-time algorithm which allows **execution** directly on the networks
- **[Morris 2014]: $O(n^3)$ yet a further improvement, obtained by considering specific paths**
- **[Mikael Nilsson, Jonas Kvarnström and Patrick Doherty 2014 at ICAPS]: $O(n^3)$ incremental algorithm**

STPS WITH PREFERENCES AND UNCERTAINTY

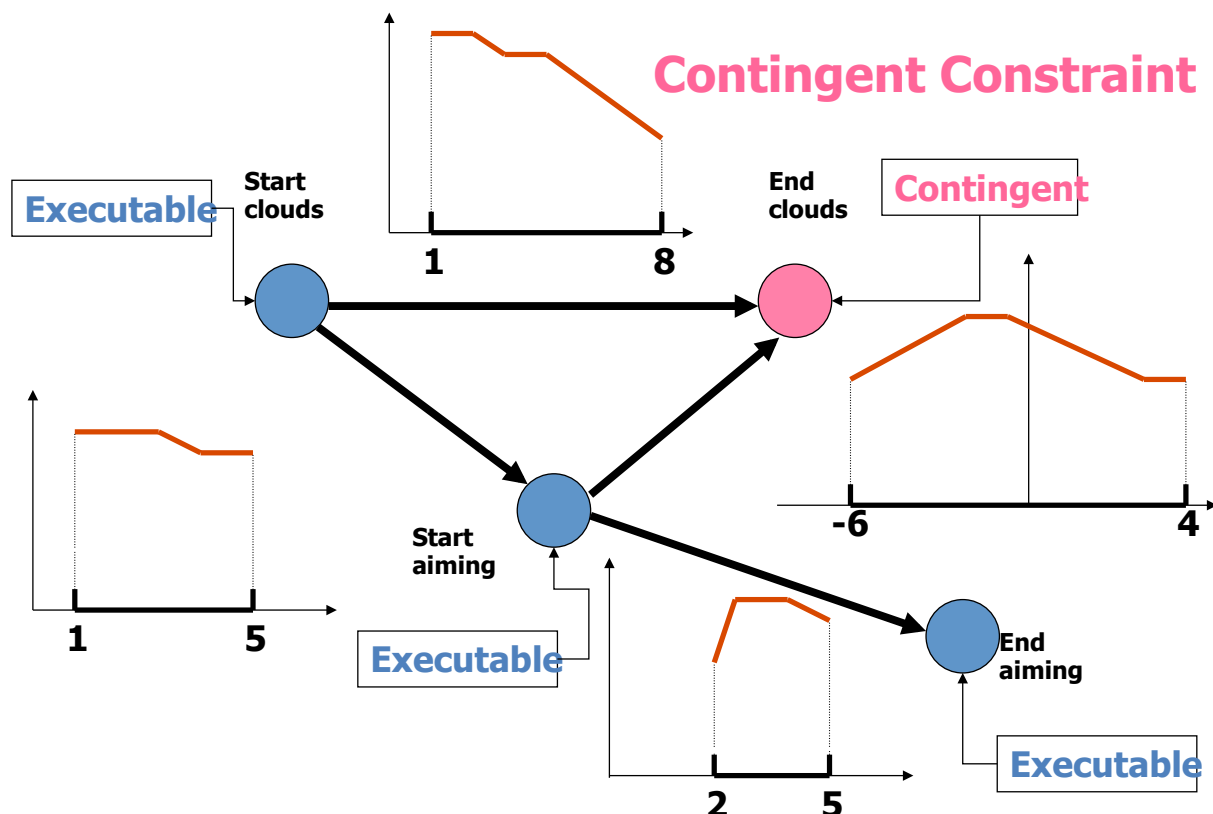
Example: satellite manoeuvring



An STPPU:

- Set of **executable timepoints** (controllable assignment);
- Set of **contingent timepoints** (uncontrollable assignment);
- Set of **soft requirement constraints**:
 - Binary
 - Temporal interval I
 - Preference function $f: I \rightarrow A$;
- Set of **soft contingent constraints**:
 - Binary: on an executable and a contingent timepoint
 - Temporal interval I
 - Preference function $f: I \rightarrow A$
- C-Semiring $\langle A, +, \times, 0, 1 \rangle$

Example: satellite manouvering



A solution of an STPPU is a complete assignment to all the timepoints.

**Solution S = (Assignment to executables S_E ,
Assignment to contingents S_C)**

Every solution has a **preference value**:

$$\text{Pref}(S) = f_1(S_1) \times \dots \times f_n(S_n)$$

f_i = preference function of i-th constraint

S_i = projection of S on i-th constraint

We assume the STP tractability conditions hold

Optimal Strong Controllability (OSC) of STPPUs

An **STP with Preferences and Uncertainty** is **Optimally SC** if there is an assignment to all the executable time points **consistent and optimal** with all the possible scenarios.

Optimal = the assignment to executables completed with any assignment to contingents has the best preference value.

Checking Optimal Strong Controllability

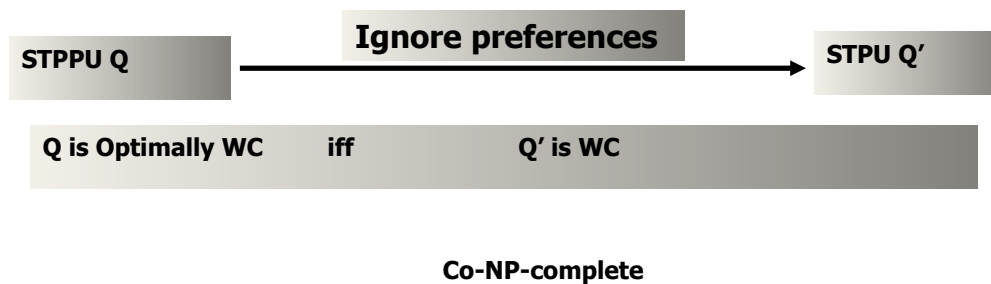
From the minimum preference up until inconsistency do:

1. **Cut** the STPPU P and get STPU Q
2. Check if Q is **Strongly Controllable**
3. **Merge** the results obtained at all preferences levels

Complexity: |preferences| x |variables|³ x |interval size|

Optimal Weak Controllability (OWC)

An **STP with Preferences and Uncertainty** is **Optimally WC** if there is an assignment to all the executable time points **consistent** and **optimal** with each of the possible scenarios.



Checking Optimal Dynamic Controllability

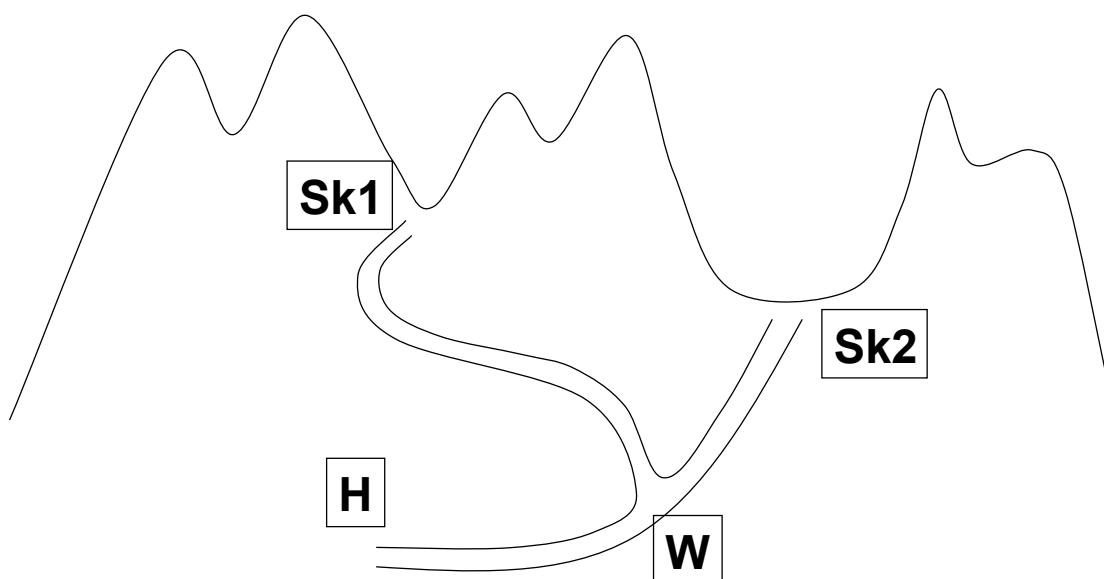
An STPPU is Optimally DC if there exists a means of extending any current partial control sequence to a complete control sequence in the future in such a way that the resulting schedule will be optimal.

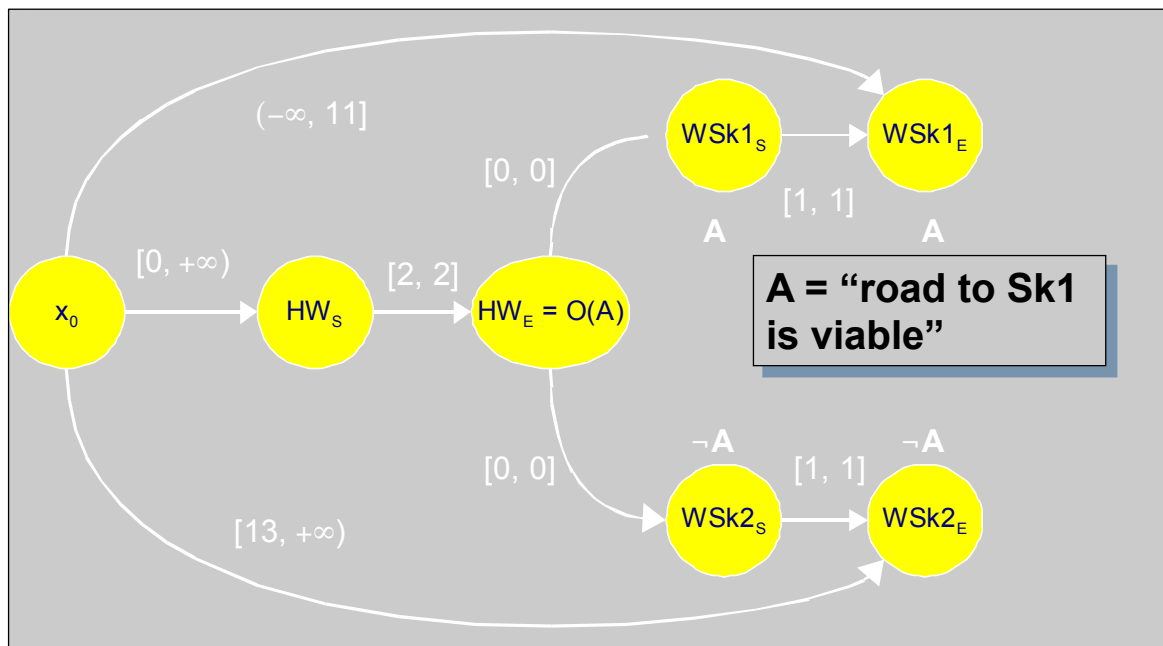
From the minimum preference up until inconsistency do:

1. For each preference level:
 1. Cut the STPPU P and get STPU Q
 2. Check if Q is **Dynamically Controllable**
 - if so, for each controllable variable we will know how long it has to wait before executing and ensuring a final preference of at least the cut level
 2. Merge results obtained at all preference levels
 - intersect intervals ensuring controllability
 - take the longest waiting time
- Complexity: |preferences| x DC Complexity

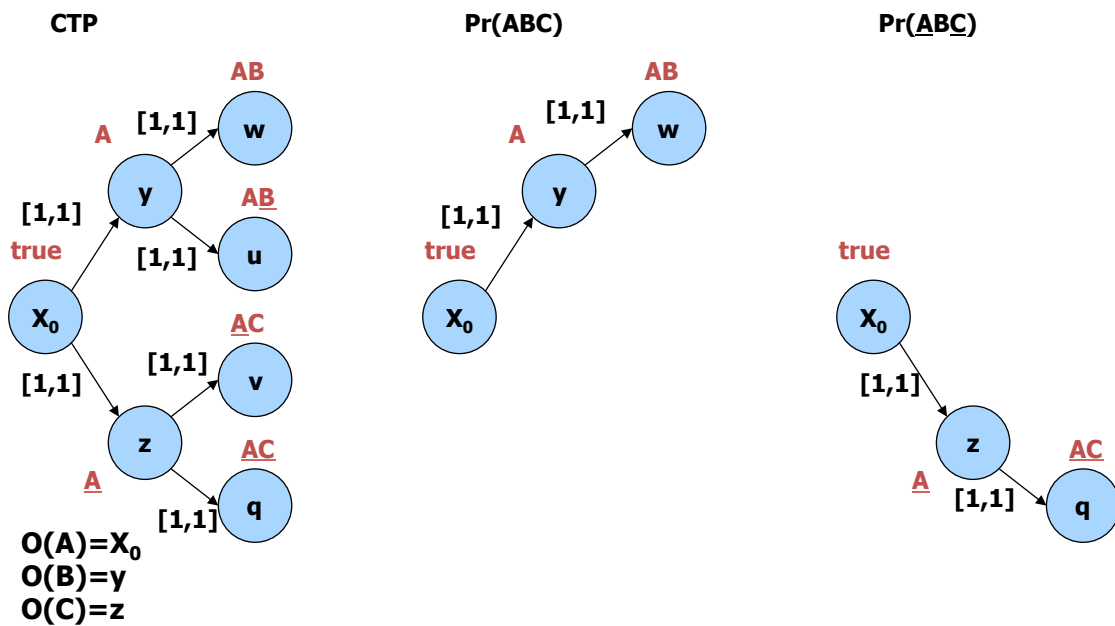
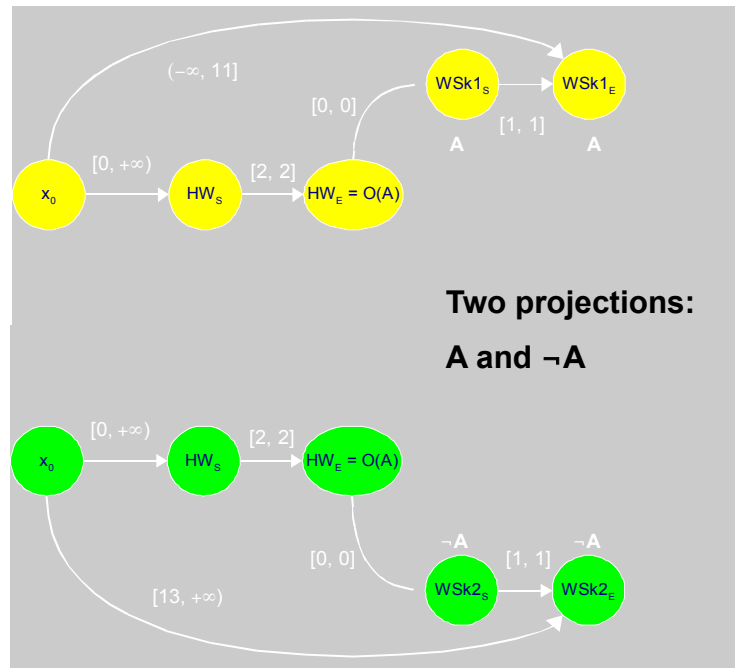
- Modeling uncertainty on whether some events will actually occur
- The main idea of CTPs is to attach to each variable, representing a time event, a label. The variable will be executed iff the label is true.
- **Label**: conjunction of literals. Ex: ABC (A and B and C)
- A CTP
 - Is a **CSTP** if the constraints in E are of STP type
 - Is a **CTCSP** if the constraints in E are of TCSP type
 - Is a **CDTP** if the constraints in E are of DTP type

Going skiing



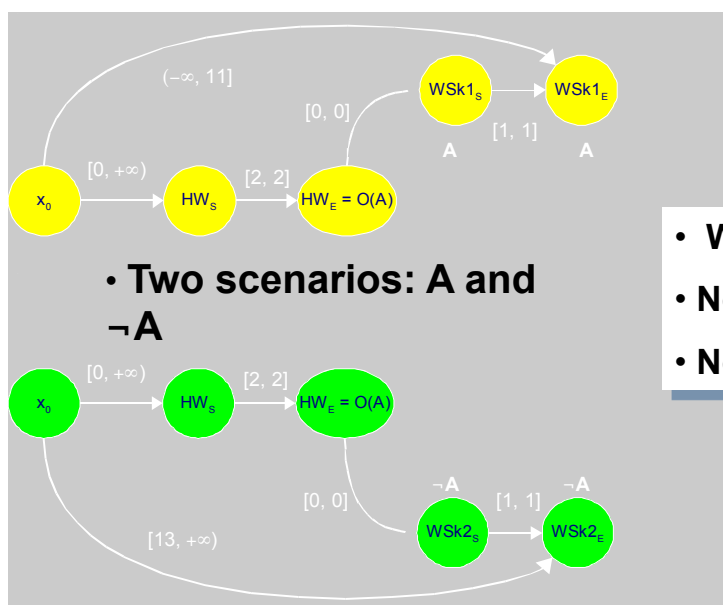


- **Execution scenario s**: label partitioning the variables in V into two sets (activated and non-activated)
- **Scenario projection** of CTP Q and scenario s , is the non conditional temporal problem (STP, TCSP, DTP) obtained considering only the activated variables and the constraints among them



- There are three notions of consistency :
 - **Strong Consistency (SC)** there is a fixed way to assign values to all the variables that satisfies all projections
 - Solving: Equivalent to the consistency of the problem containing all variables and constraints (complexity depends on the underlying problem)
 - **Weak Consistency (WC)** the projection of each scenario is consistent
 - Solving: Identify the set of minimal scenarios, then check the consistency of the corresponding projections (co-NP-complete)
 - **Dynamic Consistency (DC)** the current partial consistent assignment can be consistently extended independently of the upcoming observations.
 - Solving: specific property on pairs of projections (difficult, actual complexity unknown)

SC \rightarrow DC \rightarrow WC



• Two scenarios: A and $\neg A$

- Weakly Consistent
- Not Strongly Consistent
- Not Dynamically Consistent

- Labels, associated to variables, act as rules that select different execution paths

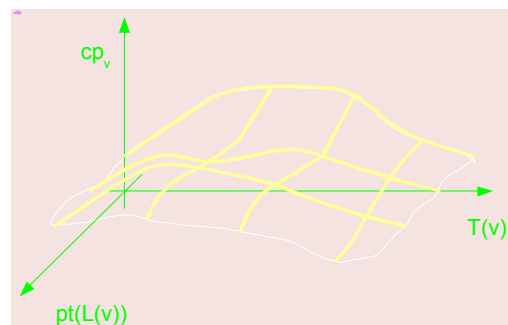
IF $L(v)$ **THEN** *EXECUTE* (v)

- Degrees can be added
 - to the premise ($pt: L(V) \rightarrow \mathcal{A}$): **t**ruth level
 - to the consequence ($cp : V \rightarrow \mathcal{A}$): **p**reference

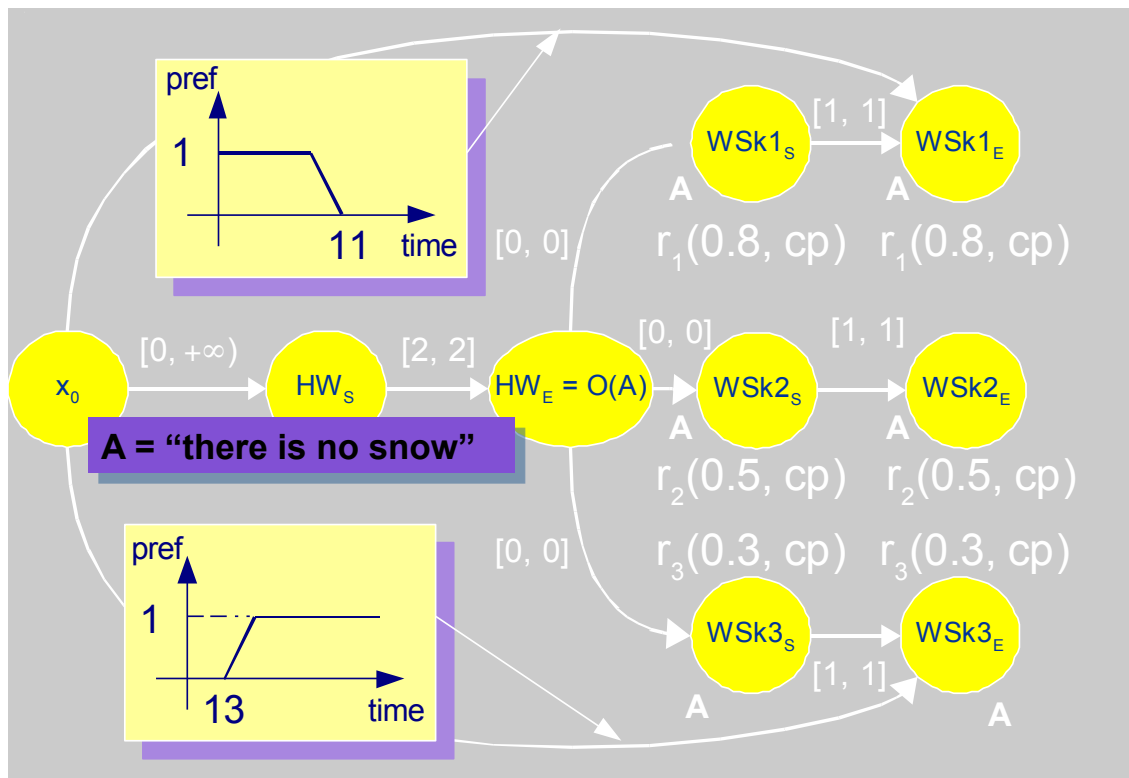
Fuzzy rules for CTPPs - definition

- IF** $pt(L(v), deg) > \alpha$ **THEN** *EXECUTE* (v) : $cp(pt(L(v), deg))$

also written $r(\alpha, cp)$



- A node v in V is executed with a preference given by cp if the truth degree of its premise given by pt , through the interpretation function deg , is greater than α



- There are three notions of consistency
 - *α -Strong Consistency (SC)*: there is a fixed way to assign values to all the variables that has preference at least α all projections
 - *α -Weak Consistency (WC)*: the projection of any scenario is consistent with optimal preference $\geq \alpha$
 - *α -Dynamic Consistency (DC)*: the current partial solution can be consistently extended independently of the upcoming observations to a solution with preference $\geq \alpha$

$$\alpha\text{-SC} \rightarrow \alpha\text{-DC} \rightarrow \alpha\text{-WC}$$

- ICAPS 2014:
 - *EfficientIDC: A Faster Incremental Dynamic Controllability Algorithm.*
Mikael Nilsson, Jonas Kvarnström and Patrick Doherty.
 - *Resolving Uncontrollable Conditional Temporal Problems using Continuous Relaxations.*
(Honorable Mention for the Outstanding Paper Award)
Peng Yu, Cheng Fang and Brian Williams.
 - *Time-dependent Simple Temporal Networks: Properties and Algorithms.*
Cedric Pralet and Gerard Verfaillie.
- TIME 2014
 - Romeo Rizzi, Roberto Posenato and Carlo Comin. A Tractable Generalization of Simple Temporal Networks and its relation to Mean Payoff Games
 - Mikael Nilsson, Jonas Kvarnström and Patrick Doherty. Incremental Dynamic Controllability in Cubic Worst-Case Time
 - Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato and Marco Roveri. Sound-and-Complete Algorithms for Checking the Dynamic Controllability of Temporal Networks with Uncertainty, Disjunction and Observation

Applications

- Planning, Scheduling and Execution Systems
 - Time is introduced in the process of transforming abstract goals into ordered collection of actions
 - Abstract --> Concrete representations
- Processing data
 - Time is introduced in the process of abstracting useful information from structured (timestamped) or unstructured (e.g. text) data.
 - Abstract ← Concrete representations

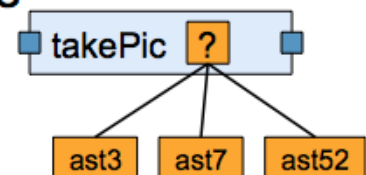
- A family of approaches based on:
 - Temporal networks
 - Timelines describing fluents over time
 - States and activity networks of variables
 - Constraint propagation
 - Planning as a dynamic constraint problem
 - Constraint problem changes over time
 - Variables, constraints, domains added or deleted.

- Representation
 - Represent activity parameters and temporal events
 - Represent constraints among parameters/events
- Reasoning
 - Identify when plan candidate is inconsistent
 - Eliminate choices not leading to valid plans
- Requirements
 - General: arbitrary constraints (domain-dependent)
 - Dynamic: constraints, variables and values added/deleted
 - Efficient: network changed and queried at each plan step
 - Trade-off between efficiency and completeness of reasoning

Constraint-based planning

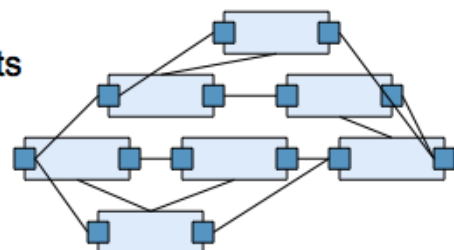
- Activities represented as intervals

- Each interval specifies activity
- Each interval has start and end
- Interval can have parameters



Candidate plan is a network of intervals

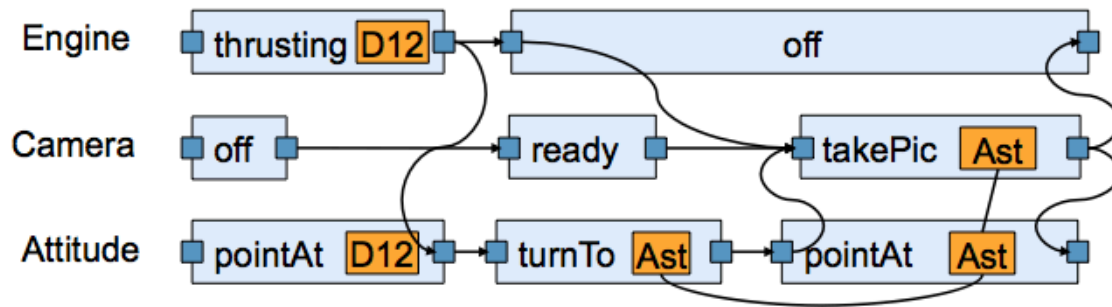
- Intervals linked by temporal constraints
- Interval parameters linked by constraints
- Gives rise to constraint network



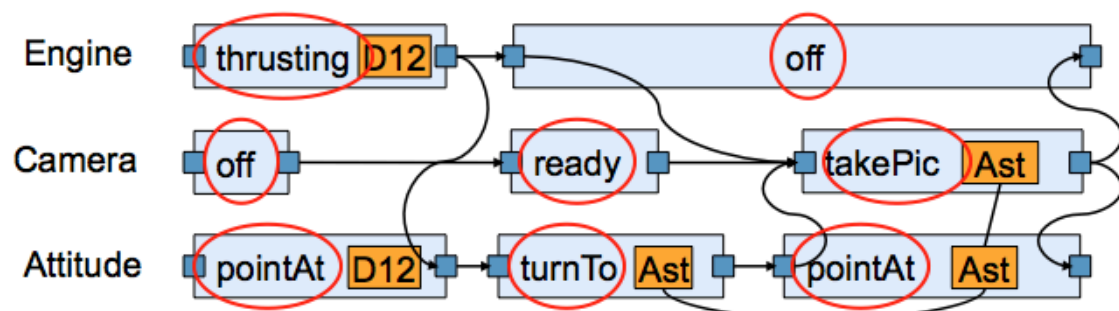
Feasibility of candidate plan

- If network is inconsistent, cannot become a valid plan

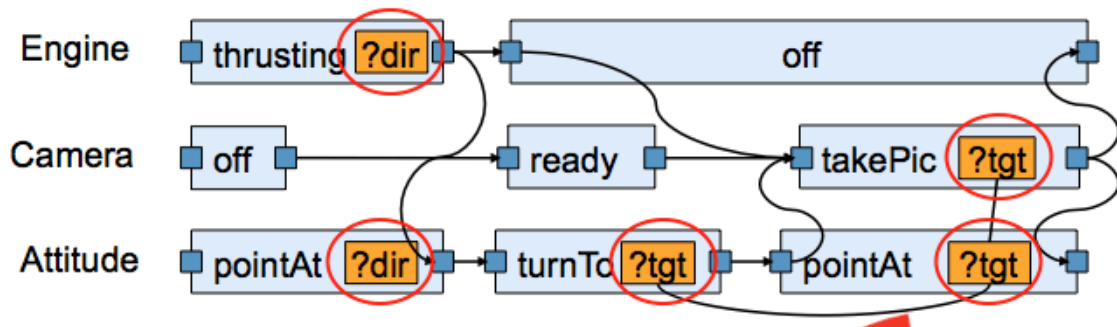
- Plan is a network of intervals representing activities



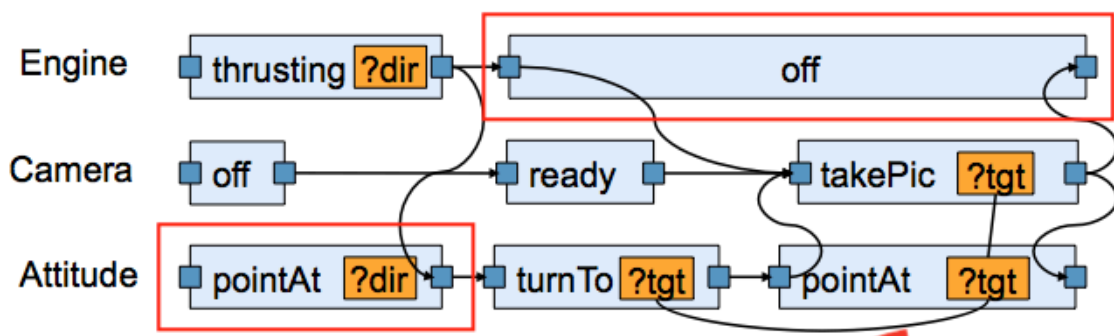
- Logical predicates describe actions and states



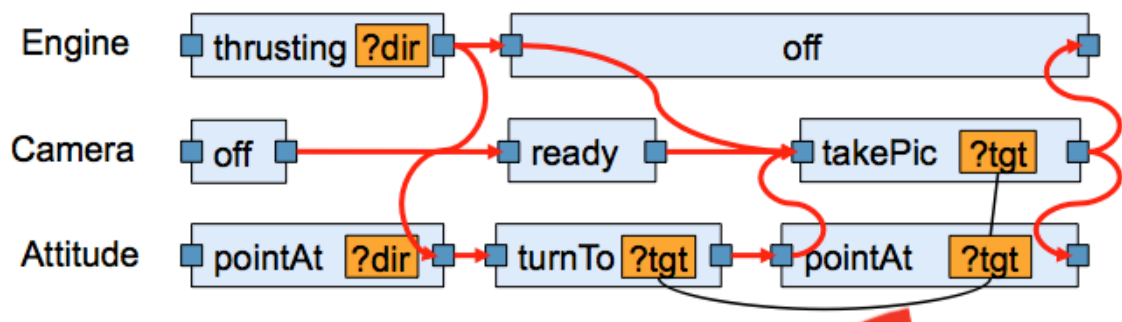
- Parameterized predicates
 - Each predicate type has a fixed set of parameters
 - Each parameter instance comes from associated domain
 - Parameters described by variables



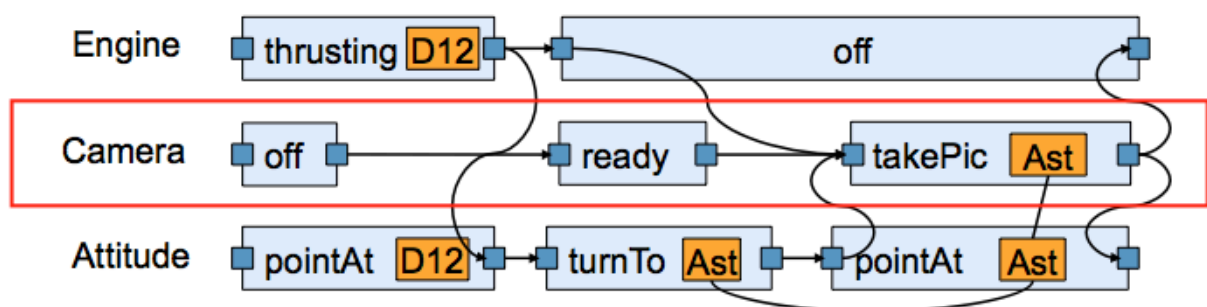
- Interval describes activity with duration
 - Start and end times ()
 - Predicate and parameter variables



- Temporal relations among intervals
 - Can be represented as constraints among start/end times



- Enforce that activities for same system do not conflict
 - Activities on same timelines are temporally ordered



- Fully automated planning usually not possible
- Humans allowed to
 - Schedule and unschedule activities
 - Edit plans: moving activities in time
- Automated planner maintains validity of plan
- Human-in-loop helps in understanding and accepting plan.

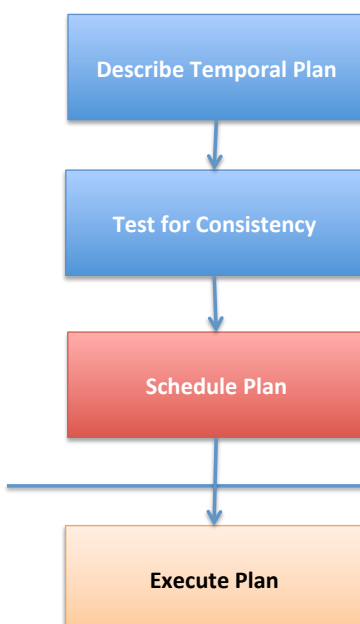
- Underlying representation of time is flexible.
- Interfaces for planning usually show single instantiation (timeline).
- How is that instantiation selected?
 - Earliest start time? (not always intuitive to human)
- Given a new goal, how should the planner update the plan?
 - Minimum perturbation

Temporal Reasoning in Execution Systems

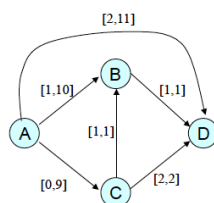
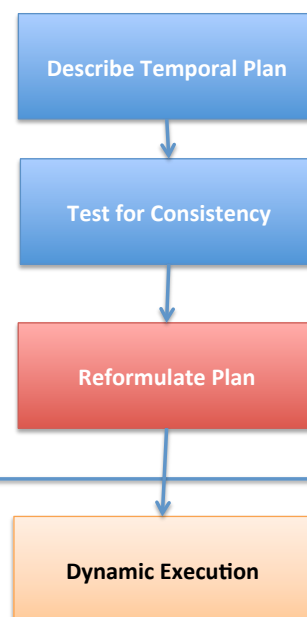
- Autonomous systems with a deliberative (planning) component combine planning with execution.
 - The subsystem responsible for carrying out a plan is called the *executive*.
 - When dispatching plans with flexible temporal constraints, there is a need for a separate *dispatcher*.
- Dispatcher notifies executive when an action **can** or **must** be executed.
 - Correctness: whatever executive does adheres to temporal constraints
 - Preserves flexibility: dispatcher never tells the executive that an action can't be performed at a certain time when it can.

Dispatching Temporal Plans

Option 1. Schedule Off Line



Option 2. Schedule On Line

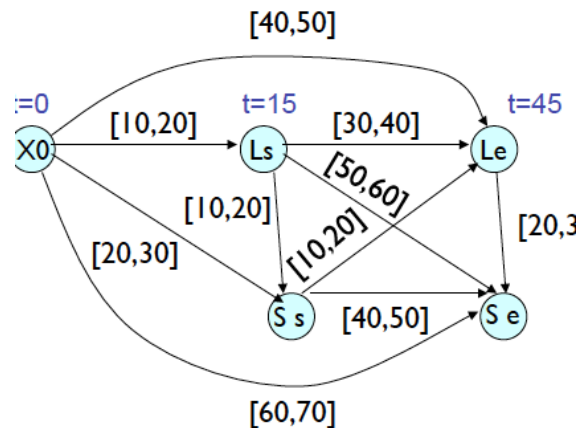
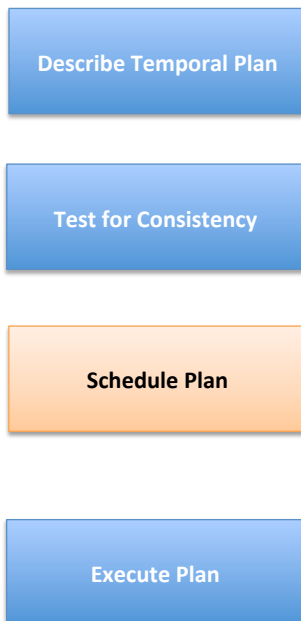


Off line

On line

Scheduling Temporal Plans

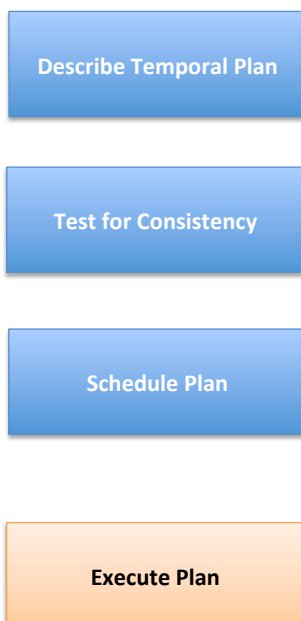
Option 1. Schedule Off Line



- Given an STN, a schedule is an assignment of times to all variables.
- To generate a schedule from a STN without search, first transform STN into a 'decomposable' STN, where all the implied constraints are revealed, using the all-pairs-shortest-path algorithm (Floyd-Warshall)
- Then incrementally assign times to variables (in any order) and propagate.

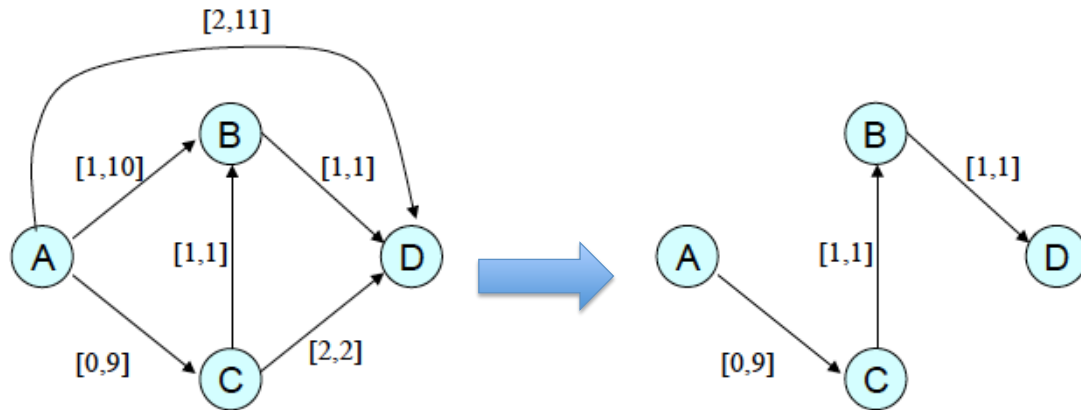
Executing Temporal Plans

Option 1. Schedule Off Line



- **Problem:** changes in task duration can cause plan failure if scheduling occurs off line.
 - Fixed schedule removes flexibility
- **Solution:** Execution adapts through dynamic scheduling.
 - Assign time to event at execution time.
 - Guarantee that all constraints will be satisfied.

Dispatching Flexible Temporal Plans



- **Problem:** execution latency while propagating effects of assigning times.
- **Solution:** generate schedule with low latency through removal of redundant edges.
- Reformulating plans minimizes latency.

Off line

Reformulate Plan

On line

Dynamic Execution

Dynamic Execution of Flexible STNs

Option 2. Schedule On Line

- Execute and event when it is *enabled* and *active*.
- **Enabled** : predecessors of event have been scheduled.
- **Active** : Current time is within bounds of event.
- **Algorithm:** when event is enabled and active, assign time and propagate effects to immediate successors.

Describe Temporal Plan

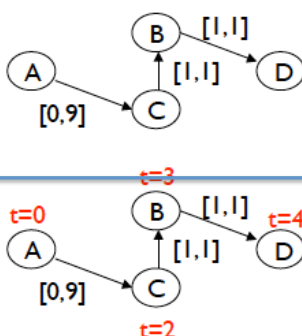
Test for Consistency

Reformulate Plan

Off line

On line

Dynamic Execution



- Dispatching STPs (Muscettola, Morris et al. 1998; Tsamardinos 1998; Tsamardinos, Morris et al. 1998; Wallace and Freuder 2000)
- Dispatching DTPs (Tsamardinos 2001)
- Executing a Reactive Model-based Programming Language (RMPL) plan using Temporal Plan Networks (Kim et al 2001)
 - TPNs are like conditional temporal networks.

Application two: Processing Temporal information in Medicine

- Temporal Database Management
 - Involves tasks of storing, processing and retrieving time-oriented information
- Temporal Abstraction
 - Relates to the task of creating interval-based concepts (abstractions) from time-stamped raw data.
- Temporal Data Visualization
 - Involves tasks of collecting, navigating and visualizing time-oriented information.
- Medical Natural Language Processing
 - Extracting temporal information from medical text

- Interval Algebra:
 - Applied to the tasks of temporal abstraction and query processing to determine qualitative temporal relationships between medical events (Shahar, 1997).
- Temporal Constraint Networks:
 - Used to facilitate patient-monitoring and problem-detection, manage medical resources, and determine consistency of temporal constraints in clinical guidelines. Also used to model temporal information in clinical discharge summaries (Zhou 2007)

Temporal Abstraction of Medical Data

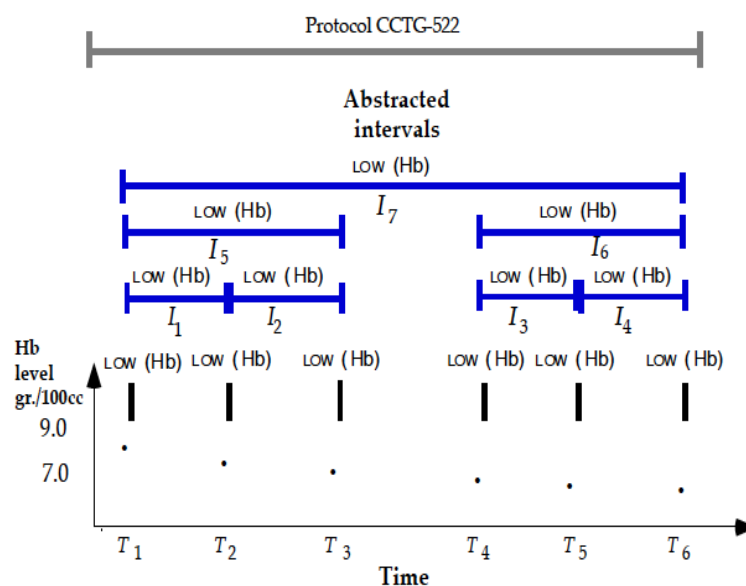


Figure 3: Processing of parameter points and intervals by the temporal-abstraction mechanisms. The (primitive) parameter points that hold at times T_1 and T_2 are abstracted into two abstraction points, over which a LOW(Hb) state abstraction is interpreted, by contemporaneous abstraction; these point abstractions are joined into a LOW(Hb) interval abstraction I_1 by temporal interpolation. Abstractions I_1 and I_2 are joined by temporal inference into the longer LOW(Hb) interval abstraction I_5 , as are I_3 and I_4 into I_6 . Interval abstractions I_5 and I_6 are joined into a LOW(Hb) interval abstraction I_7 by temporal interpolation. A DECREASING(Hb) gradient abstraction during interval I_7 can be

More References To Applications Discussed in this Section

- Executing Reactive, Model-based Programs Through Graph-based Temporal Planning” (Kim, et al) IJCAI 2001.
- “Temporal reasoning with medical data--a review with emphasis on medical natural language processing”(Zhou) Journal of Biomedical Information 2007
- “A framework for knowledge-based temporal abstraction.” (Shahar) AIJ 1997.
- “Flexible dispatch of disjunctive plans.” (Tsamardinos) Proceedings of the 6th European Conference on Planning 2001
- “Reformulating Temporal Plans For Efficient Execution” (Muscettola, Morris et al.) Principles of Knowledge Representation and Reasoning, 1998

Thanks

