Temporal Reasoning for Planning, Scheduling and Execution in Autonomous Agents

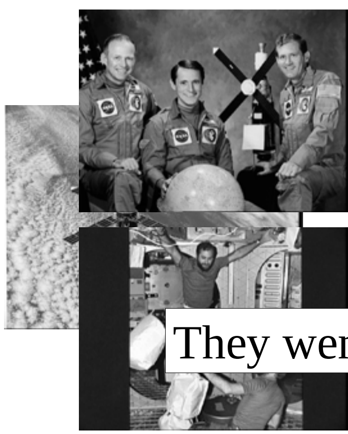
AAMAS-2005 Tutorial (T4)

Nicola Muscettola,
Ioannis Tsamardinos*,
and Luke Hunsberger

*Martha E. Pollack contributing to the slides

Real-World Agent Planning and Execution

Space Facility Crew Activity Planning



- Activity schedule very tight
- Did not adapt to uncertainties in execution
- Did not adapt to human needs for more flexibility
- 45 days into the mission they rebelled

They went on strike!

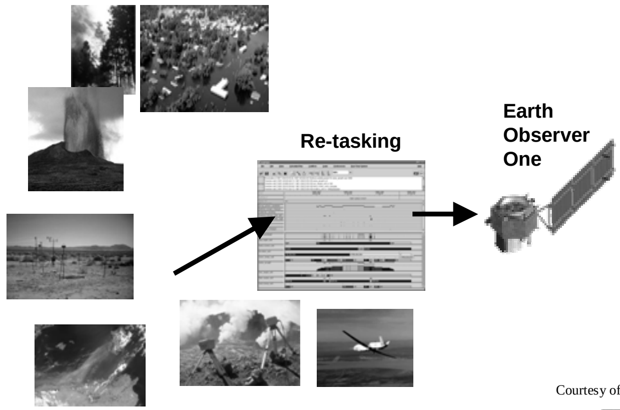
MAPGEN in Surface Operations

Surface Operations

- MAPGEN: First Artificial Intelligence (AI) based Decision-Support System to control a spacecraft on the surface of another planet
- Spirit:
  - Nominal science operations from Sol 15 to 18
    - All planned activities from 16/17 executed on board
  - Return to nominal science operations within 2-3 days
- Opportunity:
  - Informal use begins Sol 4/5
    - Commanded activities executed on board nominally
  - Nominal science operations tomorrow (Feb 6th)
- Dual rover support use of MAPGEN in full swing
  - Continues to be for MER Extended Ops
- Conservative ROI to NASA: 25% extra science returned per Sol, over a manual approach for plan synthesis
  - Approx $1.4 Million/Sol

(1 Sol = 1 Martian Day = 24hrs 37mins Earth time)

EO-1 Sensorweb



Courtesy of JPL

Triggers so far: Wildfires, Floods, Volcanoes (thermal, ash), Ice/Snow, in-situ sensors, modified by cloud cover

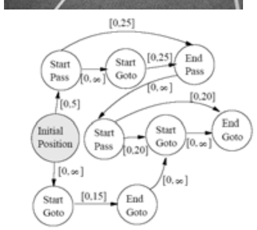
Robust Task Execution for Long Traverse Rovers

- ASTEP LITA Atacama Field Campaign (Sep-Oct 2004)
  - Zöe rover with life detecting instruments
  - On-board planning and autonomous navigation over long distances
- Rover executive results (preliminary, telemetry still being analyzed)
  - Total hours of operations (cumulative over several runs): 17 hours
  - Total distance covered: 16 km
  - Longest autonomous traverse: 3.3Km 2h 29m
  - "Roughest traverse": 1h 2m with 19 faults recovered
  - Faults addressed:
    - Navigator "confused"
    - Internal processes failed
    - Early and late arrival at waypoint

Autominder: Assistive Technology for Cognition

To assist people with memory impairment:
•Model their daily activities, including temporal constraints on their performance
•Monitor the execution of those activities
•Decide whether and when to issue reminders

Soccer!



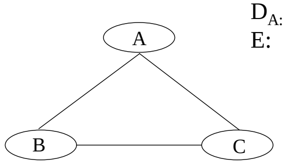
Issues in Temporal Planning and Execution

- Representation: What kinds of temporal information can we represent?
- Planning
  - Generation: How do we construct a temporal plan?
- Execution
  - Dispatch: When should the steps in the plan be executed? How do we maintain the state of the plan, given that time is passing (and events are occurring)?
- Focus Today: Constraint-Based Models

Constraint Satisfaction Problems

- <V,D,E>
  - $V = \{v_1, v_2, \ldots v_n\}$: set of constrained variables
  - $D = \{D_1, D_2, \ldots, D_n\}$: domains for each variable
  - E = relations on a subset of V: constraints, representing the legal (partial) solutions

1-Minute Review of CSPs



V: {A,B,C}

$D_{A:}$ {R, B} D_B: {$\cancel{R}$, B} D_C : {$\cancel{R}$, Y}

E: E_{AB} = {<R,B>, <B,R>}
E_{AC} = {<R, Y>, <B,R>,<B,Y>}
E_{BC} = {<R, Y>, <B,R>,<B,Y>}

•Solve with a combination of search and propagation (forward checking, arc consistency, etc.)

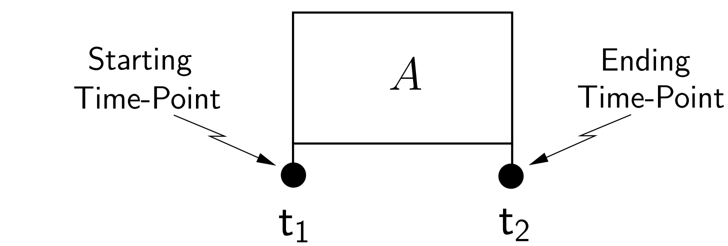
•Relations here are binary—may have higher arity as well

High Level Outline

1. Time representations in problem solving and execution
2. Planning with time (plan generation and multi-agent collaborative planning)
3. Resource reasoning

Simple Temporal Networks

Temporal Constraints on an Action

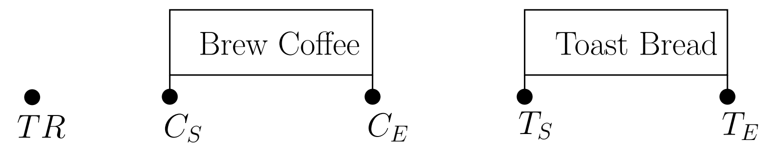


$$t_1 \geq 4 \qquad (A \text{ starts at or after } 4)$$

$$t_2 \leq 12 \qquad (A \text{ ends at or before } 12)$$

$$3 \leq t_2 - t_1 \leq 6 \qquad (A\text{'s dur. between 3 and 6})$$

Temporal Constraints on Breakfast



Goal: Prepare coffee and toast.
Have them ready within 2 minutes of each other.
Brew coffee for 3–5 minutes;
Toast bread for 2–4 minutes.

Temporal Constraints on Airline Travel

Goal: Fly from Boston to Seattle:

- Leave Boston after 4 p.m. on Aug. 8;
- Return to Boston before 10 p.m., Aug. 18;
- Away from Boston no more than 7 days;
- In Seattle at least 5 days; and
- Return flight lasts no more than 7 hours.

Simple Temporal Network (STN)*

A Simple Temporal Network (STN) is a pair, $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, where:

- $\mathcal{T}$ is a set of time-point variables: $\{t_0, t_1, \ldots, t_{n-1}\}$ and
- $\mathcal{C}$ is a set of binary constraints, each of the form: $t_j - t_i \leq \delta$, where δ is a real number.

* (Dechter, Meiri, & Pearl 1991)

Solutions, Consistency, Equivalence

- A *solution* to an STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ is a complete set of variable assignments:

  $\{t_0 = w_0,\ t_1 = w_1,\ \ldots,\ t_{n-1} = w_{n-1}\}$

  that satisfies all the constraints in $\mathcal{C}$.
- An STN with at least one solution is called *consistent.*
- STNs with identical solution sets are called *equivalent*.

The *Zero* Time-Point Variable

- Frequently, it is useful to fix one of the time-point variables to 0. That "variable" will often be called z.
- Binary constraints involving z are equivalent to unary constraints:

$$t_j - z \leq 5 \iff t_j \leq 5$$

$$z - t_i \leq -3 \iff t_i \geq 3$$

STN for Constrained Action

$$\mathcal{T} = \{z, t_1, t_2\}, \text{ where: } \begin{aligned} z &= 0 \\ t_1 &= \text{Start of } A \\ t_2 &= \text{End of } A \end{aligned}$$

$$\mathcal{C} = \left(\begin{array}{rcrl} t_2 - t_1 & \leq & 6 & (\text{Dur. less than } 6) \\ t_1 - t_2 & \leq & -3 & (\text{Dur. greater than } 3) \\ z - t_1 & \leq & -4 & (\text{A starts after } 4) \\ t_2 - z & \leq & 12 & (\text{A ends before } 12) \end{array} \right)$$

STN for Breakfast

$\mathcal{T} \;=\; \{\mathsf{T_R}, \mathsf{C_S}, \mathsf{C_E}, \mathsf{T_S}, \mathsf{T_E}\}$, where:

$$\begin{array}{rcl} \mathsf{T_R} & = & 0 \quad \text{(Reference Time-point)} \\ \mathsf{C_S}/\mathsf{C_E} & = & \text{Start/End of Coffee Brewing} \\ \mathsf{T_S}/\mathsf{T_E} & = & \text{Start/End of Bread Toasting} \end{array}$$

$$\mathcal{C} \;=\; \left(\begin{array}{rcrrcr} \mathsf{C_E} - \mathsf{C_S} & \leq & 5, & \mathsf{C_S} - \mathsf{C_E} & \leq & -3 \\ \mathsf{T_E} - \mathsf{T_S} & \leq & 4, & \mathsf{T_S} - \mathsf{T_E} & \leq & -2 \\ \mathsf{C_E} - \mathsf{T_E} & \leq & 2, & \mathsf{T_E} - \mathsf{C_E} & \leq & 2 \\ \mathsf{T_R} - \mathsf{C_S} & \leq & 0, & \mathsf{T_R} - \mathsf{T_S} & \leq & 0 \end{array}\right)$$

STN for Constrained Air Travel

$\mathcal{T} \;=\; \{z, t_1, t_2, t_3, t_4\}$, where $z =$ Noon, Aug. 8.

$\mathcal{C} =$

$$\left\{\begin{array}{rcrl} z - t_1 & \leq & -4 & \text{(Lv Bos after 4 p.m., 8/8)} \\ t_4 - z & \leq & 250 & \text{(Av Bos by 10 p.m., 8/18)} \\ t_4 - t_1 & \leq & 168 & \text{(Gone no more than 7 days)} \\ t_2 - t_3 & \leq & -120 & \text{(In Seattle at least 5 days)} \\ t_4 - t_3 & \leq & 7 & \text{(Return flight less than 7 hrs)} \end{array}\right\}$$

Graphical Representation of an STN*

The *Distance Graph* for an STN, $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, is a graph, $\mathcal{G} = (\mathcal{T}, \mathcal{E})$, where:

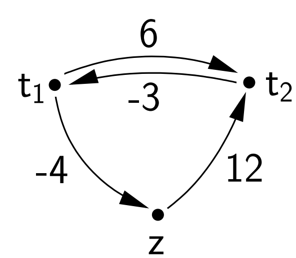
- Time-points in $\mathcal{S}$ correspond to nodes in $\mathcal{G}$.
- Constraints in $\mathcal{C}$ correspond to edges in $\mathcal{E}$:

$$t_j - t_i \leq \delta \quad \longleftrightarrow \quad t_i \xrightarrow{\;\delta\;} t_j$$

* (Dechter, Meiri, & Pearl 1991)

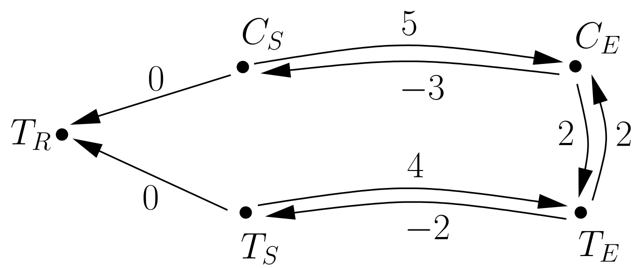
Distance Graph for Action Scenario

$$\mathcal{T} \;=\; \{z, t_1, t_2\} \qquad \mathcal{C} \;=\; \left\{\begin{array}{rcr} t_2 - t_1 & \leq & 6 \\ t_1 - t_2 & \leq & -3 \\ z - t_1 & \leq & -4 \\ t_2 - z & \leq & 12 \end{array}\right\}$$



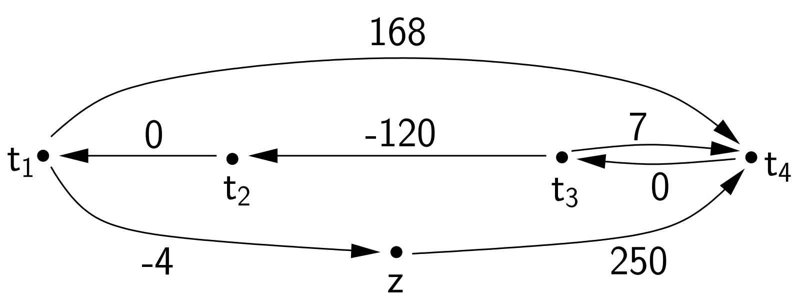
Distance Graph for Breakfast

$$\left\{\begin{array}{lll} C_E - C_S \le 5, & C_S - C_E \le -3 \\ T_E - T_S \le 4, & T_S - T_E \le -2 \\ C_E - T_E \le 2, & T_E - C_E \le 2 \\ T_R - C_S \le 0, & T_R - T_S \le 0 \end{array}\right\}$$



Distance Graph for Airline Scenario

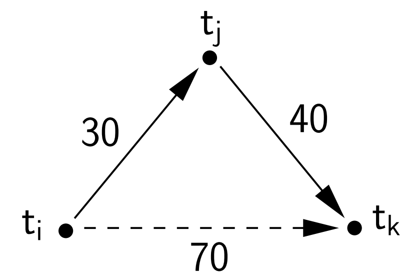
$$\left\{\begin{array}{ll} z - t_1 \le -4, & t_4 - z \le 250 \\ t_4 - t_1 \le 168, & t_2 - t_3 \le -120 \\ t_4 - t_3 \le 7, & t_1 - t_2 \le 0 \\ t_3 - t_4 \le 0 & \end{array}\right\}$$



Implicit Constraints

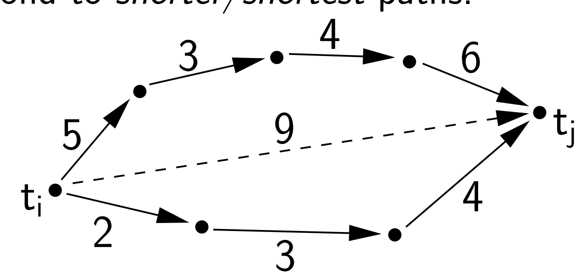
Explicit constraints in $\mathcal{C}$ can combine to form implicit constraints:

$$\begin{array}{l} t_j - t_i \le 30 \\ t_k - t_j \le 40 \\ \hline t_k - t_i \le 70 \end{array}$$



Implicit Constraints as Paths

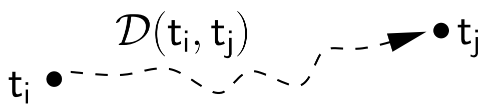
- Chains of implicit constraints in an STN correspond to *paths* in its Distance Graph.
- *Stronger/strongest* implicit constraints correspond to *shorter/shortest* paths.



Distance Matrix *

The *Distance Matrix* for an STN, $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, is a matrix $\mathcal{D}$ defined by:

$\mathcal{D}(t_i, t_j) =$ Length of Shortest Path from t_i to t_j in the Distance Graph for $\mathcal{S}$

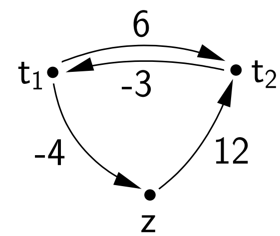


(Dechter, Meiri, & Pearl 1991)

Distance Matrix (cont'd.)

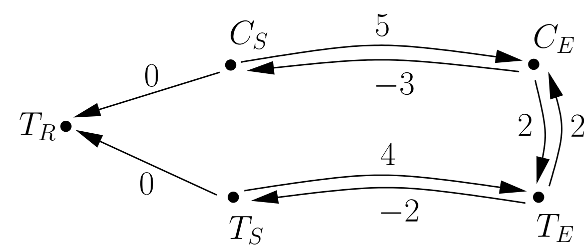
- The strongest implicit constraint on t_i and t_j in $\mathcal{S}$ is: $t_j - t_i \leq \mathcal{D}(t_i, t_j)$
- Abuse of notation: $\mathcal{D}(i, j)$ instead of $\mathcal{D}(t_i, t_j)$
- $\mathcal{D}$ is the *All-Pairs, Shortest-Path* Matrix for the Distance Graph (Cormen, Leiserson, & Rivest 1990).

Distance Matrix for Action Scenario



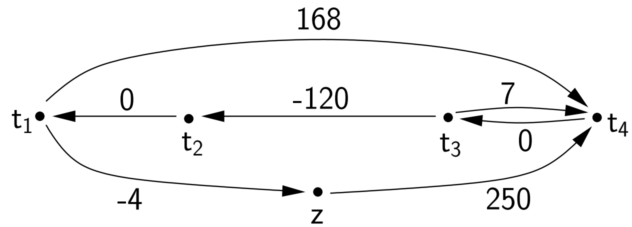
$\mathcal{D}$	z	t_1	t_2
z	0	9	12
t_1	-4	0	6
t_2	-7	-3	0

Distance Matrix for Breakfast



$\mathcal{D}$	T_R	C_S	C_E	T_S	T_E
T_R	0	∞	∞	∞	∞
C_S	0	0	5	5	7
C_E	-3	-3	0	0	2
T_S	0	3	6	0	4
T_E	-2	-1	2	-2	0

Distance Matrix for Airline Scenario



$\mathcal{D}$	z	t_1	t_2	t_3	t_4
z	0	130	130	250	250
t_1	-4	0	48	168	168
t_2	-4	0	0	168	168
t_3	-124	-120	-120	0	7
t_4	-124	-120	-120	0	0

Checking Consistency of an STN

Given an STN $\mathcal{S}$ with Distance Graph $\mathcal{G}$ and Distance Matrix $\mathcal{D}$, the following are equivalent (Dechter, Meiri, & Pearl 1991):

- $\mathcal{S}$ is consistent.
- Each loop in $\mathcal{G}$ has path length ≥ 0.
- The main diagonal of $\mathcal{D}$ contains only 0s.

Computing $\mathcal{D}$ from Scratch

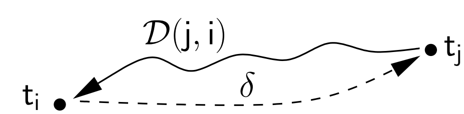
Polynomial algorithms for computing the All-Pairs, Shortest-Path Matrix (Cormen, Leiserson, & Rivest 1990):

- Floyd-Warshall Algorithm: $\mathcal{O}(n^3)$
- Johnson's Algorithm: $\mathcal{O}(n^2 \log n + nm)$

Adding Constraint to Consistent STN

- Given: $\mathcal{S} = (\mathcal{T}, \mathcal{C})$, a consistent STN.
- Adding the new constraint, $t_j - t_i \leq \delta$, to $\mathcal{S}$ will maintain the consistency of $\mathcal{S}$ iff:

$$-\mathcal{D}(j,i) \;\leq\; \delta \quad (\text{i.e., } 0 \;\leq\; \mathcal{D}(j,i) + \delta).$$

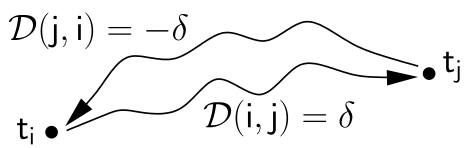


Note: This result is stated in different forms by many authors (Dechter, Meiri, & Pearl 1991; Demetrescu & Italiano 2002; Tsamardinos & Pollack 2003; Hunsberger 2003; Rohnert 1985).

Rigidly Connected Time-Points

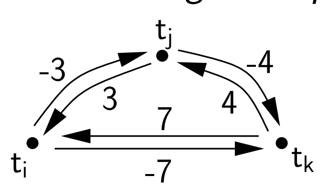
For consistent STNs, the following are equivalent:

- $(t_j - t_i) \;=\; \delta$, for some δ.
- $\mathcal{D}(i,j) + \mathcal{D}(j,i) \;=\; 0$
- t_i and t_j belong to a loop of path-length 0.



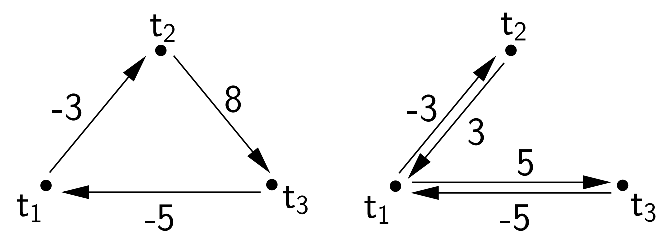
Rigidly Connected Time-Points (ctd.)

- t_i and t_j are said to be *rigidly connected* if $\mathcal{D}(i,j) = -\mathcal{D}(j,i)$.
- A set of time-points that are pairwise rigidly connected form a *rigid component*.



Note: Many authors consider rigidly connected time-points and rigid components (Tsamardinos, Muscettola, & Morris 1998; Gerevini, Perini, & Ricci 1996; Wetprasit & Sattar 1998).

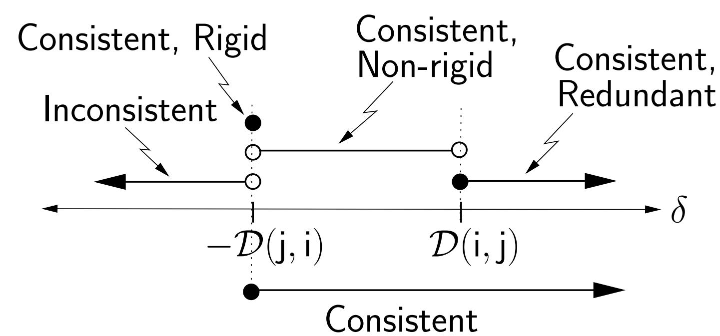
Examples of Rigid Components



Cyclical representation requires the fewest edges.

Adding Constraints to Consistent STNs

Result of adding the constraint, $t_j - t_i \leq \delta$:



Rohnert (1985) distinguishes most of these cases.

Finding a Solution to an STN*

While some time-points in are *not* rigid with z,

Pick some t_i not rigidly connected to z.

Pick some $\delta \in [-\mathcal{D}(t_i, z),\ \mathcal{D}(z, t_i)]$.

Add the constraint, $t_i = \delta$

(i.e., $t_i - z \leq \delta$ and $z - t_i \leq -\delta$).

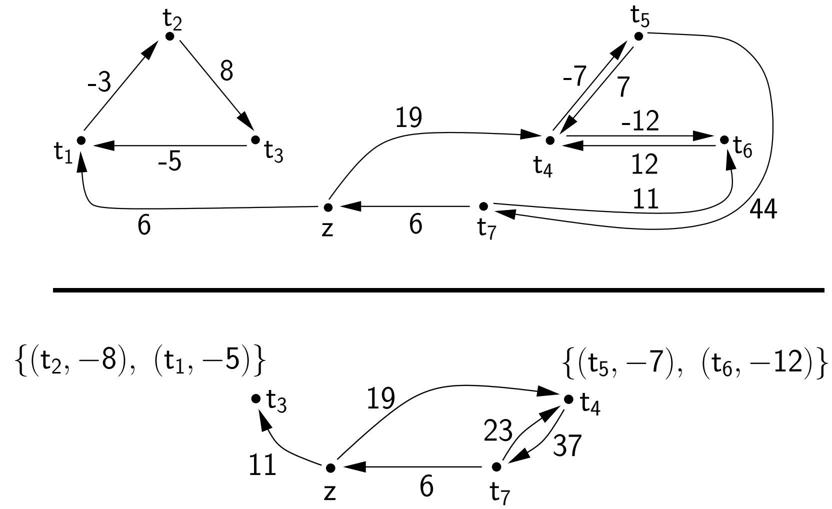
* This algorithm derives from Dechter et al. (1991).

Collapsing Rigid Components

- Select one time-point from each rigid component to serve as its *representative*
- Re-orient edges involving non-representative members of rigid components
- Associate additional information with each representative sufficient to enable reconstruction of its rigid component

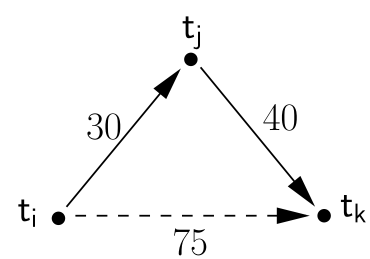
(Tsamardinos, Muscettola, & Morris 1998; Gerevini, Perini, & Ricci 1996; Wetprasit & Sattar 1998).

Collapsing Rigid Components: Example



Dominated Constraints

An explicit constraint, c: $t_j - t_i \leq \delta$, in an STN $\mathcal{S}$ is said to be *dominated* in $\mathcal{S}$ if *removing* c from $\mathcal{S}$ would result in no change to the distance matrix $\mathcal{D}$.

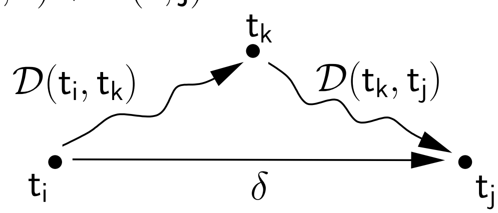


Note: Tsamardinos (1998) defines a different notion of dominance.

Dominated Constraints (cont'd.)

If $\mathcal{S}$ is consistent and has *no* rigid components then:

- If $\mathcal{D}(\mathrm{i},\mathrm{j}) < \delta$, then c is dominated in $\mathcal{S}$.
- If $\mathcal{D}(\mathrm{i},\mathrm{j}) = \delta$, then c is dominated in $\mathcal{S}$ iff there is some time-point $t_k \in \mathcal{T}$ such that: $\delta = \mathcal{D}(\mathrm{i},\mathrm{k}) + \mathcal{D}(\mathrm{k},\mathrm{j})$.

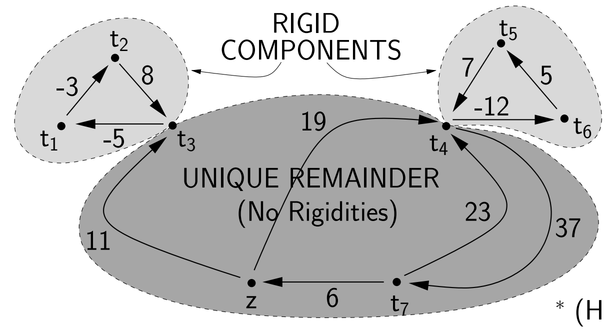


Undominated Constraints

If $\mathcal{S}$ has no rigid components, then the set of undominated constraints in $\mathcal{S}$ is uniquely defined and represents the fewest constraints in any STN equivalent to $\mathcal{S}$. (Hunsberger 2002b)

Canonical Form of an STN *

- Convert rigid components to cyclical form.
- Remove all *dominated* edges from the (unique) non-rigid remainder of the STN.



* (Hunsberger 2002b)

Incremental Algs for Distance Matrix

Computing Dist. Matrix Incrementally

- Incremental algorithms compute changes resulting from adding a single constraint.
- A naïve incremental algorithm can compute such changes in $\mathcal{O}(n^2)$ time.
- Better incremental algorithms based on constraint propagation—still $\mathcal{O}(n^2)$.

Adding a Constraint to Consistent STN

Given: New constraint c: $\mathsf{t_j} - \mathsf{t_i} \leq \delta$.

- Case 1: $\delta \; < \; -\mathcal{D}(\mathsf{j},\mathsf{i})$. — Inconsistent!
- Case 2: $\delta \; \geq \; \mathcal{D}(\mathsf{i},\mathsf{j})$. — Redundant!
- Case 3: $\delta \; \in \; [-\mathcal{D}(\mathsf{j},\mathsf{i}),\; \mathcal{D}(\mathsf{i},\mathsf{j})\;)$.

  — Adding c would require updating $\mathcal{D}$.

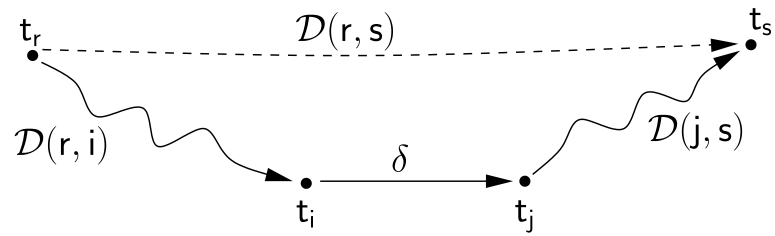
$\Rightarrow$ Incremental algorithms focus on Case 3.

Naïve Incremental Algorithm

For each entry, $\mathcal{D}(\mathsf{r},\mathsf{s})$,

If $\mathcal{D}(\mathsf{r},\mathsf{i}) + \delta + \mathcal{D}(\mathsf{j},\mathsf{s}) \; < \; \mathcal{D}(\mathsf{r},\mathsf{s})$, then set

$$\mathcal{D}(\mathsf{r},\mathsf{s}) \; = \; \mathcal{D}(\mathsf{r},\mathsf{i}) + \delta + \mathcal{D}(\mathsf{j},\mathsf{s}).$$



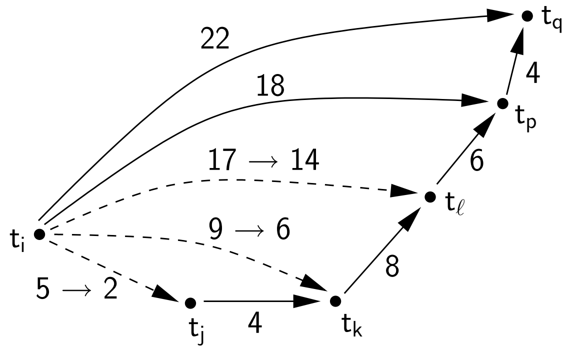
.

Constraint Propagation Algorithm*

- Propagate updates to $\mathcal{D}$ along edges in graph.
- Only propagate along *tight* edges.
  (Note: $\mathsf{t_s} - \mathsf{t_r} \leq \delta$ is tight iff $\mathcal{D}(\mathsf{r},\mathsf{s}) = \delta$.)
- Phase I: prop. forward; Phase II: prop. bkwd.
- Checks no more than $\mathsf{k}{*}\Delta$ cells of $\mathcal{D}$, where:
  $\Delta =$ number of cells needing updating; and
  $\mathsf{k} =$ max num edges incident on any node.

* This algorithm is based on the work of several authors (Rohnert 1985; Even & Gazit 1985; Ramalingam & Reps 1996).

Propagating Forward

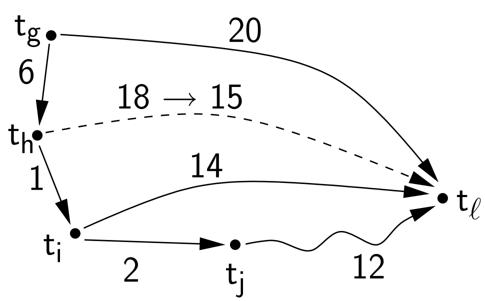


Adding the edge, $t_j - t_i \leq 2$, requires updating

$\mathcal{D}(i,j)$, $\mathcal{D}(i,k)$ and $\mathcal{D}(i,\ell)$, but not $\mathcal{D}(i,p)$.

Propagating Backward

For each t_ℓ such that $\mathcal{D}(i,\ell)$ changed during Forward Propagation, propagate backward from t_i:



Here, $\mathcal{D}(h,\ell)$ needs updating, but not $\mathcal{D}(g,\ell)$.

Improvements to Incremental Alg.

- Maintain canonical form of STN.
- Only update $\mathcal{D}$ for non-rigid portion of STN.
- Propagate only along *undominated* edges.
- Case 3.1: $\delta > -\mathcal{D}(j,i)$. (No new rigidities)
- Case 3.2: $\delta = -\mathcal{D}(j,i)$. (New rigidity(ies))

The Gory Details – Case 3.1

Inputs to Prop$_{3.1}$:

$\mathcal{S} = (\mathcal{T}, \mathcal{C}^u)$, an STN with only *undominated* constraints.

$\mathcal{D}$, the distance matrix for $\mathcal{S}$ (an array).

For each $t_r \in \mathcal{T}$, $\text{Succs}(t_r) = \{(t_s - t_r \leq \delta_{rs}) \in \mathcal{C}^u\}$ (a hash-table).

For each $t_r \in \mathcal{T}$, $\text{Precs}(t_r) = \{(t_r - t_q \leq \delta_{qr}) \in \mathcal{C}^u\}$ (a hash-table).

AffectedTPs, an empty hash-table.

EncounteredTPs, an empty hash-table.

$(t_j - t_i \leq \delta)$, a new constraint where: $-\mathcal{D}(j,i) < \delta < \mathcal{D}(i,j)$.

Note: This algorithm most closely resembles that of Ramalingam and Reps (1996).

The Gory Details – Case 3.1 (cont'd.)

$\mathbf{Prop}_{3.1}()$

Set: $\mathcal{D}(i,j) = \delta$.

Insert t_j into *AffectedTPs*.

PropFwd(t_j), which adds time-points to *AffectedTPs*.

For each $t_v \in$ *AffectedTPs*,

Clear *EncounteredTPs* hash-table.

PropBkwd(t_i, t_v).

The Gory Details — Case 3.1 (cont'd.)

PropFwd(t_y), where a path from t_i to t_y has already been processed and $\mathcal{D}(i,y)$ has been updated to the value $\delta + \mathcal{D}(j,y)$.

For each $t_z \in$ Succs(t_y),

If $t_z \notin$ *EncounteredTPs*,

Insert t_z into *EncounteredTPs*

If $\mathcal{D}(j,y) + \delta_{yz} = \mathcal{D}(j,z)$,

If $\delta + \mathcal{D}(j,y) + \delta_{yz} \leq \mathcal{D}(i,z)$,

Remove t_z from Succs(t_i) (if in there)

Remove t_i from Precs(t_z) (if in there)

If $\delta + \mathcal{D}(j,y) + \delta_{yz} < \mathcal{D}(i,z)$,

Set: $\mathcal{D}(i,z) = \delta + \mathcal{D}(j,y) + \delta_{yz}$

Insert t_z into *AffectedTPs*

PropFwd(t_z).

The Gory Details — Case 3.1 (cont'd.)

PropBkwd(t_s, t_v), where a path from t_s to t_v has already been processed and $\mathcal{D}(s,v)$ has been updated to the value $\mathcal{D}(s,i) + \delta + \mathcal{D}(j,v)$.

For each $t_r \in$ Precs(t_s),

If $t_r \notin$ *EncounteredTPs*,

Insert t_r into *EncounteredTPs*

If $\delta_{rs} + \mathcal{D}(s,i) = \mathcal{D}(r,i)$,

If $\delta_{rs} + \mathcal{D}(s,i) + \mathcal{D}(i,v) \leq \mathcal{D}(r,v)$,

Remove t_r from Precs(t_v) (if in there)

Remove t_v from Succs(t_r) (if in there)

If $\delta_{rs} + \mathcal{D}(s,i) + \mathcal{D}(i,v) < \mathcal{D}(r,v)$,

Set: $\mathcal{D}(r,v) = \delta_{rs} + \mathcal{D}(s,i) + \mathcal{D}(i,v)$

PropBkwd(t_r, t_v)

Case 3.2: Creating New Rigidity

Adding constraint, $t_j - t_i \leq -\mathcal{D}(j,i)$.

- Determine newly rigid time-points.
- Collapse new rigid component down to two points, using t_i as rep. for incoming edges and t_j as rep. for outgoing edges.
- Update set $\mathcal{C}^u$ of undominated constraints.
- Run $\text{Prop}_{3.1}$ algorithm.
- Collapse t_i and t_j into a single point.

Further Reading

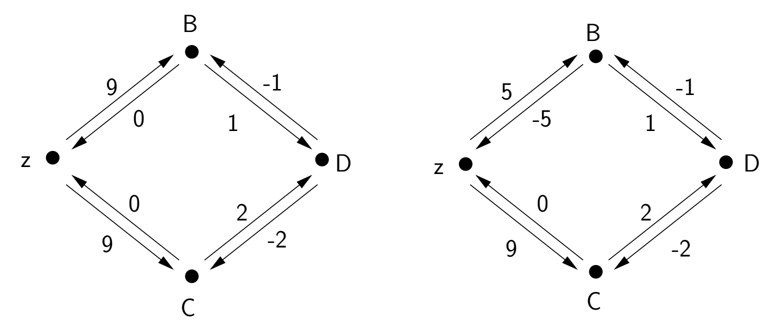
- Demetrescu and Italiano (2001; 2002) consider special cases where each edge can assume a bounded number of values; or where all edge weights are non-negative.
- Ramalingham and Reps (1996) introduce *incremental complexity analysis*.
- Zaroliagis (2002) discusses incremental and *decremental* algorithms.

Real-time Issues

Executing a Temporal Network

- To *execute* a time-point means to assign that time-point to the current moment.
- Goal: Maintain consistency of network while executing its time-points.
- Challenges:
  Decisions must be made in real time.
  Updating $\mathcal{D}$ takes time.

A Sample Execution*



After executing B at time 5, C must be executed at time 4 (which is already past).

* (Muscettola, Morris, & Tsamardinos 1998)

Greedy Dispatcher*

While some time-points not yet executed:

Wait until some time-point is executable.

If more than one, pick one to execute.

Propagate updates only to *neighboring* time-points (i.e., do no fully update $\mathcal{D}$).

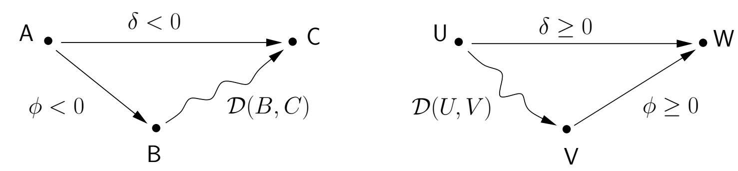
* (Muscettola, Morris, & Tsamardinos 1998)

Dispatchability*

- An STN that is guaranteed to be satisfied by Greedy Dispatcher is called *dispatchable*.
- Any *consistent* STN can be transformed into an equivalent *dispatchable* STN.
- Step I: The corresponding All-Pairs graph is equivalent and dispatchable.
- Step II: Remove *lower/upper-dominated* edges (does not affect dispatchability).

* (Muscettola, Morris, & Tsamardinos 1998)

Lower and Upper Dominance*

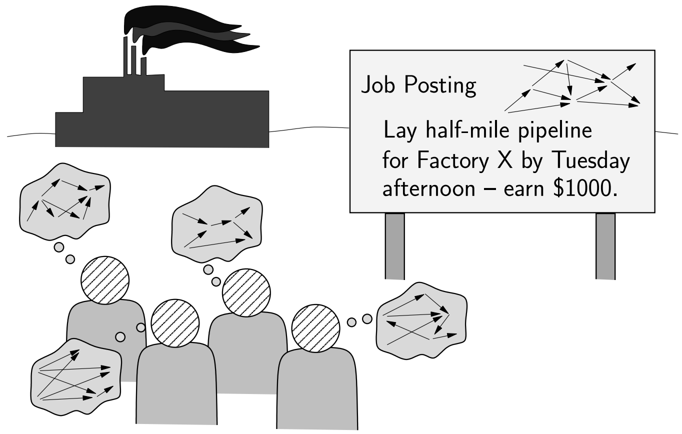


- The *negative edge* AC is *lower-dominated* if: $\delta = \phi + \mathcal{D}(B,C)$.
- The *non-negative edge* UW is *upper-dominated* if: $\delta = \mathcal{D}(U,V) + \phi$.

* (Muscettola, Morris, & Tsamardinos 1998)

Collaborative Planning with STNs

Initial-Commitment Decision Prob.*

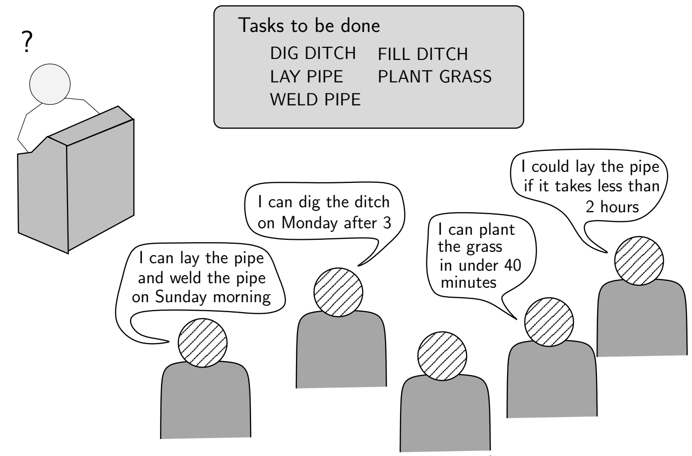


* ICDP (Hunsberger & Grosz 2000; Hunsberger 2002b)

The ICDP – in Words

- A group of agents, each with pre-existing commitments subject to temporal constraints
- A new opportunity for group action (a set of tasks also subject to temporal constraints)
- Agents must reason locally and globally about whether to commit (alone and together) to the proposed action.

ICDP Mech. using Combin'l. Auction*



* (Hunsberger & Grosz 2000; Hunsberger 2002b)

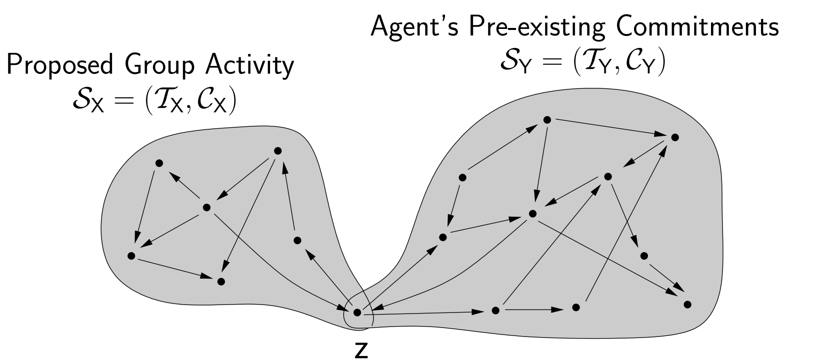
ICDP Mechanism – in Words

- Agents (reasoning locally) bid on subsets of tasks in group activity: a *combinatorial auction* (Rassenti, Smith, & Bulfin 1982).
- Agents include temporal constraints in their bids to protect their pre-existing commitments.
- Global goal: find an *awardable* set of bids (each task covered by some bid; temporal constraints in bids jointly satisfiable).

Problems to Solve re: ICDP

- Bid Generation:
  Select tasks and generate protective temporal constraints
- Winner Determination:
  Find an awardable set of bids.
- Post-Auction Coordination:
  Deal with temporal dependencies among tasks being done by different agents without requiring excessive communication overhead.

Bid Generation using STNs

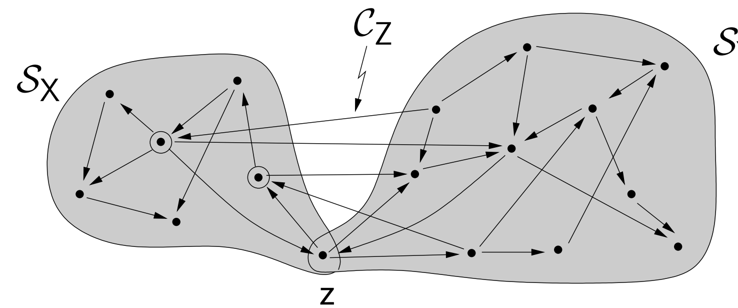


$\mathcal{S}_Z = (\mathcal{T}_Z, \mathcal{C}_Z) = (\mathcal{T}_X \cup \mathcal{T}_Y,\ \mathcal{C}_X \cup \mathcal{C}_Y)$ is consistent

if $\mathcal{S}_X$ and $\mathcal{S}_Y$ are (since they only share z).

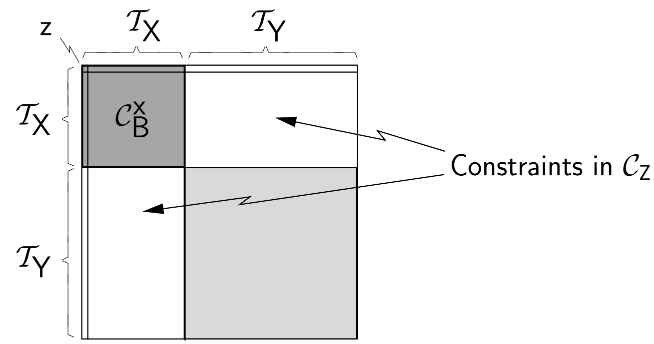
Bid Generation using STNs (cont'd.)

$\mathcal{S}_B = (\mathcal{T}_X \cup \mathcal{T}_Y,\ \mathcal{C}_X \cup \mathcal{C}_Y \cup \mathcal{C}_Z)$ includes additional constraints, $\mathcal{C}_Z$, to ensure that tasks done by the agent do not overlap.



Bid Generation using STNs (cont'd.)

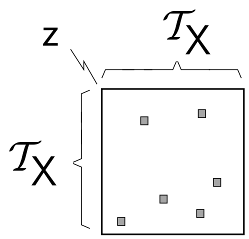
$\mathcal{D}_B$: The distance matrix for $\mathcal{S}_B$



$\mathcal{C}_B^x = \{t_j - t_i \leq \mathcal{D}_B(i,j) \mid t_i, t_j \in \mathcal{T}_X\}$ would suffice (in bid) to protect agent's pre-existing commitments.

Bid Generation using STNs (cont'd.)

…but necessary to include only edges in *canonical form* of $(\mathcal{T}_X, \mathcal{C}_B^x)$ that are *stronger* than the corresponding edges in $\mathcal{S}_X = (\mathcal{T}_X, \mathcal{C}_X)$ — i.e., edges for which $\mathcal{D}_B(i,j) < \mathcal{D}_X(i,j)$. (Hunsberger 2001)



Winner Determination *

- Modify existing WD algorithm (Sandholm 2002) to accommodate temporal constraints.
- Depth-first search in space of partial bid-sets
- Maintain STN, $(\mathcal{T}_X,\ \mathcal{C}_X \cup \mathcal{C}_{\mathcal{B}})$, containing constraints from proposed activity plus those from bids currently being considered.
- Backtrack if this STN becomes inconsistent.

* (Hunsberger & Grosz 2000)

Post-Auction Coordination

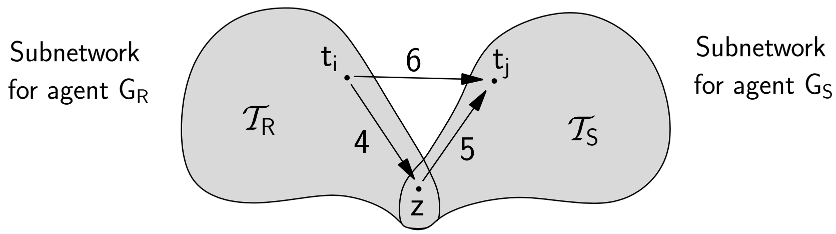
- Auction yields viable allocation of tasks, but typically results in temporal dependencies among tasks being done by different agents.
- Solution 1: *Temporally decouple* the task-sets being done by different agents (adds constraints, but no need for subsequent coord'n.).
- Solution 2: *Relative Temporal Decoupling* (weaker constraints, but requires some subsequent coordination).

Temporal Decoupling (TD)*

- Goal: Enable agents to operate independently —and hence without communication.
- Method: Add new constraints to ensure *mergeable solutions property*.
- Will focus on two-agent case, but works for arbitrarily many agents.

(Hunsberger 2002a; 2002b)

Typical Case for TD Problem



- Edge from t_i to t_j not dominated by a path through z.
- Can fix by strengthening edge from t_i to z, or edge from z to t_j, or both.

TD Algorithm*

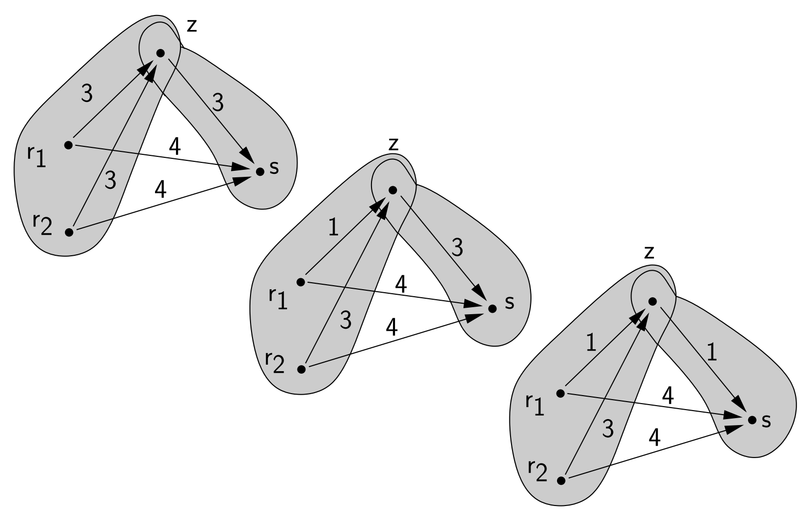
- Add *intra*-subnetwork constraints to ensure that each tight, proper, *inter*-subnetwork constraint is dominated by a path through z.
- Requires processing each such edge only once.
- Afterward, no matter how each agent tightens constraints in its own subnetwork, all inter-subnetwork constraints will be satisfied.

(Hunsberger 2002b)

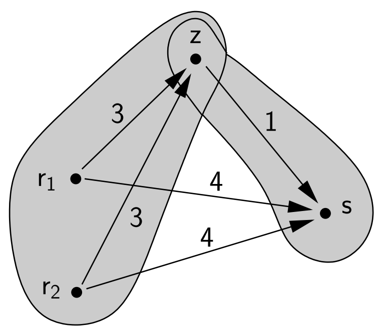
Improvements to TD Algorithm

- When selecting inter-subnetwork edges to work on, and when deciding how much to tighten each intra-subnetwork edge, use heuristics to increase flexibility in resultant decoupled subnetworks.
- Use *Iterative Weakening* algorithm to ensure *minimal* temporal decoupling (i.e., one in which any further weakening would foil the decoupling).

Generating a Non-Minimal Decoupling

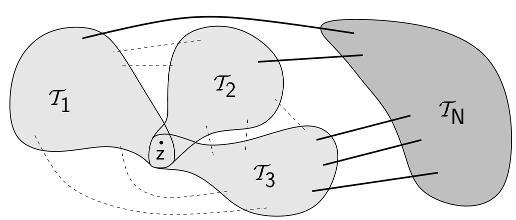


Alternative Minimal Decouplings



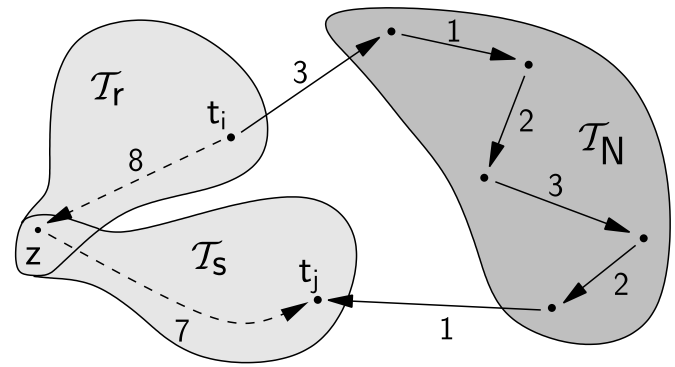
Relative Temporal Decoupling (RTD)*

- Goal: Use weaker constraints, but allow some inter-subnetwork dependence to remain.
- Method: Given N subnetworks, (N-1) are fully decoupled; but N^{th} dependent on the rest.



(Hunsberger 2003; 2002b)

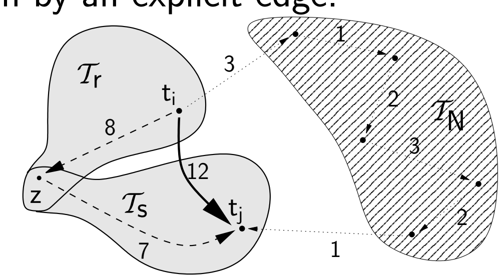
Typical Case for RTD Problem



Inter-subnetwork path from t_i to t_j is not dominated by path through z.

RTD Algorithm*

(1) Replace each *tight, proper*, inter-subnetwork path by an explicit edge.



(2) Run TD algorithm ignoring N^{th} subnetwork.

(Hunsberger 2003; 2002b)

Lambda Bounds for RTD*

- After RTD, agent controlling N^{th} subnetwork is dependent on the rest.
- Must not re-introduce any inter-subnetwork paths that would threaten the RTD. (Requirements captured in *Lambda Bounds*.)
- Unlike other agents, N^{th} agent may add edges linking N^{th} subnetwork with other subnetworks.

(Hunsberger 2003; 2002b)

Other Applications of RTD

- Submitting a bid imposes restrictions on the bidder that are precisely captured by the Lambda Bounds (where $N = 2$).
- The RTD algorithm may be recursively applied yielding an arbitrarily complex hierarchy of dependence and independence.
- Hadad et al. (2003) present an alternative approach to temporal reasoning in the context of collaboration.

References

- Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. Cambridge, MA: The MIT Press.
- Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.
- Demetrescu, C., and Italiano, G. F. 2001. Fully dynamic all pairs shortest paths with real edge weights. In *42nd Annual Symposium on Foundations of Computer Science (FOCS 2001)*. IEEE Computer Society. 260–267.
- Demetrescu, C., and Italiano, G. 2002. A new approach to dynamic all pairs shortest paths. Technical Report ALCOMFT-TR-02-92, ALCOM-FT. To appear in Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03), San Diego, California, June 2003.
- Even, S., and Gazit, H. 1985. Updating distances in dynamic graphs. *Methods of Operations Research* 49:371–387.
- Gerevini, A.; Perini, A.; and Ricci, F. 1996. Incremental algorithms for managing temporal constraints. Technical Report IRST-9605-07, IRST.
- Hadad, M.; Kraus, S.; Gal, Y.; and Lin, R. 2003. Time reasoning for a collaborative planning agent in a dynamic environtment. *Annals of Mathematics and Artificial Intelligence* 37(4):331–380.
- Hunsberger, L., and Grosz, B. J. 2000. A combinatorial auction for collaborative planning. In *Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, 151–158. IEEE Computer Society.
- Hunsberger, L. 2001. Generating bids for group-related actions in the context of prior commitments. In Meyer, J.-J. C., and Tambe, M., eds., *Intelligent Agents VIII (ATAL-01)*, volume 2333 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.
- Hunsberger, L. 2002a. Algorithms for a temporal decoupling

problem in multi-agent planning. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002).*

- Hunsberger, L. 2002b. *Group Decision Making and Temporal Reasoning*. Ph.D. Dissertation, Harvard University. Available as Harvard Technical Report TR-05-02.
- Hunsberger, L. 2003. Distributing the control of a temporal network among multiple agents. In *Proceedings of the Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS-03).*
- Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98).*
- Ramalingam, G., and Reps, T. 1996. On the computational complexity of dynamic graph problems. *Theoretical Computer Science* 158:233–277.
- Rassenti, S.; Smith, V.; and Bulfin, R. 1982. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics* 13:402–417.
- Rohnert, H. 1985. A dynamization of the all pairs least cost path problem. In Mehlhorn, K., ed., *2nd Symposium of Theoretical Aspects of Computer Science (STACS 85)*, volume 182 of *Lecture Notes in Computer Science*. Springer. 279–286.
- Sandholm, T. 2002. An algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence* 135:1–54.
- Tsamardinos, I., and Pollack, M. E. 2003. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence* 151:43–89.
- Tsamardinos, I.; Muscettola, N.; and Morris, P. 1998. Fast transformation of temporal plans for efficient execution. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98).* Cambridge, MA: The MIT Press. 254–261.
- Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. Master's thesis, University of Pittsburgh.
- Wetprasit, R., and Sattar, A. 1998. Qualitative and quantitative temporal reasoning with points and durations (an extended abstract). In *Fifth International Workshop on Temporal Representation and Reasoning (TIME-98)*, 69–73.
- Zaroliagis, C. D. 2002. Implementations and experimental studies of dynamic graph algorithms. In Fleischer, R.; Moret, B.; and Meineche-Schmidt, E., eds., *Experimental Algorithmics—The State of the Art*. Springer-Verlag. chapter 11, 229–278.

Beyond Simple Temporal Problems:

Increasing expressiveness and including uncertainty

The Breakfast Plan (Version 2)

Prepare coffee and toast. Have them ready within 2 minutes of each other. Brew coffee for 3-5 minutes; toast bread for 2-4 minutes.

Expressiveness and Uncertainty

- Increasing the expressiveness of the temporal constraints:
  - Definition Disjunctive Temporal Problem
  - Solving DTPs
  - Dispatching DTPs
  - Planning with temporal constraints
- Explicitly representing uncertainty
  - Uncontrollability and unobservability

Disjunctive Temporal Problems

Real Plans often have Disjunctive Constraints

- Typical Plan for an Autominder User

ACTION	TARGET TIME
Start laundry	Before 10 a.m.
Put clothes in dryer	Within 20 minutes of washer ending
Fold clothes	Within 20 minutes of dryer ending
Prepare lunch	Between 11:45 and 12:15
Eat lunch	At end of prepare lunch
Check pulse	Between 11:00 and 12:00, and between 3:00 and 4:00
Depending on pulse, take meds	At end of check pulse

Activity disjunct: Watch the news at 10pm or 11pm

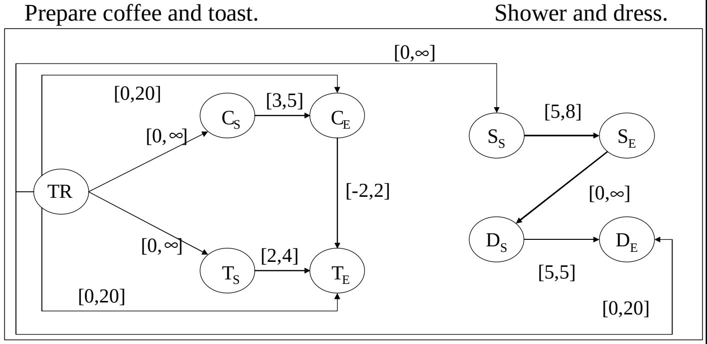
Non-overlap: $L_E - P_S \leq 0 \vee M_E - L_S \leq 0$

The ~~Breakfast~~ Plan (Version 3) Morning

Prepare coffee and toast. Have them ready within 2 minutes of each other. Brew coffee for 3-5 minutes; toast bread for 2-4 minutes. Also take a shower for 5-8 minutes, and get dressed, which takes 5 minutes. Be ready to go by 8:20.

The Morning Plan

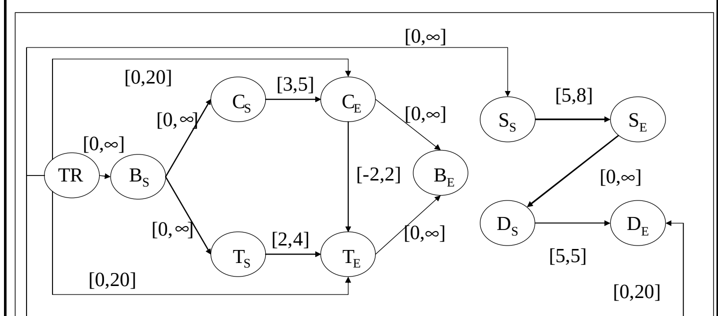
Prepare coffee and toast. Shower and dress.



$[(T_E \leq S_S) \wedge (C_E \leq S_S)] \vee [(D_E \leq C_S) \wedge (D_E \leq T_S)]$

Eat first. Dress first.

The Morning Plan



$B_E - S_S \leq 0 \vee D_E - B_S \leq 0$ disjunctive, not binary

Disjunctive Constraints

- Represent non-overlaps (as in our example)
- Can also represent other forms of disjunction
  - E.g., take a shower for 5 minutes or a bath for 10 minutes

Disjunctive Temporal Problems

- A set of time points (variables) V and a set of constraints C of the form:

  $lb_{ji} \leq X_i - X_j \leq ub_{ji} \vee \ldots \vee lb_{mk} \leq X_k - X_m \leq ub_{mk}$
- Benefit: Additional expressive power
- Cost: Additional computational expense—reasoning is NP-Hard
  - True even for *binary* problems, i.e., constraints have the form

  $lb_{ji} \leq X - Y \leq ub_{ji} \vee \ldots \vee lb_{mk} \leq X - Y \leq ub_{mk}$

DTPs as CSPs

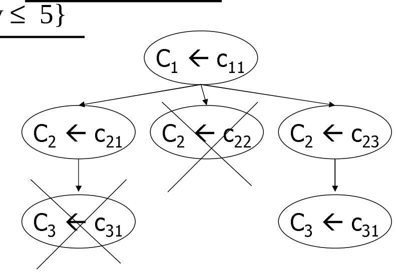
- One-Level Approach
  - Direct assignment of times to DTP variables.
  - Limitations: difficult to deal with infinite domains; produces overconstrained solution
- Two-Level Approach
  - Construct a meta-level CSP
  - Variables: DTP constraints
  - Domains: Disjuncts from DTP constraints.
  - Constraints: Implicit, assignment must lead to a *consistent component STP*

DTP Solving Example

$C_1 : \{c_{11} : y - x \leq 5\}$

$C_2 : \{c_{21} : w - y \leq 5\} \vee \{c_{22} : x - y \leq -10\} \vee \{c_{23} : z - y \leq 5\}$

$C_3 : \{c_{31} : y - w \leq -10\}$



Component STP:
$C_1 \leftarrow c_{11}, C_2 \leftarrow c_{23}, C_3 \leftarrow c_{31}$

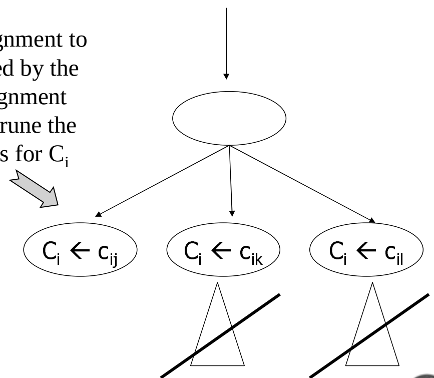
One exact solution:
$\{x = 0, y = 1, z = 2, w = 12\}$

Strategies for Efficiency

- Forward checking / incremental forward checking
- Conflict-directed backjumping
- Removal of subsumed variables
- Semantic branching
- No-good learning
- Use efficient SAT solvers for meta-level

Removal of Subsumed Variables

If this assignment to C_i is implied by the partial assignment above it, prune the other values for C_i



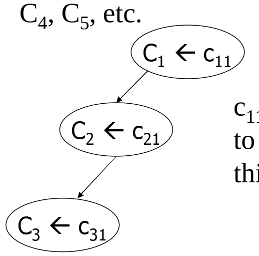
Removal of Subsumed Variables

$C_1 : \{c_{11} : y - x \leq 5\}$

$C_2 : \{c_{21} : x - z \leq 5\} \vee \{c_{22} : w - y \leq -10\}$

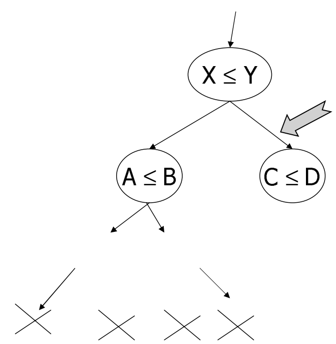
$C_3 : \{c_{31} : y - z \leq 15\} \vee \{c_{32} : z - v \leq 10\} \vee \ldots$

C_4, C_5, etc.



c_{11} and c_{21} imply c_{31}, so no need to try other values for C_3 along this branch

Semantic Branching



Also impose B < A
(I.e. ¬ A ≤ B)

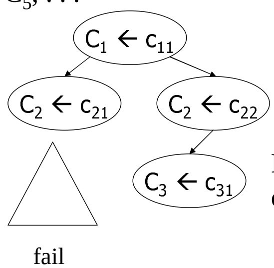
Semantic Branching

$C_1 : \{c_{11} : x - y \leq 5\}$

$C_2 : \{c_{21} : x - z \leq 3\} \vee \{c_{22} : w - z \leq -6\}$

$C_3 : \{c_{31} : y - w \leq 2\} \vee \{c_{32} : w - y \leq 0\} \vee \ldots$

$C_4, C_5, \ldots$



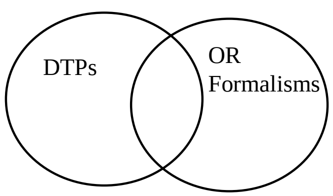
Add $\neg c_{21}$: $x\text{-}z > 3$

Fail immediately:
$c_{11}, c_{22}, c_{31}, \neg c_{21}$ inconsistent

So, how fast?

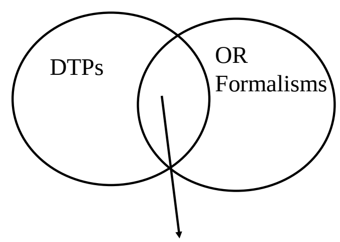
- Current fastest solver, TSAT++, reports:
  - ~10 seconds to solve problems with
    - 35 variables
    - ~210 disjunctive constraints (critical region)
    - Each with 2 disjuncts

DTP Solving and OR Scheduling Formalisms



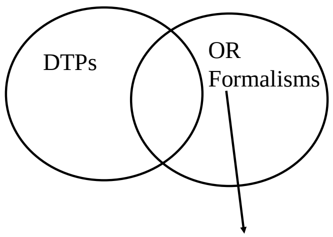
DTPs designed for the needs of
planning with temporal constraints

DTP Solving and OR Scheduling Formalisms



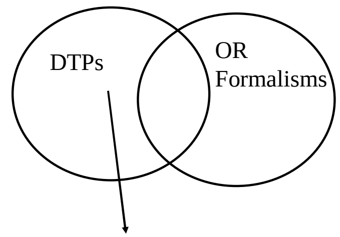
<u>Example:</u> Job Shop Scheduling
Temporal precedence constraints: easy to model with DTPs
Resource constraints: more cumbersome with DTPs

DTP Solving and OR Scheduling Formalisms



<u>Example:</u> Preemption

DTP Solving and OR Scheduling Formalisms

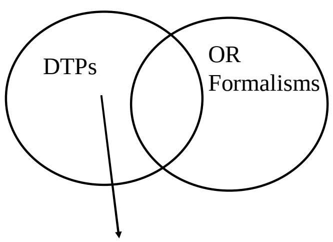


<u>Example:</u> Arbitrary Disjunction

JSS & DTP can both express non-overlap constraints

$A < B \vee B < A$ (binary with intervals (tasks), non-binary with time points)

DTP Solving and OR Scheduling Formalisms



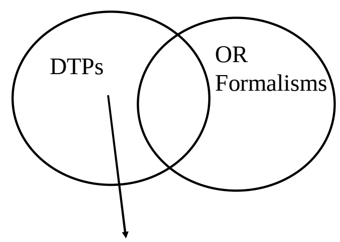
But only DTPs can express general constraints

"If treatment A doesn't last long enough, perform treatment B for a given duration."

$\sim((A_E - A_S) > d) \rightarrow (B_E - B_S) > e$

$\equiv \ (A_S - A_E) < -d \vee (B_S - B_E) < -e$

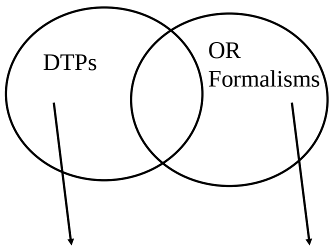
DTP Solving and OR Scheduling Formalisms



Some DTP solvers provide justifications of failure (e.g., minimal sets of inconsistent input constraints)

Useful in plan generation

DTP Solving and OR Scheduling Formalisms



Decision problems:
Often hard to satisfy

Optimization problems:
Often easy to satisfy, but
hard to optimize

Dispatch with DTPs

DTP Dispatch Method #1

- With total control of the execution process:

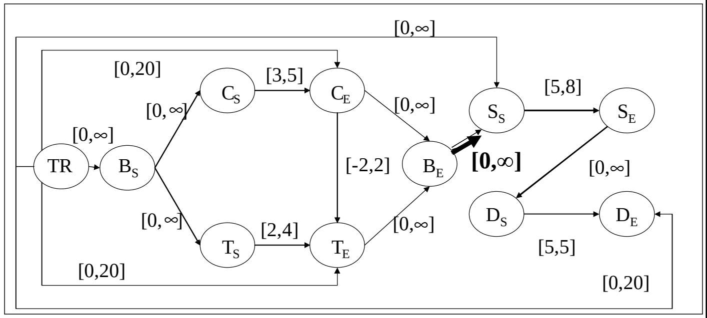
- Given a DTP, find a consistent component STP S
- Dispatch S using STP dispatch algorithm

DTP Dispatch Method #2

- With partial control of the execution process (e.g., in execution monitoring)

- Given a DTP, find a consistent component STP S
- While no events inconsistent with S occur
  - Dispatch S using STP dispatch algorithm
- Otherwise, if event e occurs at time t that is inconsistent with S
  - Add an execution constraint, $t \leq e - \mathrm{TR} \leq t$
  - Find a new consistent component STP S

The Morning Plan



[redacted] $\vee\ D_E - B_S \le 0$

Detect S_s at 8:03

Add: $3 \le Ss - TR \le 3$

A Problem

- Might "miss" a solution

- [redacted] $\vee\ X = 1$
- [redacted] $\vee\ Y = 2$
- [redacted]

- Don't see anything at 1
- See Y at 2

All remaining consistent component STPs are eliminated

DTP Dispatch Method #3

- Produce information about what *can be done*
  - Execution Table
    - Specifies what actions are live and enabled (what can be done)
    - An event *e* in a DTP is live iff *now* is in its time window
    - An event *e* in a DTP is enabled iff it is enabled in at least one consistent component STP
- And what *must be done*
  - Deadline Formula
    - Specifies what deadline must be satisfied next (what must be done)

Example

C_1: $\{c_{11}: 5 \le x - TR \le 10\} \vee \{c_{12}: 15 \le x - TR \le 20\}$

C_2: $\{c_{21}: 5 \le y - TR \le 10\} \vee \{c_{22}: 15 \le y - TR \le 20\}$

C_3: $\{c_{31}: 6 \le x - y \le \infty\} \vee \{c_{32}: 6 \le y - x \le \infty\}$

C_4: $\{c_{41}: 11 \le z - TR \le 12\} \vee \{c_{42}: 21 \le z - TR \le 22\}$

Consistent Component STPs:

1. STP1: c_{11}, c_{22}, c_{32}, c_{41} x before y, z early
2. STP2: c_{11}, c_{22}, c_{32}, c_{42} x before y, z late
3. STP3: c_{12}, c_{21}, c_{31}, c_{41} y before x, z early
4. STP4: c_{12}, c_{21}, c_{31}, c_{42} y before x, z late

Example

C_1: $\{c_{11}: 5 \le x - TR \le 10\} \vee \{c_{12}: 15 \le x - TR \le 20\}$

C_2: $\{c_{21}: 5 \le y - TR \le 10\} \vee \{c_{22}: 15 \le y - TR \le 20\}$

C_3: $\{c_{31}: 6 \le x - y \le \infty\} \vee \{c_{32}: 6 \le y - x \le \infty\}$

C_4: $\{c_{41}: 11 \le z - TR \le 12\} \vee \{c_{42}: 21 \le z - TR \le 22\}$

Execution Table:

<x, {[5,10], [15,20]}>

<y, {[5,10], [15,20]}>

Enabled events and their time windows

Deadline Formula:

<x ∨ y, 10>

CNF formula that must be satisfied "next"

Dispatch Method

- Computing the Execution Table:
  - Find all enabled events
  - Compute their time windows in every consistent component STP
- Computing the Deadline Formula:
  - Find the next time at which *some* event must occur
  - Find all events that *might* have to occur by that time point
  - Compute the minimal event sets that would ensure that not all remaining consistent component STPs are eliminated

Generating the Deadline Formula

Generate-DF (Solutions: STP [i])

Let U = the set of upper bounds on time windows, U(x,i) for each still unexecuted action x and each STP i.

Let NC, the next critical time point, be the value of the minimum bound in U.

Let $U_{MIN} = \{U(x, i) | U(x,i) = NC\}$.

For each x such that $U(x,i) \in U_{MIN}$, let $S_x = \{i \mid U(x,i) \in U_{MIN}\}$

Initialize F = true;

For each minimal solution MinCover of the set-cover problem (Solutions , $\cup S_x$), let $F = F \wedge (\vee\, x \mid S_x \in$ MinCover $x)$.

Output DF = <F, NC>.

Generating the Deadline Formula

Generate-DF (Solutions: STP [i])

Let U = the set of upper bounds on time windows, U(x,i) for each still unexecuted action x and each STP i.

Let NC, the next critical time point, be the value of the minimum upper bound in U.

Let $U_{MIN} = \{U(x, i) | U(x,i) = NC\}$.

For each x such that $U(x,i) \in U_{MIN}$, let $S_x = \{i \mid U(x,i) \in U_{MIN}\}$

Initialize F = true;

For each minimal solution MinCover of the set-cover problem (Solutions , $\cup S_x$), let $F = F \wedge (\vee\, x \mid S_x \in$ MinCover $x)$.

Output DF = <F, NC>.

Example

C1: {c11: $5 \le x - TR \le 10$} $\vee$ {c12: $15 \le x - TR \le 20$}

C2: {c21: $5 \le y - TR \le 10$} $\vee$ {c22: $15 \le y - TR \le 20$}

C3: {c31: $6 \le x - y \le \infty$} $\vee$ {c32: $6 \le y - x \le \infty$}

C4: {c41: $11 \le z - TR \le 12$} $\vee$ {c42: $21 \le z - TR \le 22$}

Consistent Component STPs:

STP1: c11, c22, c32, c41

STP2: c11, c22, c32, c42

STP3: c12, c21, c31, c41

STP4: c12, c21, c31, c42

U(x,1) = U(x,2) = 10
U(x,3) = U(x,4) = 20
U(y,1) = U(y,2) = 20
U(y,3) = U(y,4) = 10
U(z,1) = U(z,3) = 12
U(z,2) = U(z,4) = 22

Generating the Deadline Formula

Generate-DF (Solutions: STP [i])

Let U = the set of upper bounds on time windows, U(x,i) for each still unexecuted action x and each STP i.

Let NC, the next critical time point, be the value of the minimum upper bound in U.

Let U_{MIN} = {U(x, i)| U(x,i) = NC}.

For each x such that U(x,i) $\in U_{MIN}$, let S_x = {i | U(x,i) $\in U_{MIN}$}

Initialize F = true;

For each minimal solution MinCover of the set-cover problem (Solutions , $\cup S_x$), let F = F $\wedge$ ($\vee$ x | $S_x \in$ MinCover x).

Output DF = <F, NC>.

Example

C1: {c11: $5 \le x - TR \le 10$} $\vee$ {c12: $15 \le x - TR \le 20$}

C2: {c21: $5 \le y - TR \le 10$} $\vee$ {c22: $15 \le y - TR \le 20$}

C3: {c31: $6 \le x - y \le \infty$} $\vee$ {c32: $6 \le y - x \le \infty$}

C4: {c41: $11 \le z - TR \le 12$} $\vee$ {c42: $21 \le z - TR \le 22$}

Consistent Component STPs:

STP1: c11, c22, c32, c41

STP2: c11, c22, c32, c42

STP3: c12, c21, c31, c41

STP4: c12, c21, c31, c42

U(x,1) = U(x,2) = 10
U(x,3) = U(x,4) = 20
U(y,1) = U(y, 2) = 20
U(y,3) = U(y, 4) = 10
U(z,1) = U(z,3) = 12
U(z,2) = U(z,4) = 22

NC = 10
U_{MIN} = {(x,1), (x,2),(y,3),(y,4)}

Generating the Deadline Formula

Generate-DF (Solutions: STP [i])

Let U = the set of upper bounds on time windows, U(x,i) for each still unexecuted action x and each STP i.

Let NC, the next critical time point, be the value of the minimum upper bound in U.

Let U_{MIN} = {U(x, i)| U(x,i) = NC}.

For each x such that U(x,i) $\in U_{MIN}$, let S_x = {i | U(x,i) $\in U_{MIN}$}

Initialize F = true;

For each minimal solution MinCover of the set-cover problem (Solutions , $\cup S_x$), let F = F $\wedge$ ($\vee$ x | $S_x \in$ MinCover x).

Output DF = <F, NC>.

Example

41

C1: {c11: $5 \le x - TR \le 10$} $\vee$ {c12: $15 \le x - TR \le 20$}

C2: {c21: $5 \le y - TR \le 10$} $\vee$ {c22: $15 \le y - TR \le 20$}

C3: {c31: $6 \le x - y \le \infty$} $\vee$ {c32: $6 \le y - x \le \infty$}

C4: {c41: $11 \le z - TR \le 12$} $\vee$ {c42: $21 \le z - TR \le 22$}

Consistent Component STPs:

STP1: c11, c22, c32, c41

STP2: c11, c22, c32, c42

STP3: c12, c21, c31, c41

STP4: c12, c21, c31, c42

NC = 10

U_{MIN} = {(x,1), (x,2),(y,3),(y,4)}

S_x = {1,2}

S_y = {3,4}

Generating the Deadline Formula

42

Generate-DF (Solutions: STP [i])

Let U = the set of upper bounds on time windows, U(x,i) for each still unexecuted action x and each STP i.

Let NC, the next critical time point, be the value of the minimum upper bound in U.

Let U_{MIN} = {U(x, i)| U(x,i) = NC}.

For each x such that U(x,i) $\in U_{MIN}$, let S_x = {i | U(x,i) $\in U_{MIN}$}

Initialize F = true;

For each minimal solution MinCover of the set-cover problem (Solutions , $\cup S_x$), let F = F $\wedge$ ($\vee$ x | $S_x \in$ MinCover x).

Output DF = <F, NC>.

Example

43

C1: {c11: $5 \le x - TR \le 10$} $\vee$ {c12: $15 \le x - TR \le 20$}

C2: {c21: $5 \le y - TR \le 10$} $\vee$ {c22: $15 \le y - TR \le 20$}

C3: {c31: $6 \le x - y \le \infty$} $\vee$ {c32: $6 \le y - x \le \infty$}

C4: {c41: $11 \le z - TR \le 12$} $\vee$ {c42: $21 \le z - TR \le 22$}

Consistent Component STPs:

STP1: c11, c22, c32, c41

STP2: c11, c22, c32, c42

STP3: c12, c21, c31, c41

STP4: c12, c21, c31, c42

S_x = {1,2}

S_y = {3,4}

MSC({1,2,3,4},{S_x, S_y}) = {S_x, S_y}

F = $x \vee y$

Larger Deadline Formula

44

- Suppose
  - 4 consistent component STPs
  - NC = 10
  - U (x, 1) = U (x, 2) = U (y, 3) = U (y, 4) = U (z, 4) = U (w, 3) = 10
- The minimal set covers are
  - {S_x, S_y} and {S_x, S_w, S_z}
- So the deadline formula is
  - $(x \vee y) \wedge (x \vee z \vee w)$

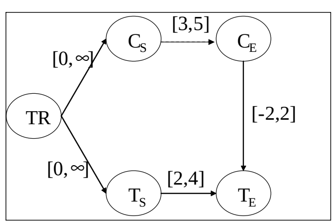
The Dispatch Bottleneck

- Requires computation of *all* component STPs
- May be exponentially many of them
- Open Research Question: Can we identify "representative" sets of component STPs?

Uncontrollability and unobservability

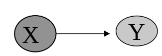
Breakfast Again

- You don't really get to control how long the coffee brews (but you can pop the toast at any time).

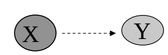


Handling Temporal Uncertainty

- TP-u (e.g., STP-u)
- Distinguish between two kinds of events:
  - Controllable: the executing agent controls the time of occurrence
  - Uncontrollable: "nature" controls the time of occurrence



Controllable edge (Y controllable event)



Uncontrollable edge (Y uncontrollable event)

Three Notions of "Solution"

- *Strongly Controllable*: There is an assignment of time points to the controllable events such that the constraints will be satisfied regardless of when the uncontrollables occur.
- One (or more) solutions that work no matter what!

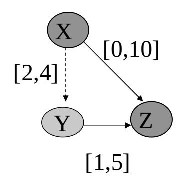
Three Notions of "Solution"

- *Weakly Controllable*: For each outcome of the uncontrollables, there is an assignment of time points to the controllables such that the constraints are satisfied.
- One (or more) solutions that work for each outcome.

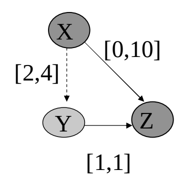
Three Notions of "Solution"

- *Dynamically Controllable*: As time progresses and uncontrollables occur, assignments can be made to the controllables such that the constraints are satisfied.
- Solutions that are guaranteed to work can be created conditionally to observations.

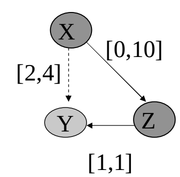
Controllability in STP-u's



Strongly Controllable
{X=0, Z = 5}



Dynamically Controllable
{X=0, Z = Y + 1}

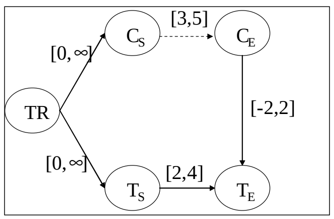


Weakly Controllable
{X=0, Z = Y – 1}

Strong => Dynamic => Weak

Breakfast Again

- You don’t really get to control how long the coffee brews (but you can pop the toast at any time).



Is it controllable?

Yes, strongly controllable:
$C_S = 0$
$T_s = 0$
$T_E = 3$ (but not 2)

Controllability and Observability

- Different notions of controllability make different assumptions about what can be observed
- *Strong Controllability:* uncontrollable events cannot be observed and consistency must be guaranteed
- *Dynamic Controllability:* uncontrollable events can be observed and consistency must be guaranteed
- *Weak Controllability:* “I’m feeling lucky”… and luck will always be in a position to help achieve consistency

Controllability and Dispatchability

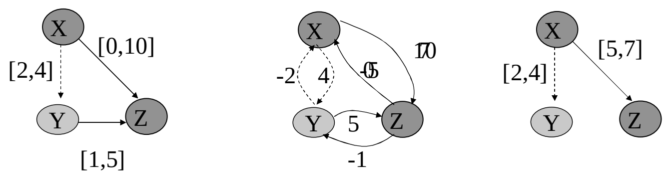
- Controllability: defines policies to determine times for controllable events depending on knowledge of uncontrollable events occurrence
- Dispatchability: identifies effective propagation paths such that knowledge on the execution of an event constrains the possible execution times for other events

Execution Policies

- Controllability definition emphasizes existence of solutions
- At execution time we need policies to make decision as a function of our knowledge
  - Clock time
  - Observation of event occurrence (if possible)
- Like in the case of STPs, provide ways to determine bounds and repropagation methods to create solutions on the fly

Strongly controllable policies

•We need to come up with policies assuming no knowledge about the uncontrollable event
•Solution: disconnect any dispatchable link from the event



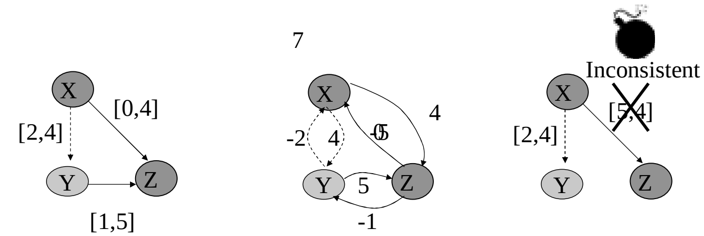
Strongly Controllable

$YZ \leq YX + XZ$

Step 1: tighten
$XZ = YZ - YX$

Step 2: delete YZ

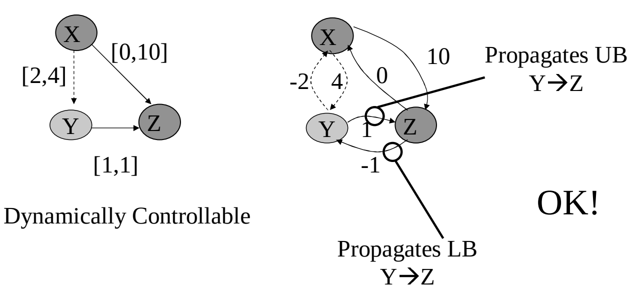
Strongly controllable policies



Pseudo-Controllability

- The upper and lower bounds of an uncontrollable event are not necessarily propagated outside of the uncontrollable link (no *necessary tightening* of uncontrollable links) ☺
- Bound propagation can originate from an uncontrollable event because we can have knowledge of its occurrence… ☺
- … but during execution there can be executions that propagate *into* the uncontrollable event tighter bounds than the uncontrollable link (*possible tightening* of the uncontrollable links) ☹

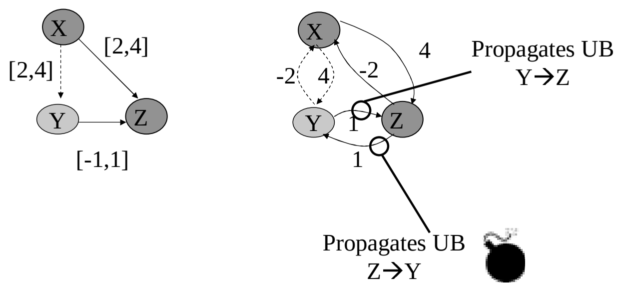
Pseudo-controllable policies



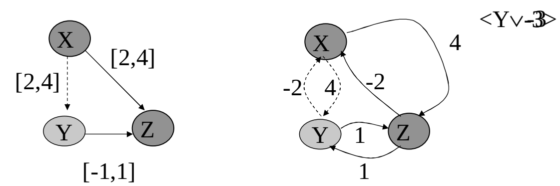
Dynamically Controllable

OK!

Pseudo-controllable policies



Tightening of controllable links

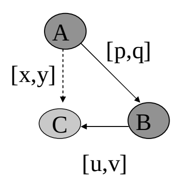


Step 1: tighten
ZX = ZY – XY
(no knowledge of Y occurrence)

Step 2: Add conditional stop
If Y has occurred, then Z can

Computing Dynamic Controllability of an STPU

- Use *triangular reductions*
- Case 1: v < 0
  - B *follows* C, so d.c.
- Case 2: u ≥ 0
  - B *precedes* C: tighten AB to [y-v, x-u] to make d.c.
- Case 3: u < 0 and v ≥ 0
  - B is *unordered* w.r.t C: tighten lower bound of AB to (C or y-v) to make d.c.
- Iterate on the entire network

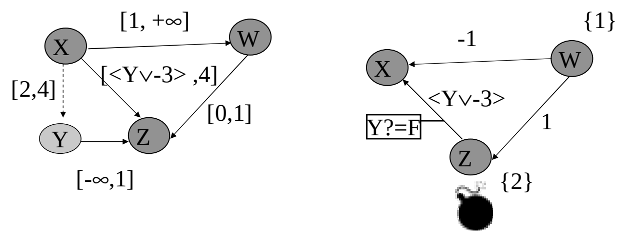


Wait Propagation Rules

- “Wait links” are a new type of “partially uncontrollable” link
- If they are present, they cause execution to be contingent on the occurrence of events
- Unlike uncontrollable links, they can be eliminated through tightening

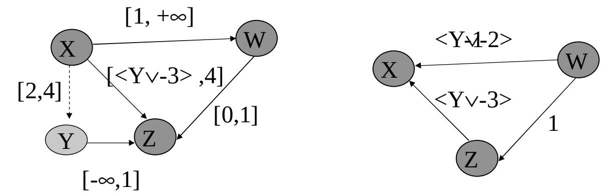
Wait Propagation over Controllable Edges

65



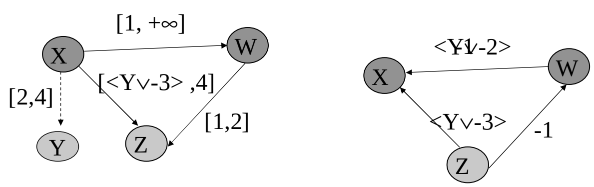
Wait Propagation over Controllable Edges

66



Wait Propagation over Uncontrollable Edges

67



Full Dynamic Controllability Algorithm

68

Loop
{
Compute pseudo-controllability of network;
if (network is inconsistent or not-pseudo controllable)
return “NON DYNAMICALLY CONTROLLABLE”
if (network is pseudo-controllable)
For all ABC triangles in the temporal network
perform all applicable tightenings (triangular reductions and wait regressions)
if no tightening were performed
return “DYNAMICALLY CONTROLLABLE”
}

69

Termination Condition

- Without further analysis, the algorithm is pseudo-polynomial
  - Pseudo-controllability: $O(NE + N^2 \log N)$
  - Tightening: $O(N^3)$
  - Number of repetition of cycle: U, number of time units in widest time bound
- Complexity: $O(U\ N^3)$
- U could be very large

70

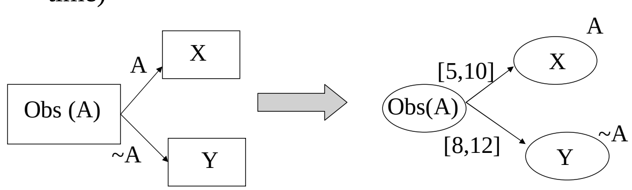
Cutoff bound

- Since the number of edges is finite, indefinite tightening is due to the existence of propagation cycles
- Cycle traversal must repeat after a maximum number of propagation (as in the Bellman-Ford algorithm for shortest paths
- Cutoff bound for dynamic controllability:
  - O(NK) with K = number of non-controllable links
- Cutoff on the number of cycles gives $O(KN^4)$ complexity bound.

71

Handling Causal Uncertainty

- CTP (e.g., CSTP)
- Label each node—events are executed only if their associated label is true (at a specified observation time)

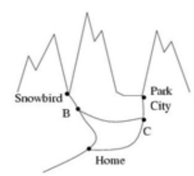


Conditional Plan

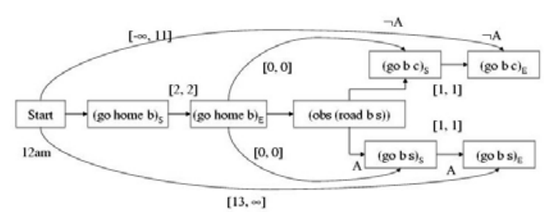
CTP

72

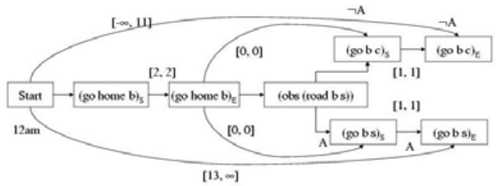
Conditional Plan as CTP



Travel from Home to S, but if the road is blocked from B to S, go to P.
If you go to S, arrive after 1p.m. (to take advantage of the discounts).
If you go to P, arrive at C by 11 a.m. (because traffic gets heavy).

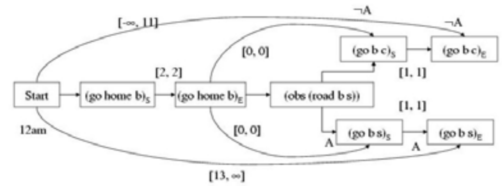


Strong Consistency



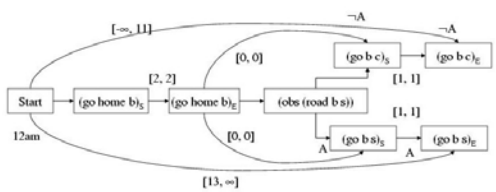
- Not strongly consistent: Must not be at B before 12 (if A is true); must be at B by 10 (if A is false)—and can't observe A until you're at B.

Weak Consistency



- Weakly consistent: When A is true, leave home after 10 (and all other assignments directly follow). When A is false, leave home by 9.

Dynamic Consistency



- Not dynamically consistent: Can't tell when you need to leave home until it's too late.
- Variant that is is dynamically consistent: Add a parking lot at B where you can wait.

Generating Temporal Plans

Generating Temporal Plans

- Various models have been developed, dating back to the early 1980's (DEVISER)
- Beginning to see a convergence in the *Constraint-Based Interval* approach
- Model the world with
  - Attributes (features): e.g., coffee
  - Values that hold over intervals: e.g., brewing
  - Times points that bound the intervals: e.g., b_t, b_e
  - Axioms that relate the values

Features and Values

<u>Feature</u>	<u>Domain of Values</u>
Coffee	none, brewing, ready, stale
Bread	untoasted, toasting, toast
Toaster-Status	on, off
Toaster-Contents	empty, full
Showering	yes, no
Bathing	yes, no
Clean	yes, no
Dressed	no, dressing, yes
Location	at(X), going(X,Y)

Temporally Quantified Assertions

- Each feature takes a *single* value at a time, i.e. formally there are a set of functions $f_i(\text{feature}_i, \text{time}_j) \rightarrow \text{value}_{i,j}$
  where $\text{value}_{i,j} \in \text{domain}(\text{feature}_i)$
- Temporally qualified assertions (tqa's or just "assertions"):
  holds (coffee, 8:03, 8:05, brewing)
  holds (toaster-content, X, Y, empty)
- Uniqueness Constraints:
  $\text{holds}(F,s,e,P) \wedge \text{holds}(F,s',e',Q) \rightarrow$
  $[e < s' \vee e' < s \vee P = Q]$

Planning Axioms

- Used to model actions
- Basic form

  Effect $\rightarrow$

  $(\text{Action}_1 \wedge \text{Preconditions}_1 \wedge \text{Constraints}_1) \vee$

  $(\text{Action}_2 \wedge \text{Preconditions}_2 \wedge \text{Constraints}_2) \vee$

  . . .

  $(\text{Action}_n \wedge \text{Preconditions}_n \wedge \text{Constraints}_n)$
- Can also partition the knowledge differently
- And can also use axioms to model other types of constraints (e.g., mutual exclusion)

Example 1

holds(coffee, r_s, r_e, ready) → ⟸ Effect
holds(coffee, b_s, b_e, brewing) $\wedge$ ⟸ Action
($b_e = r_s$) $\wedge$ ($3 \leq b_e - b_s \leq 5$) ⟸ Add'l. Constraints
holds(coffee, n_s, n_e, none) $\wedge$ ⟸ Preconditions
$n_e = b_s$ ⟸ Add'l. Constraints

Can also split out into two axioms
Effect → Action
Action → Preconditions

Example 2

holds(clean, c_s, c_e, yes) → ⟸ Effect
[holds(showering, h_s, h_e, yes) $\wedge$
$h_e = c_s \wedge c_e - c_s \leq 120$] $\vee$
[holds(bathing, b_s, b_e, yes) $\wedge$
$b_e = c_s \wedge c_e - c_s \leq 120$]

Alternative Actions

Example 3

holds(bread, r_s, r_e, toasting) →
holds(toaster-status, t_s, t_e, on) $\wedge$
$t_s = r_s \wedge t_e = r_e$
holds(toaster-contents, c_s, c_e, full) $\wedge$
$c_s \leq r_s \wedge r_e \leq c_e \wedge$

More "interesting" temporal constraints

Example 4

"Don't blow a fuse!"
[holds(coffee, b_s, b_e, brewing) $\wedge$
holds(toaster-status, t_s, t_e, on)] →
$b_e < t_s \vee t_e < b_s$ ⟸ Mutual exclusion

- Additional mutual exclusion constraints are implicit in uniqueness constraints

Planning Axioms

General Form:

Assertion $\wedge$ Assertion $\wedge$ . . . Assertion $\rightarrow$ $\Leftarrow$ Head

(Assertions $\wedge$ Constraints) $\vee$

(Assertions $\wedge$ Constraints) $\vee$ $\Leftarrow$ Alternatives

. . .

(Assertions $\wedge$ Constraints)

The Planning Problem

- Given a set of features and their domain, a (partial) plan is
  - a set of assertions on those features and
  - a set of constraints on the time points of the assertions
- A solution is
  - a complete assignment of values to features
  - such that all of the constraints are satisfied

The Initial Partial Morning Plan

	assertions	constraints
Coffee	$\boxed{\text{ready}(r_s,r_e)}$	$-2 \le r_e - t_e \le 2$
Bread	$\boxed{\text{toast}(t_s,t_e)}$	$r_e - TR \le 500$
Toaster-status		$t_e - TR \le 500$
Toaster-contents		$d_e - TR \le 500$
Clean		
Showering		
Bathing		
Dressed	$\boxed{\text{yes}(d_s,d_e)}$	

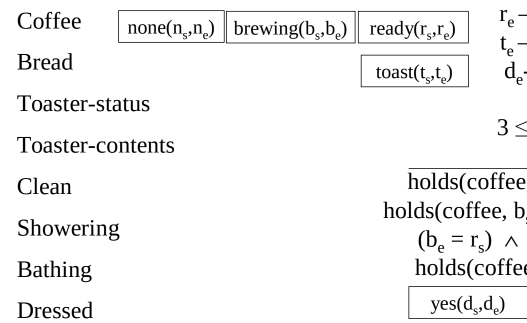
Expanding a Plan

- Select an assertion
- Find all the axioms that *apply* to it
- For each of those axioms
  - Choose an alternative (one disjunct in the tail of the axiom)
  - Ensure that the assertions and constraints in the chosen disjunct are in the plan, either by adding them or unifying them with assertions and constraints already present

Applicable Axioms

- Given
  - plan P
  - assertion A and
  - axiom M: $X_1 \wedge \ldots X_n \rightarrow$ r.h.s.
- M *applies to* A if
  - For some i, unify $(X_i, M) = \theta$, and
  - For all j = 1. . . n s.t. $j \neq i$, unify$(X_j, B) = \theta'$ where
    (i) θ' is an extension of θ, and
    (ii) B is an assertion in P

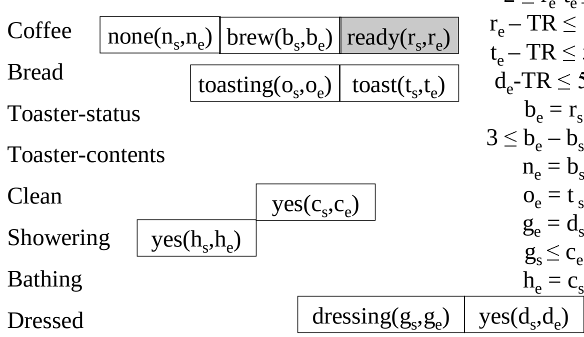
Expanding the Initial Plan I



$-2 \leq r_e - t_e \leq 2$
$r_e - TR \leq 500$
$t_e - TR \leq 500$
$d_e - TR \leq 500$
$b_e = r_s$
$3 \leq b_e - b_s \leq 5$
$n_e = b_s$

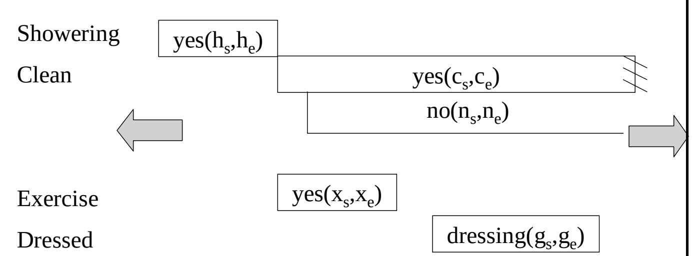
holds(coffee, r_s, r_e, ready) $\rightarrow$
holds(coffee, b_s, b_e, brewing) $\wedge$
$(b_e = r_s) \wedge (3 \leq b_e - b_s \leq 5)$
holds(coffee, n_s, n_e, none) $\wedge$
$n_e = b_s$

Expanding the Initial Plan II



$-2 \leq r_e - t_e \leq 2$
$r_e - TR \leq 500$
$t_e - TR \leq 500$
$d_e - TR \leq 500$
$b_e = r_s$
$3 \leq b_e - b_s \leq 5$
$n_e = b_s$
$o_e = t_s$
$g_e = d_s$
$g_s \leq c_e$
$h_e = c_s$
$c_e - c_s \leq 120$

Causal Links and Uniqueness Conditions



Uniqueness Constraint: $c_e \leq n_s \vee n_e \leq c_s$

Step Reuse

Coffee	none(n_s,n_e)	brewing(b_s,b_e)	ready(r_s,r_e)
Bread		toasting(o_s,o_e)	toast(t_s,t_e)
Location	at(kitchen,l_s,l_e)		

$l_s \leq b_s$
$b_s \leq l_e$
$l_s \leq o_s$
$o_s \leq l_e$

Underlying Constraint Network

- The temporal constraints form a DTP
- Technically, a dynamic DTP, since time points are added incrementally
- Use DTP techniques to check consistency efficiently

CBI Planning Algorithm

Unchecked, Assertions ← initial assertions
<u>Expand</u> (Unchecked, Assertions, Constraints, Axioms)
  If Constraints are inconsistent, fail.
  If Unchecked = ∅, return <Assertions, Constraints>.
  Select u ∈ Unchecked
    For every axiom X ∈ Axioms that applies to u
    Choose an alternative d from X {d is the result of the unification that causes X to be applicable}
      For each assertion s ∈ d
      Choose:
        Reuse: Unify s with an assertion in Assertions
        New: Add s to Assertions and Unchecked
      Add constraints c ∈ d to Constraints
  Expand(Unchecked, Assertions, Constraints, Axioms)

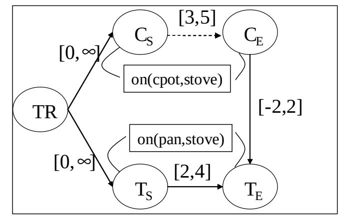
Resource Constraint Reasoning: Scheduling

Outline

- Resource representations
- Relationship between planning and scheduling representations
- Search spaces: flexible plans and fixed time instantiations
- Resource contention measures
  - Probabilistic
  - Lower/upper bounds
  - Envelopes

Breakfast at Yosemite

- You are backpacking so you cook the toast on a pan…
- …and you have a stove with just one burner.



Operating the stove
The Planning Perspective

?x ∈ {cpot, pan}

clear(stove) —pre→ putOn (?x, stove)
- add → on (?x, stove)
- del → clear (stove)

on (?x, stove) —pre→ takeOff (?x, stove)
- add → clear (stove)
- del → on (?x, stove)

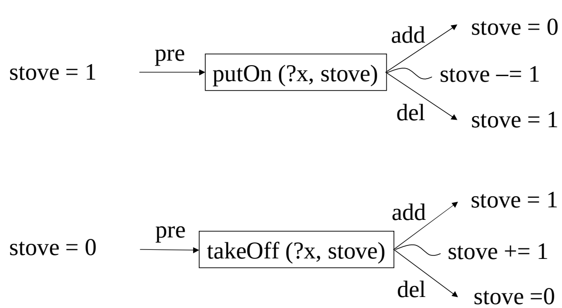
From Planning to Scheduling

stove = 1, clear(stove) —pre→ putOn (?x, stove)
- add → stove = 0, on (?x, stove)
- del → stove = 1, clear (stove)

stove = 0, on (?x, stove) —pre→ takeOff (?x, stove)
- add → stove = 1, clear (stove)
- del → stove =0, on (?x, stove)

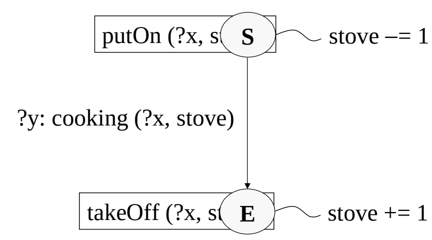
From Planning to Scheduling

0 ≤ stove ≤ 1



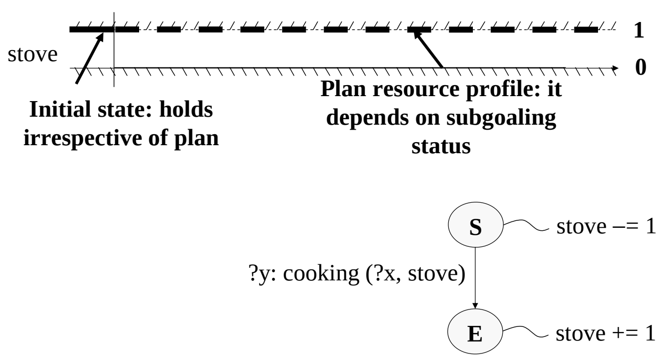
From Planning to Scheduling

0 ≤ stove ≤ 1



<?y, ?x> ∈ {<coffe, cpot>, <toast, pan>}

Breakfast as Scheduling

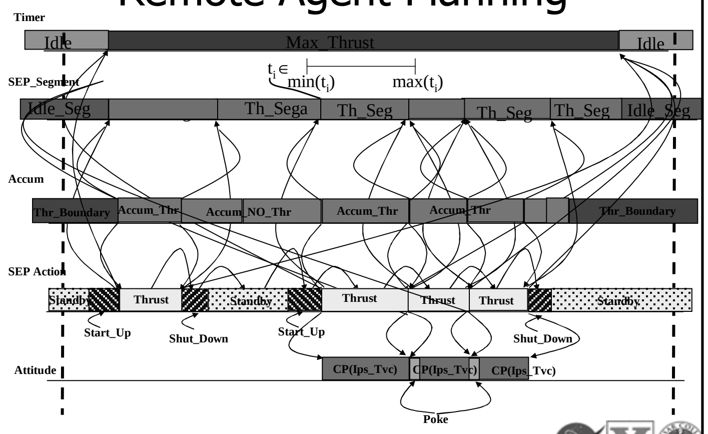


<?y, ?x> ∈ {<coffe, cpot>, <toast, pan>}

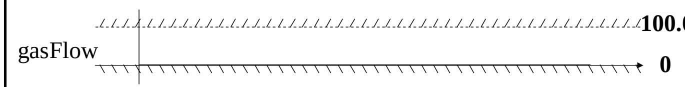
A View of Planning and Scheduling

- Planning primarily focuses on constructing a consistent evolution of the world (states and transitions)
- Scheduling almost entirely focuses on handling mutual exclusion and deadlines
- …but since the beginning planning was also addressing scheduling – flaws can be often seen as scheduling conflicts
- Graphplan and mutual exclusions implicitly brought this concept to the forefront

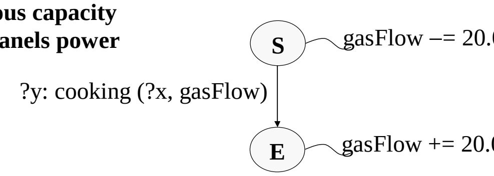
Remote Agent Planning



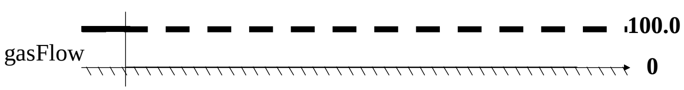
Resource Models



- **Discrete/continuous capacity**
- **Example: solar panels power**

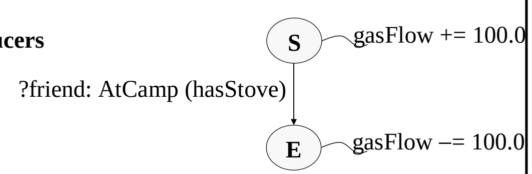


Resource Models

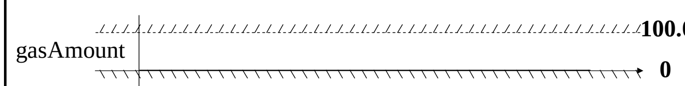


holds (Me, $Plan_s$,$Plan_e$, AtCamp(hasStove)

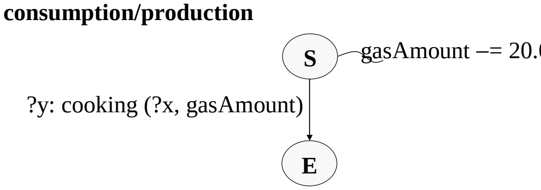
- **Resource producers**



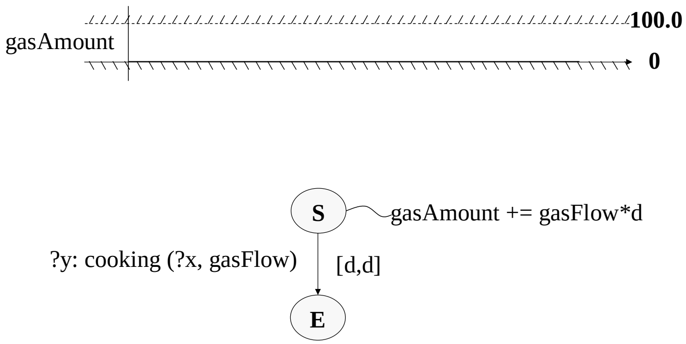
Resource Models



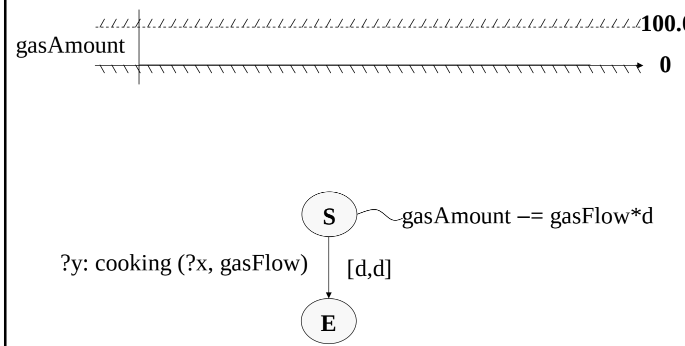
- **Permanent consumption/production**



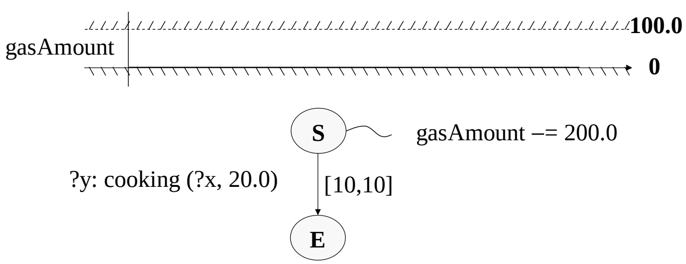
Insufficiency of Stepwise-Constant Resource Model



Insufficiency of Stepwise-Constant Resource Model

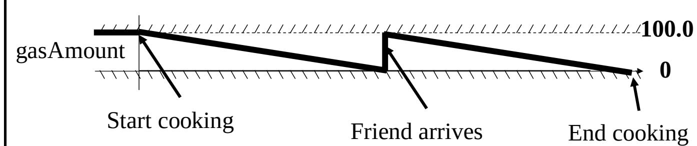


Insufficiency of Stepwise-Constant Resource Model

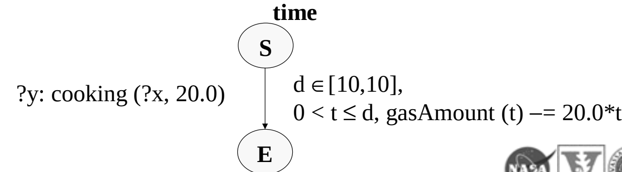


Cannot cook Texan barbeque in a California backcountry camp with limits on amount of storable fuel!

Insufficiency of Stepwise-Constant Resource Model



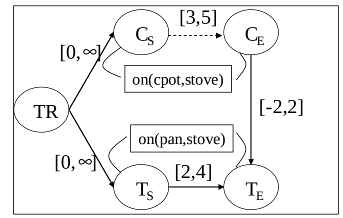
What counts is how the consumption rate accumulates over time



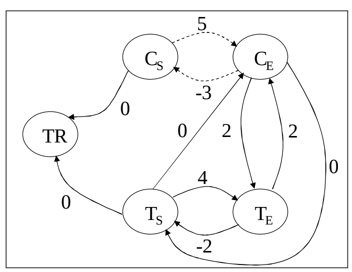
Flexibility in Plans/Schedules

- After a plan is executed, all variables (time, parameters) will be set to specific values
- Potential execution strategy: select the fixed values in advance and simply send them to the controlled device at the appropriate time.
- Worked reasonably well for spacecraft like Voyager.
- Not a lot is happening in the vacuum of space, though…
- Fundamental obstacles in the real world
  - Uncontrollability
  - Unobservability
- Two possible strategies
  - Flexible policies
  - "Fix values and repair"

How to Build a Flexible Breakfast Schedule

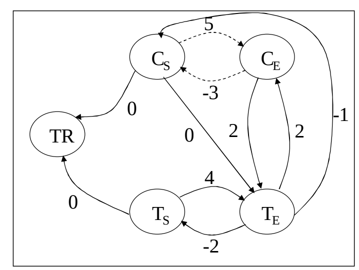


How to build a flexible schedule



Can we start making the toast after the coffee is brewed? **YES**

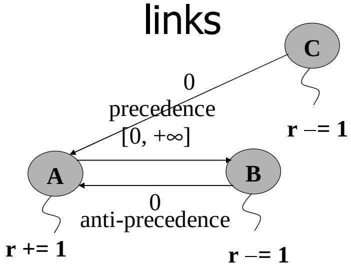
How to build a flexible schedule



Can we start brewing the coffee after the toast is ready? 

One interpretation of precedence links

117



- B→A anti-precedence creates a consumer/produced "coupling"
- B can rely on A to produce the resource it needs. Therefore, B will never cause a resource oversubscription
- With the addition of C→A, C and B compete to "match" with A
- Introducing "coupling" links and managing actual "matches" is what a flexible scheduling algorithm really does

PCP scheduling

118

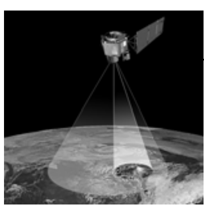
- [Cheung and Smith, 1997] use scratch propagation for unary capacity makespan optimization job-shop scheduling
- Scratch propagation can be done using Dijkstra algorithm from each end time to the start times on the same resource
- Scratch propagation cost: O(N^2logN) but can terminate early when all starts on same resource have been reached
- Incremental propagation achieves better speed
- Three cases for each pair of activities:
  - Inconsistency: no ordering is possible
  - Pruning: only one ordering is possible
  - Heuristic selection: if both orders are possible, select one according to a heuristic (e.g., maximum slack)
- Heuristic selection pair to resolve next is determined by a heuristic (e.g., minimum average slack)
- Search methods
  - Iterative Sampling with randomization

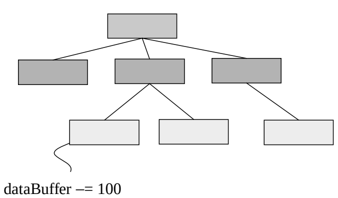
Fixed Time Scheduling and Execution Policies

119

[Chien et al. 2005] Automated Sciencecraft Experiment

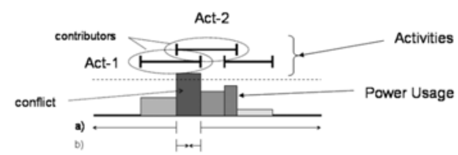
{PowerUp (Imager)} before
{s ∈ [10:00, 13:30], Image(lat, long, Mt.Etna)}





Fixed-time scheduling and execution policies

120



Constraint	Property that must hold for plan to be valid	Must always use less power than available
Conflict	Violation of a constraint	Current plan uses more power than available over (b)
Repair Method	Modification to plan that may remove conflict	Delete activity using power during conflict (b)
Repair Choice	Which activity to delete	Delete largest user?

Conflict Repair Methods

121

- Use a repair method to eliminate a conflict
- ASE uses a planner, not just a scheduler.
- Hence it is possible to generate new activities or select different task decompositions
- Repair methods
  - move an activity
  - delete an activity
  - add a new activity
  - detailing an activity
  - abstracting an activity
  - etc.

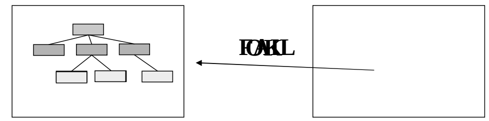
Add producer of resource. Not handled in classical scheduling

Chose different activity decomposition

From Planning to Execution The ideal situation

122

Repair plan using same method to generate it



Planner **Executive**

Comparison of Flexible and Fixed Policies (1)

123

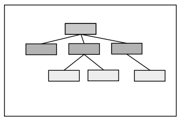
- Fixed policies
  - Pros
    - Simple and intuitive to implement
      - It is easier to think of heuristics based on resource profiles
    - More compact data structures
    - Less costly propagation
  - Cons
    - Plan does not give “declarative” measure of robustness
      - Execution repair is fundamental to robustness
    - A full plan repair process may be too expensive at execution time
      - ASE has only 4 MIPS available

Comparison of Flexible and Fixed Policies (2)

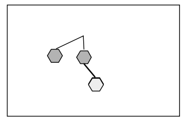
124

- Flexible policies
  - Pros
    - Plan guarantees measure of robustness
      - Flexible policies break less often
    - Execution time adjustments are intrinsically fast (propagation vs planning)
  - Cons
    - More complex
      - But complexity and computational expenses mostly affect off-line planning
    - Actual value of flexibility is only as good as the semantics of the representation
      - … and this is why you are taking this tutorial!

From Planning to Execution What actually happens on ASE



Planner

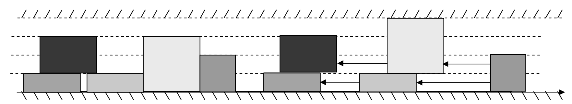


Executive

- Planner's detailed command expansion finds a "witness" to plan consistency
- If failures propagates at the highest activity level, this is a major problem
- Eliminating top-level failure requires careful tuning of "abstraction"
- Differences in internal planner/executive representations pushes toward conservatism to avoid mismatches and inconsistencies (it happened in Remote Agent…)
- Therefore, robustness is achieved at design time through careful modeling
- Flexible representations could help that design process

Building flexible policies from fixed time schedules

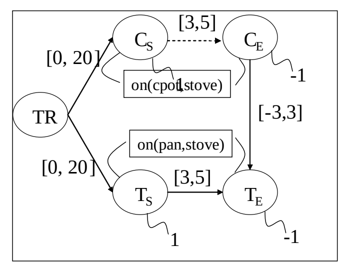
- Simple strategy for single capacity resources: simply keep the ordering constraints and uncommit the times from the fixed values
- Continuous/discrete capacity resources require the introduction of anti-precedence couplings between consumers and producers



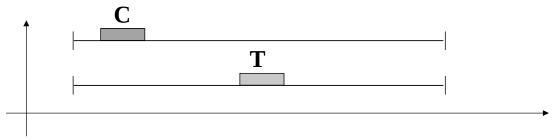
- [Policella et al, 2004] Transform fixed schedule into "chaining form" partial order
- Decompose multiple capacity resource into "virtual" single capacity resources and add couplings on chains

Probabilistic measures of resource contention

Contentious Breakfast

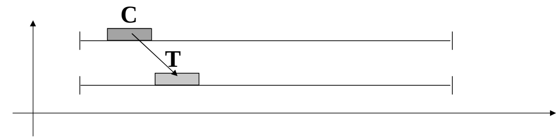


Time bounds and resource conflicts



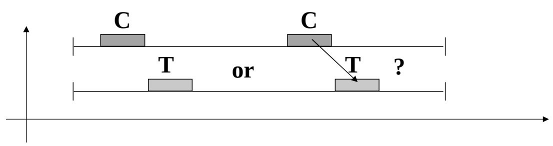
- Without further coordination, C and T are free to collide for the use of the stove
- The inclusion of anti-precedence links ("couplings" of producers to consumers) reduce and eventually eliminate the possibility of conflict

Time bounds and resource conflicts



- Without further coordination, C and T are free to collide for the use of the stove
- The inclusion of anti-precedence links ("couplings" of producers to consumers) reduce and eventually eliminate the possibility of conflict

Temporal Information for Contention Analysis



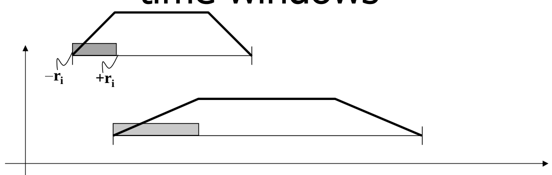
- Partial temporal information (e.g., time bounds for events) is insufficient to determine informative contention measures.
- More (full) temporal information is expensive to acquire and maintain
- There needs to be a balance between cost and utility of temporal/research inferences. Eventual value is in search improvement

Probabilistic Resource Contention

- Use probabilistic assumptions to generate time assignments given a temporal network
- Combine probabilistic assignments into contention statistics
- Use contention statistics as the basis for search heuristics
- Heuristic factors in probabilistic analysis:
  - Selection of problem sub-structure at the basis of statistics
  - Probabilistic assumptions on how activities request resource capacity
  - Variable/value ordering rules that use statistics

Probabilistic contention based on time windows

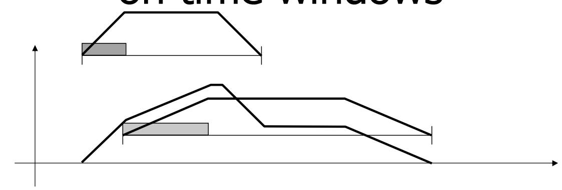
133



- [Beck & Fox 2000] Assumptions:
  - Fixed durations, consumption at start, same production at end
  - Uniform distribution of start times
  - Time bounds only
- Individual action demand inside the time bound:
  - $d_i(t) = \Sigma_{\max(est,\ t-dur) \leq \tau \leq \min(lft,\ t+dur)}\ r_i/(lft - est)$

Probabilistic contention based on time windows

134



- Aggregate demand = sum demand curves = expected value of instantaneous resource requests
- How to use it
  - Find maximum over all curves → maximum contention
  - Find pair with maximum demand at contention point that are not already ordered

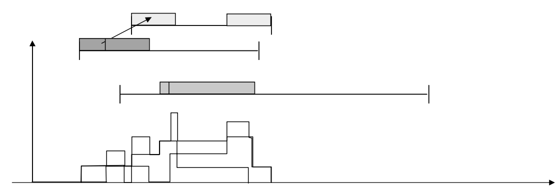
Another way to characterize conflicts

135

- Minimum Conflict Sets (MCS) [Laborie & Ghallab 1995]
- Minimum size sets of potentially conflicting activities with capacity request exceeding availability
- Order any activity pair in an MCS and eliminate one or more MCS
- No conflicts when there are no more MCSs
- Potentially an exponential number MCS but we only really care about ordering pairs of activities ($O(N^2)$) so there are very strong dominance rules

Probabilistic contention using precedence information

136



- Monte Carlo resource contention [Muscettola 1994]
- Consider all known temporal constraints
- Simulate a sample of executions ignoring resource contention
- Then compare expected resource request to resource limit to identify conflict areas
- Monte Carlo methods are also used in analysis of plan executions

Comparison of statistical contention measures

137

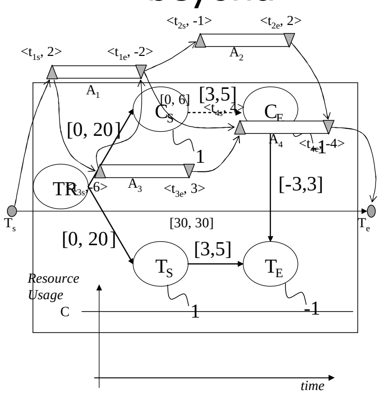
- Monte Carlo simulation is more informed
- Time-window method is less computationally expensive
  - Time windows: O(N) in time and space
  - Monte Carlo: with sample size S
    - O(S E) in time (if network is dispatchable)
    - O(S N) in space
- Monte Carlo method also biases sample depending on stochastic rule used to simulate the network
  - … but the rule can increase realism if it accurately describes execution conditions

138

Resource Usage Bounds

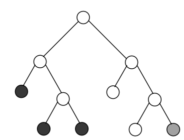
From breakfast to infinity and beyond

139



Search Guidance

140

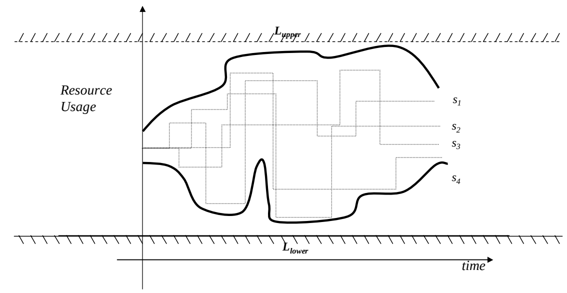


- The ability of detecting early that the flexible plan is resource/time inconsistent can save exponential amount of work
- Same for early detection of a solution

Need for exact resource bounds

- Statistical methods of resource contention give sufficient conditions to determine that a solution has not been achieved
- They cannot guarantee either inconsistency or achievement of a solution
- Exact resource bounds can

Resource Bounds

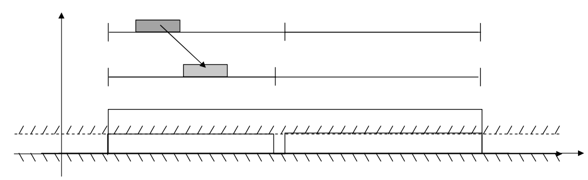


- Case 1: bounds always within limits → solution
- Case 2: bounds at least once outside the limit → inconsistency
- Case 3: otherwise → search

Bounds are costly

- In summary, bounds try to summarize the status of an exponential number of schedules
- As in the case of probabilistic measures, we can obtain different bounds depending of how much structural information on producer/consumer coupling we use
- The more information, the tighter the bound
- The more information, the more costly the bound

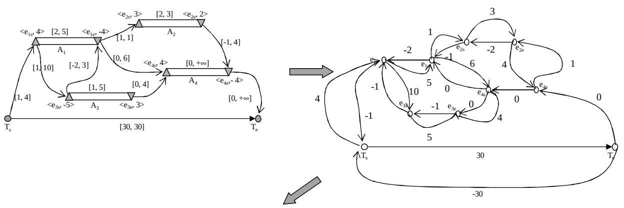
Least informative bounds

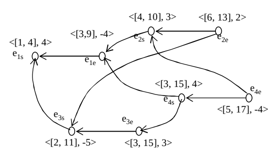


- Same situation as for statistical measures
- Bounds have to become non-overlapping to eliminate contention
- This cannot be done by the addition of precedence constraints alone if the schedule is very flexible
- Produced schedules are “flexible fixed time” schedules (i.e., constraint earliest and latest event times)

Temporal Information in Flexible Plans

145



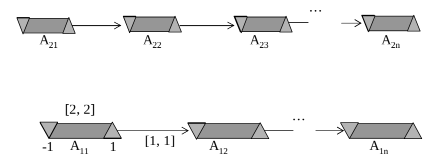


Anti-Precedence Graph

$[et(e), lt(e)] \Leftrightarrow et(e) = -|e\ T_s|$
$\wedge\ lt(e) = |T_s\ e|$
$|e_1 e_2| \leq 0 \Leftrightarrow e_1 \rightarrow \ldots \rightarrow e_2$

Balance Constraint Bounds

146



$L_{min,\leq} = -n - 1$ $\quad L_{min,\geq} = -n$

- Event centered: measure contention from the point of view of an event, not an absolute time reference
- Fundamental idea:
  - Make exact measures of consumption/production for predecessors and successors
  - Make worst case assumptions for all other events

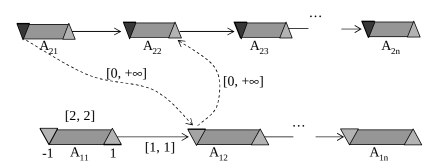
Cost of balance constraint bound

147

- Non incremental cost (compute the bound from scratch)
  - Find the anti-precedence network: O(NE) / O(NE + N^2log N)
  - Compute bounds from each event: O(NE) / O(N^2)
- Total cost (time propagation + bounds): O(NE) / O(NE + N^2 logN)
- Incremental propagation can reduce cost per each iteration
- Used succesfully for optimal scheduling in [Laborie 2001]

Looseness of Balance Constraint Bound

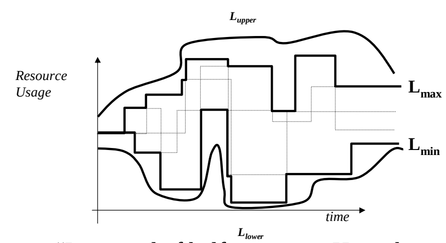
148



- If the two chains in the example operate on a resource with capacity 2, no constraint need to be added
- The Balance Constraint Bound however needs the addition of quite tight precedence constraints to detect a consistent solution
- The cause is the lack of consideration of the structure of the network not necessarily ordered with the event

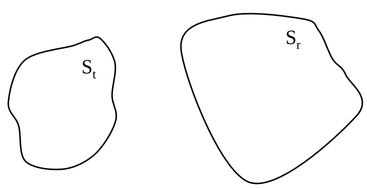
Resource Envelopes

Resource Envelope



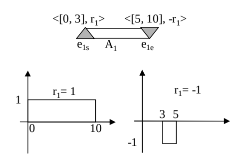
- Manager: “I am tired of half measures. How about giving me the tightest possible bounds?”
- Computer Scientist A: “Hmmm…I don’t know. It looks difficult. Remember the exponential number of schedules?
- Rocket Scientist B: “Aw, no problem. I’ll give you a fast polynomial algorithm for it …”

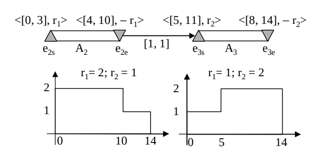
“WHAT?”



- $\exists s \in S_t \mid s \in S_r$ Scheduling problem NP-hard
- $\forall s \in S_t \mid s \in S_r$ Resource envelope looks hard(er)

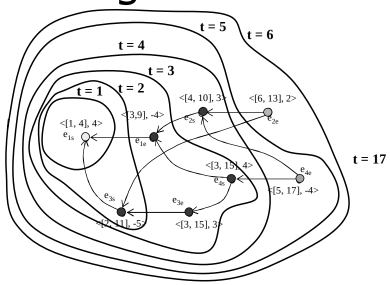
Resource Envelope Method Intuitive Description

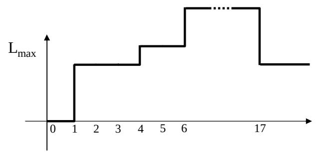




153

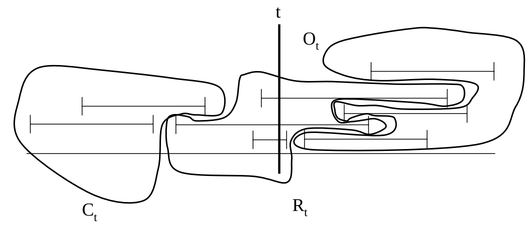
Building a full envelope

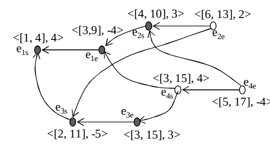




154

Pending Events





P_X = predecessor set of event set X

$P_{\{e2s,\, e3e\}} = \{e_{2s}, e_{3e}, e_{3s}, e_{1e}, e_{1s}\}$

P_{max} = predecessor set of maximum total weight

155

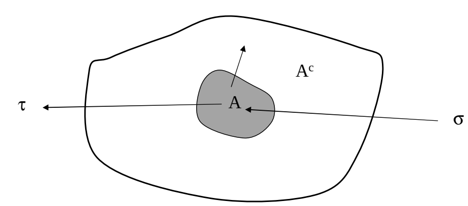
Key algorithm step

- "Find predecessor set within events that are pending at *t* that causes the maximum envelope increment"
- If we consider all "couplings" (due to anti-precedence links posted by the scheduler or due to original requirements), we can find sets of events that match. These will balance each other and cause no effect of the envelope level
- Events that do not match create a surplus or a deficit
- The amount of surplus (if any) represents the increase in resource envelope level.
- KEY PROBLEM: how do we compute the maximum match?

156

Maximum flows

$f(e_1, e_2) = -f(e_2, e_1)$	skew symmetry
$f(e_1, e_2) \le c(e_1, e_2)$	capacity constraint
$f(\{\sigma\}, A) = f(A, \{\tau\}) + f(A, A^c)$	flow conservation



$f(\{\sigma\}, A)$ = value of flow.
Maximize it .

Residual network
For each pair of nodes: $r_f(e_1, e_2) = c(e_1, e_2) - f(e_1, e_2)$

Augmenting path = path from σ to τ with positive residual
No augmenting path = flow is maximum

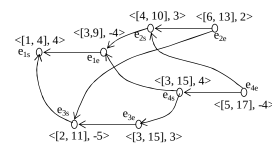
Maximum Flow Algorithms

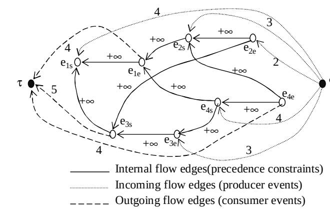
157

Algorithm	Time Complexity	Complexity Key
Labeling	*O(N E U)*	Total pushable flow
Capacity scaling	*O(NE logU)*	Total pushable flow
Successive shortest paths	*O(N^2E)*	Shortest distance to τ
Generic Preflow-push	*O(N^2E)*	Distance label
FIFO Preflow-push	*O(N^3)*	Distance label

Resource Increment Flow Network

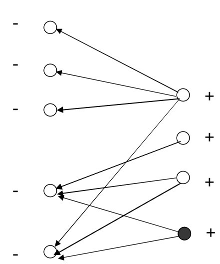
158





A simple P_{max} selection problem

159



Maximum Resource-Level Increment Predecessor Set

160

Theorem 1 : P_{max} = set of events that is reachable from σ in the residual network of a f_{max}

Theorem 2 : P_{max} is unique and has the minimal number of events

Separation Schedule and Separation Time

161

We know how to compute a P_{max} but …

… given a P_{max} is there a temporally consistent schedule and a time t_x such that all events in C_H and P_{max} are schedule at or before t_x and all events in P^c_{max} and O_H are scheduled after t_x?

Theorem 3: Yes!

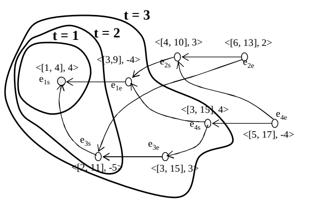
Maximum Resource Level and Resource Envelope

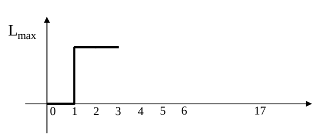
162

- Complete envelope profile [Muscettola, CP 2002]
  - $L_{max}(t) = \Delta(C_t) + \Delta(P_{max}(R_t))$
  - $P_{max}(R_t)$ and C_t change only at et(e) and lt(e).
  - Complexity: O(n O(maxflow(n, m, U)) + nm)
- Can we do better?

Building a full envelope

163



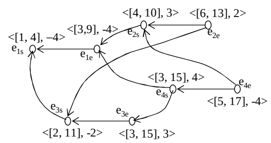


Staged Resource Envelope

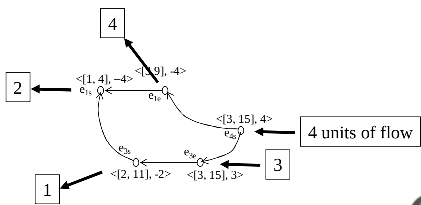
164

- Do not repeat flow operations on portion of the network that has already been used to compute envelope levels
- Deletion of flow due to elimination of consumers at time out do not cause perturbation to incremental flow
- We can reuse much (all?) of the flow computation at previous stages, increasing performance

How does it work?

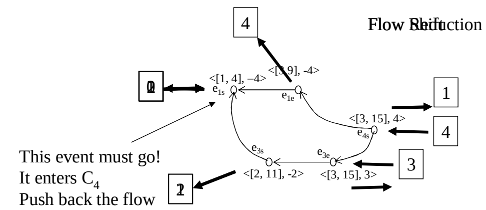


t = 3



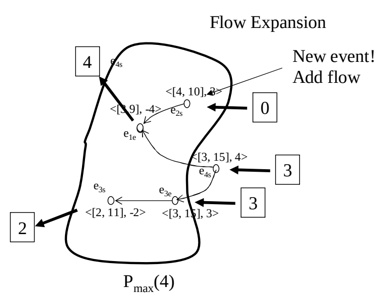
Flow Contraction

t = 4



Flow Expansion

t = 4



Recursive Equation

$L_{max}(t_i) = L_{max}(t_{i-1})$ +

Δ(E_1 = events in $P^c_{max}(t_{i-1})$ closed at time t_i) +

Δ(E_2 = events in P_{max} after Flow Contraction on remainder of E_1 elimination) +

Δ(E_3 = events in P_{max} after Flow Expansion on remainder of E_2 elimination)

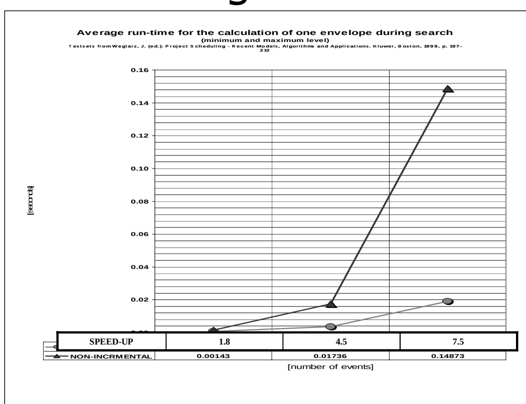
Complexity Analysis

- Look at all known Maximum Flow algorithms
- Identify complexity key
  - Total pushable flow (Labeling methods)
  - Shortest distance to τ (Successive Shortest Paths)
  - Distance label (Preflow-push methods)
- Show that complexity keys have same monotonic properties across multiple envelope stages that over a computation of maximum flow over entire network.
- Hence, complexity is O(Maxflow(n, m, U))

Summarized excerpt from helpful comments of friendly ICAPS 2004 reviewers

"Sure, nice theory. But theory ain't much. Where are the empirical results, eh?"

Empirical speedup of staged algorithm

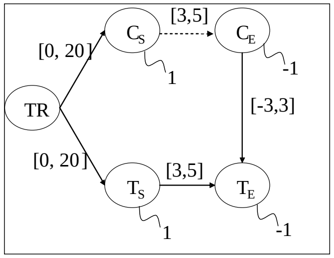


Envelope scheduling so far

- [Policella et al. 2004]
- Non-backtrack, non-randomized commitment procedure
  - either it finds a schedule at the first trial or it never will
- Two kinds of contention profiles tested
  - Resource envelopes
  - Earliest start profiles – profiles obtained by schedule executing all activities as early as possible
- Methods using earliest start profiles perform better on tested benchmark
- Open problem: is there other structural information in the envelopes that can be useful outside of contention identification?

One More Breakfast

THE END



References I

The literature on temporal reasoning and planning is extensive. Here we list only some initial sources for ideas and, where avaiable, survey papers that provide detail and additional references; these survey papers are in **boldface and color**.

<u>Constraint-Satisfaction Processing:</u>

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

<u>Qualitative Models of Time:</u>

- J. Allen, "A General Model of Action and Time," *Artificial Intelligence* 23(2), 1984.
- M. Vilain and H. Kautz, "Constraint Propogation Algorithms for Temporal Reasoning," *Proc. Of the 5th National Conference on Artificial Intelligence (AAAI)*, 1986.
- **Chapter 12 of Dechter, *Constraint Processing*, (see above).**

<u>Simple and Disjunctive Temporal Problems:</u>

- R. Dechter, I. Meiri, and J. Pearl, "Temporal Constraint Networks," *Artificial Intelligence*, 49(1-3), 1991.
- **E. Schwalb and R. Dechter, "Processing Temporal Constraint Networks," *Artificial Intelligence* 93(1-2), 1997.**
- K. Stergiou and M. Kourbarakis, "Backtracking Algorithms for Disjunctions of Temporal Constraints," *Artificial Intelligence* 120(1), 2000.
- A. Oddi and A. Cesta, "Incremental Forward Checking for the Disjunctive Temporal Problem," *Proc. Of the 14th European Conference on Artificial Intelligence (ECAI)*, 2000.
- I. Tsamardinos and M. E. Pollack, "Efficient Solution Techniques for Disjunctive Temporal Reasoning Problems," *Artificial Intelligence*, 2003.
- A. Armando, C. Castellini, E. Giunchiglia, and M. Maratea, "A SAT-Based Decision Procedure for the Boolean Combination of Difference Constraints," *Proc. Of the 7th International Conference on Theory and Applications of Satisfiability Testing*, 2004.

References II

Dispatch of Disjunctive Temporal Problems:

- I. Tsamardinos, M. E. Pollack, and P. Ganchev, "Flexible Dispatch of Disjunctive Temporal Problems," *Proc. Of the 6th European Conference on Planning (ECP)*, 2001.

Unobservability and Uncontrollability:

- T. Vidal and H. Fargier. "Handling contingency in temporal constraint networks: from consistency to controllabilities" *Journal of Experimental and Theoretical Artificial Intelligence,* 11(1):23-45, 1999.
- P. Morris, N. Muscettola, and T. Vidal, "Dynamic Control of Plans with Temporal Uncertainty," *Proc. Of the 7th International Joint Conference on Artificial Intelligence,* 2001.
- I. Tsamardinos and M. E. Pollack, "CTP: A New Constraint-Based Formalism for Conditional, Temporal Planning," *Constraints* 8, 2003.

Planning with Temporal Constraints:

- M. Ghallab and H. Laruelle, "Representation and Control in IxTeT, a Temporal Planner," *Proc. 2nd Intl. Conference on AI Planning Systems (AIPS),* 1994.
- N. Muscettola, "HSTS: Integrating Planning and Scheduling," in *Intelligent Scheduling,* Monte Zweben & Mark Fox eds., Morgan Kaufmann, 1994.
- **Chapter 12 of Dechter, *Constraint Processing*, (see above).**
- **Chapters 13 and 14 of M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*, Elsevier, 2004**
- **D. E. Smith, J. Frank, and A. Jonsson, "Bridging the Gap between Planning and Scheduling," *The Knowledge Engineering Review,* 15, 2000.**
- **J. Frank and A. Jonsson, "Constraint-Based Attribute and Interval Planning," *Constraints* 8, 2003.**

References III

Resource Constraint Reasoning: Scheduling:

- Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian C. Williams. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1/2), August 1998
- Cheng, C. and S.F. Smith, *Applying Constraint Satisfaction Techniques to Job-Shop Scheduling (The Long Version)*, Robotics Institute Technical Report CMU-RI-TR-95-03, January, 1995. [Published in *Annals of Operations Research*, Vol. 70, Special Issue on Scheduling: Theory and Practice, 1997.]
- S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, D. Mandel, S. Frye, B. Trout, S. Shulman, D. Boyer. "Using Autonomy Flight Software to Improve Science Return on Earth Observing One", *Journal of Aerospace Computing, Information, and Communication* . April 2005 + PDF
- N. Policella, A. Oddi, S.F. Smith and A. Cesta. "Generating Robust Partial Order Schedules" In *Proc of CP 2004*, Lecture Notes on Computer Science (LNCS) Vol. 3258, pp. 496-511, M. Wallace (Ed.), Springer, 2004.

Probabilistic Measures of Resource Contention:

- Beck, J.C. & Fox, M.S., Constraint Directed Techniques for Scheduling with Alternative Activities, *Artificial Intelligence*, 121(1-2), 211-250, 2000.
- Nicola Muscettola: On the Utility of Bottleneck Reasoning for Scheduling. AAAI 1994: 1105-1110

Resource Usage Bounds:

- **Philippe Laborie "Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results", *Artificial Intelligence,* 143(2), pp. 151-188, 2003**

Resource Envelopes:

- **R.K.Ahuja, T.L.Magnanti, J.B.Orlin. *Network Flows,* Prentice Hall, 1993.**
- N. Muscettola "Computing the envelope of Stepwise-Constant Resource Allocations", *Proc. of CP 2002*, Ithaca, NY, 2002.
- N. Muscettola "Incremental Maximum Flows for Fast Envelope Computation", *Proceedings of the 14th International Conference on Automated Planning & Scheduling, ICAPS04, Whistler, British Columbia, Canada, 2004.*
- N. Policella, S.F. Smith, A. Cesta and A. Oddi (2004). "Generating Robust Schedules through Temporal Flexibility" In , *Proceedings of the 14th International Conference on Automated Planning & Scheduling, ICAPS04*, Whistler, British Columbia, Canada, 2004.