ICAPS 2005
Monterey, California

ICAPS05

TU1

Tutorial on Temporal and Resource Reasoning for Planning, Scheduling and Execution

Nicola Muscettola
NASA Ames Research Center, USA

Martha Pollack
University of Michigan, USA

ICAPS 2005
Monterey, California, USA
June 6-10, 2005

Tutorial on Temporal and Resource Reasoning for Planning, Scheduling and Execution

Nicola Muscettola
NASA Ames Research Center, USA

Martha Pollack
University of Michigan, USA

Tutorial on Temporal and Resource Reasoning for Planning, Scheduling and Execution

Table of contents

Tutorial on Temporal and Resource Reasoning for Planning, Scheduling and Execution

Preface

Planning and scheduling algorithms are increasingly guiding autonomous systems that interact with the environment and with humans in the real world. Without effective management of time and resources these autonomous systems cannot guarantee safe and efficient operations over a long period of time. In this tutorial we review basic and advanced topics in time and resource constraint reasoning and their applications to planning, scheduling and execution. The emphasis on plan execution is increasingly important as planning moves from the laboratory to real applications. Significant CPU and memory limitations during plan execution provide a strong driver for the design of efficient algorithms. Several such algorithms will be presented in this tutorial together with their justification from applications such as space exploration, health care systems, military systems and manufacturing. The tutorial will present a comprehensive review of current temporal and resource constraint-based formalisms, their motivation, their propagation algorithms and their use in planning, scheduling and execution systems.

Instructors

- *Nicola Muscettola, NASA Ames*
- *Martha E. Pollack, University of Michigan*

ICAPS 2005 Tutorial:

Temporal and Resource Reasoning for Planning, Scheduling, and Execution

Nicola Muscettola, NASA Ames
Martha E. Pollack, Univ. of Michigan

Real-World Planning and Execution

Space Facility Crew Activity Planning

They went on strike!

- 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

MAPGEN in Surface Operations

Oct '00-Sept '01	Oct '01-Sept'02	Oct'02-Sept'03	Oct'03-April'04

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

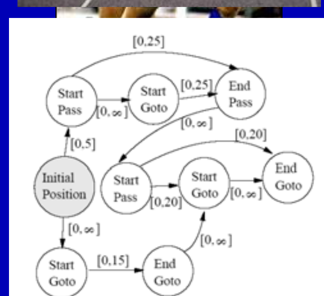
Robust Task Execution for Long Traverse Rovers

- 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!



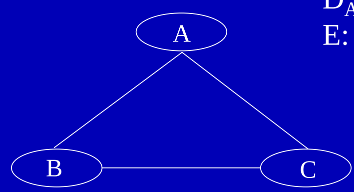
Issues in Temporal Planning and Execution

- 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: {~~R~~, B} D_C : {~~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
3. Resource reasoning

Qualitative Temporal Models

Outline

1. Qualitative Temporal Models
2. Representing and Solving Simple Temporal Problems
3. Dispatching Plans Modeled as STPs
4. Representing and Solving Disjunctive Temporal Problems
5. Dispatching DTPs
6. Generating Temporal Plans
7. Adding Uncertainty: Temporal and Causal

Then on to resources. . .

Interval Algebra

X before Y

X equal Y

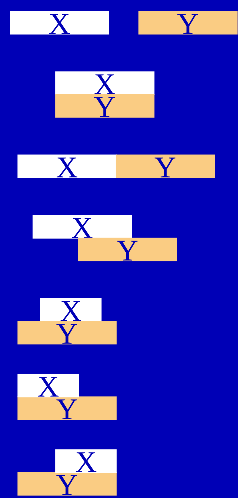
X meets Y

X overlaps Y

X during Y

X starts Y

X finishes Y



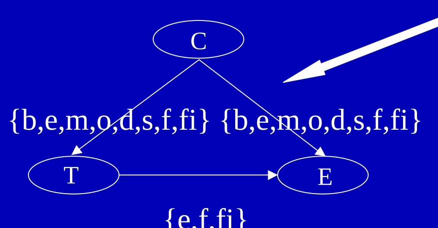
- With inverses: 13 primitive relations
- Complete (can describe all possible relations between intervals)
- Construct compound relations:
- "Y ends sometime after X":

$X\ b\ Y \vee X\ m\ Y \vee X\ o\ Y \vee X\ d\ Y \vee X\ s\ Y$

The Breakfast Plan

Prepare coffee, toast, and eggs. Have coffee ready no later than the rest of the meal, and have toast and eggs ready at the same time.

Assignments to *pairs* of variables, e.g. C,E ← o



Ternary constraints rule out some possibilities, e.g.

T,E ← e

C,T ← b

C,E ← o

Reasoning with the Interval Algebra

- Model the ternary constraints with a composition table; use to check path-consistency

	b	m	o	. . .
b	b	b	b	
m	b	b	b	
o	b	b	b,m,o	
. . .				

- Reasoning tasks
  - Check consistency
  - Find a solution
- Both tasks are NP hard
  - Path consistency not sufficient

Point Algebra

$P < Q$

$P = Q$

$P > A$

- Now model intervals with 2 points (start and end)
- Construct 8 compound relations

  "(Interval) Y ends no later than (interval) X":

  $Y_e > X_e \vee Y_e = X_e$
- Can check consistency and find solutions in polynomial time
- But loss of expressive power: Can't represent all IA relations

  $X \{b,bi\} Y \equiv X_e < Y_s \vee Y_e < X_s$ ⬅ Not binary!

Real Plans often have Quantitative Constraints

- US NINDS Guidelines for Treatment of Potential Stroke (Thrombolytic) Patient

ACTION	TARGET DURATION
Hospital door to doctor	10 minutes
Door to neurological expert	15 minutes
Door to CT scan completion	25 minutes
Door to CT scan interpretation	45 minutes
Depending on test results, door to treatment	60 minutes
Depending on test results, admission to monitored bed	3 hours

Real Plans often have Quantitative Constraints

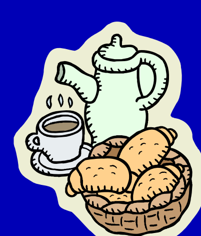
- Typical Plan for an Autominder User

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 1:00, and between 3:00 and 5:00
Depending on pulse, take meds	At end of check pulse

Simple Temporal Problems

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.



Temporal Constraint Problems

- Family of constraint-satisfaction problems (CSPs), $<V,E>$ where
  - V = events
  - E = interval-based constraints
- The domains D are left implicit: real numbers or integers
- Members of the family are defined by the form of the constraints

Simple Temporal Problems

- In a Simple Temporal Problem (STP) $<V,E,>$, the constraints have the form $y - x \le u$, where $x, y \in V$, and $u \in \Re$.
- W.l.o.g. assume $u \in Z$.

The Breakfast Plan as an STP

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.

Variables: TR, C_S, C_E, T_S, T_E

Constraints:

$3 \leq C_E - C_S \leq 5$

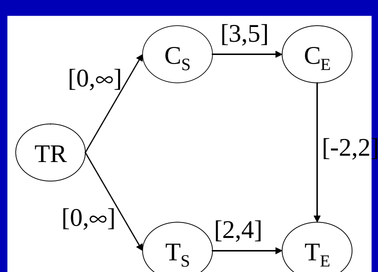
$2 \leq T_E - T_S \leq 4$

$-2 \leq C_E - T_E \leq 2$

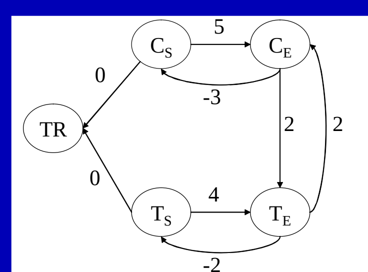
$0 \leq C_S - TR \leq \infty$

$0 \leq T_S - TR \leq \infty$

Graphical Representations of STPs



Simple Temporal Network (STN)

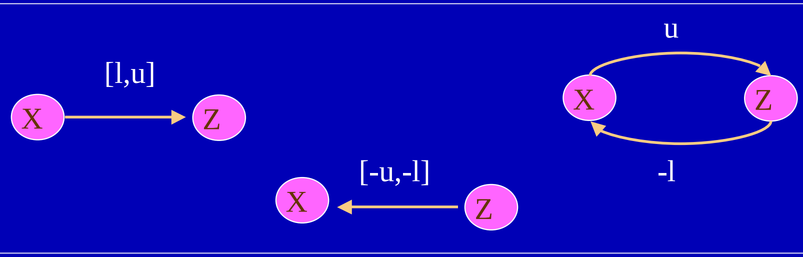


Distance Graph

Equivalences

The following are equivalent:

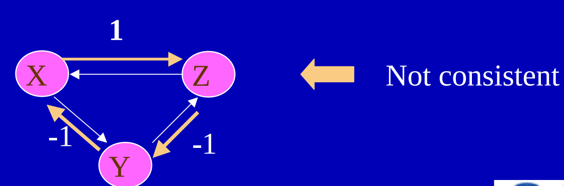
$l \leq z\text{-}x \leq u$ $\qquad$ $z\text{-}x \leq u \wedge x\text{-}z \leq -l$



Be careful—the following are *not:*

Solving STPs

- A *solution* to an STP <V,E> is an assignment of a time point to each variable in V s.t. all the constraints in E are satisfied.
- An STP is *consistent* (has a solution) iff its distance graph contains no negative cycles.



Negative Cycles

Given cycle $X = i_0, i_i, \ldots I_n = X$

Constraints : $i_1 - X \leq b_{i0,i1}$

$i_2 - i_i \leq b_{i1,i2}$

$\ldots$

$X - i_{n-1} \leq b_{in-1,in}$

$$\sum_{j=1}^{n} b_{ij-1,\,ij} = d_{XX}$$

Sum up the inequalities:

$X - X = 0 \leq d_{XX},$ i.e., $d_{xx} \geq 0$

So if $d_{xx} < 0$, have a contradiction

Computing Consistency

- Can thus check the consistency of an STP this in polynomial time, using an all-pairs shortest path algorithms (e.g., Floyd-Warshall)
- Consistent iff 0's along the main diagonal
- The value of the shortest path from X to Y is called the *distance* from X to Y, written d_{XY}
- Graphical form of the APSP matrix is called the *d-graph*

Floyd-Warshall Algorithm

Given a graph G,

W = adjacency-matrix(G), n = size(G)

$D^{(0)} = W$

For k = 1 to n

For i = 1 to n

For j = 1 to n

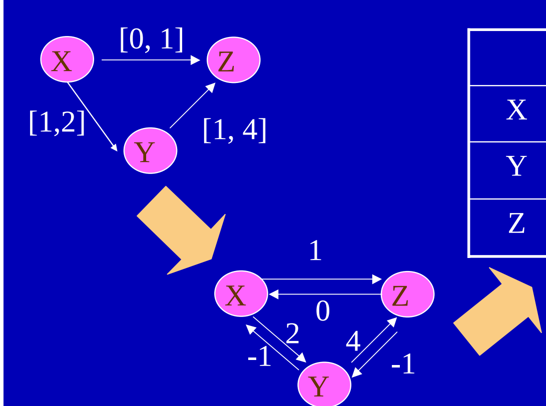
$D^{(k)}_{i,j} = \min(D^{(k-1)}_{i,j},\ D^{(k-1)}_{i,k} + D^{(k-1)}_{k,j})$

Return $D^{(n)}$

Paths from i to j with intermediate nodes from 1 to k-1

Paths from i to j with intermediate nodes from 1 to k

An Example

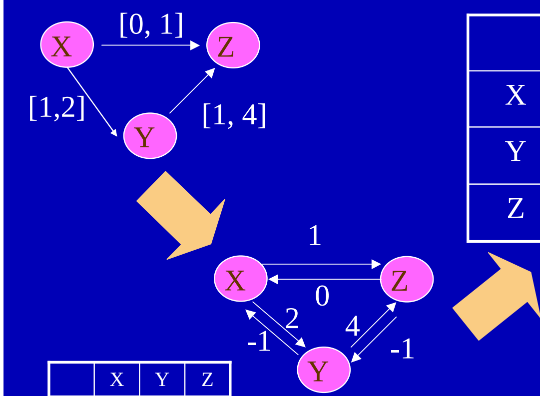


$D^{(0)}$

	X	Y	Z
X	0	2	1
Y	-1	0	4
Z	0	-1	0

An Example

D(1), paths through X

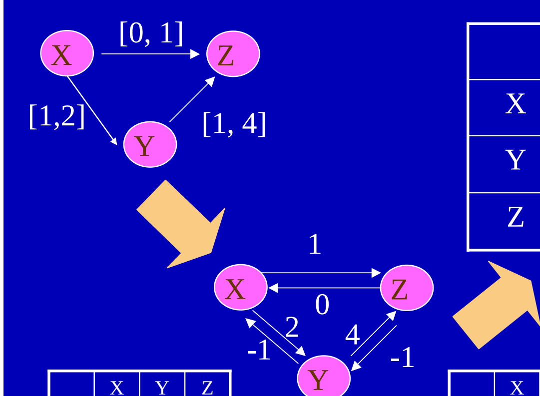


	X	Y	Z
X	0	2	1
Y	-1	0	4
Z	0	-1	0

	X	Y	Z
X	0	2	1
Y	-1	0	0
Z	0	-1	0

An Example

D(2), paths through X,Y

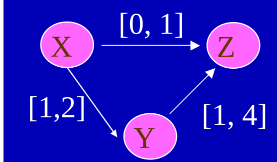


	X	Y	Z
X	0	2	1
Y	-1	0	4
Z	0	-1	0

	X	Y	Z
X	0	2	1
Y	-1	0	0
Z	0	-1	0

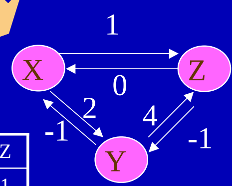
	X	Y	Z
X	0	2	1
Y	-1	0	0
Z	-2	-1	0

An Example



D(3), paths through X,Y,Z

	X	Y	Z
X	**-1**		
Y			
Z			

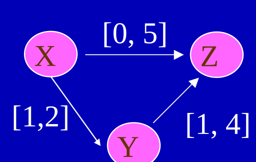


	X	Y	Z
X	0	2	1
Y	-1	0	4
Z	0	-1	0

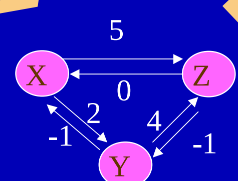
	X	Y	Z
X	0	2	1
Y	-1	0	**0**
Z	0	-1	0

	X	Y	Z
X	0	2	1
Y	-1	0	0
Z	**-2**	-1	0

Another Example



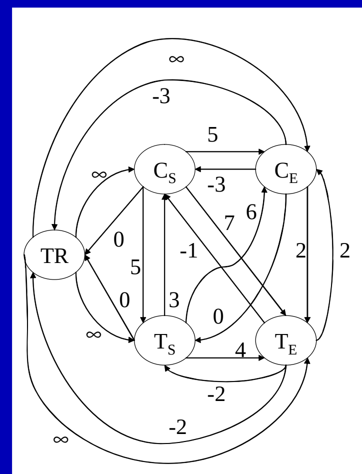
	X	Y	Z
X	**0**	2	5
Y	-1	**0**	4
Z	-2	-1	**0**



D-Graph for the Breakfast Plan

	TR	C_S	C_E	T_S	T_E
TR	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

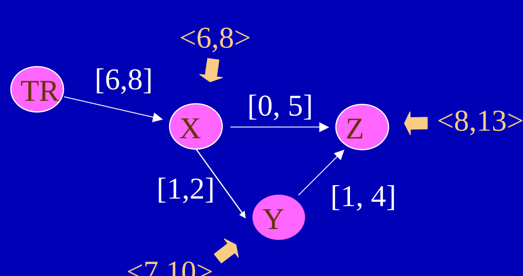
APSP Matrix



d-graph

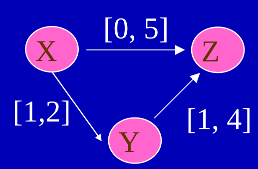
TR's and TW's

- Use a *Temporal Reference Point (TR)* to specify absolute clock times
- Compute the *Time Window (TW)* for every event *e*
  - Minimal distance to/from *TR* $(d_{TR,X}, d_{X,TR})$



Decomposability

- An STP is *decomposable* if every locally consistent assignment can be extended to a solution.



X ← 0, satisfying Constraints({X})

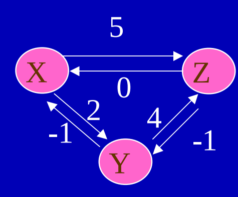
Z ← 0, satisfying Constraints({X,Z})

No way to extend with an assignment to Y –
not decomposable

- The all-pairs, shortest path graph (the d-graph) for any STP is decomposable.

Generating STP Solutions

- Can "read off" solutions from the d-graph



	X	Y	Z
X	**0**	2	5
Y	-1	**0**	4
Z	-2	-1	**0**

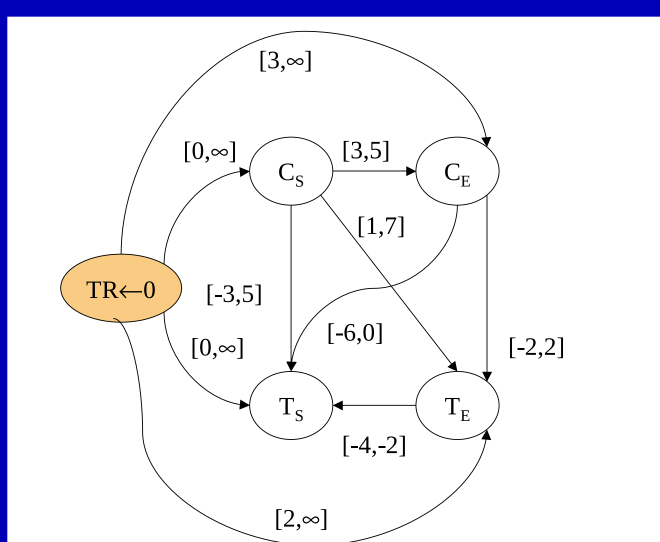
Immediate Solutions:
{x = 0,y = 2, z = 5}
{x = -1, y = 0, z = 4}
{x = -2, y = -1, z = 0}

More generally…

- Construct the d-graph and order the nodes, v_0, . . . v_n (usually v_0 = TR)
- Select a value $x_0 \in TW(v_0)$
- Solution = $\{v_0 \leftarrow x_0\}$
- For k = 2 to n
  - Propagate: $TW(v_k) = \cap_{i=1}^{k-1} (x_i + [-d_{k,i}, d_{i,k}])$
  - Select $x_k \in TW(v_k)$
  - Solution = Solution $\cup \{v_k \leftarrow x_k\}$

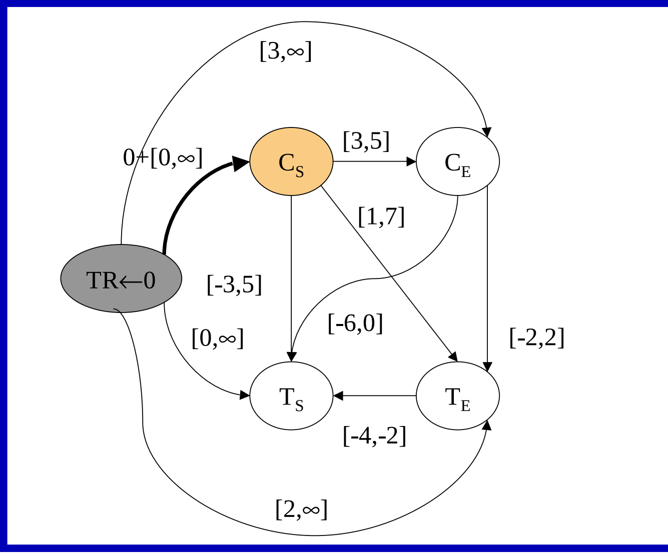
Exploit decomposability

Solving the Breakfast STP I



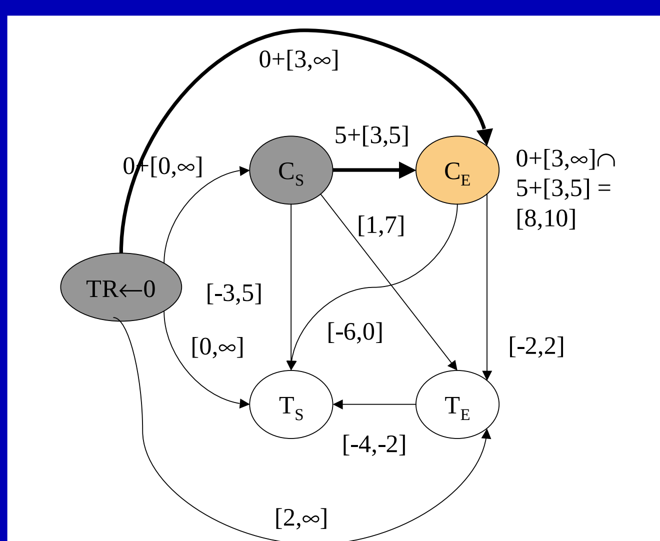
TR ← 0

Solving the Breakfast STP II



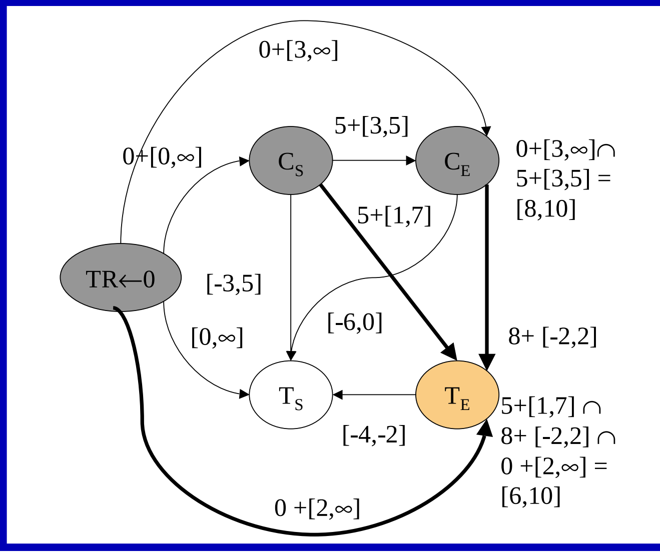
TR ← 0
CS ← 5

Solving the Breakfast STP III



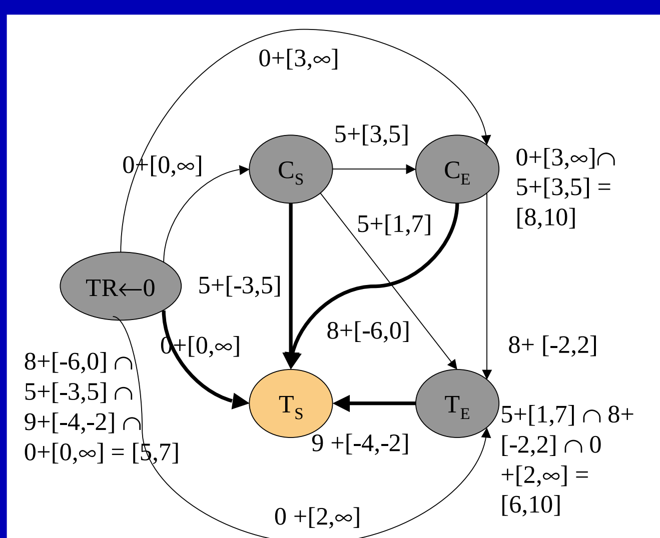
TR ← 0
C_S ← 5
C_E ← 8

Solving the Breakfast STP IV



TR ← 0
C_S ← 5
C_E ← 8
T_E ← 9

Solving the Breakfast STP V



TR ← 0
C_S ← 5
C_E ← 8
T_E ← 9
T_S ← 6

Plan Dispatch With STPs

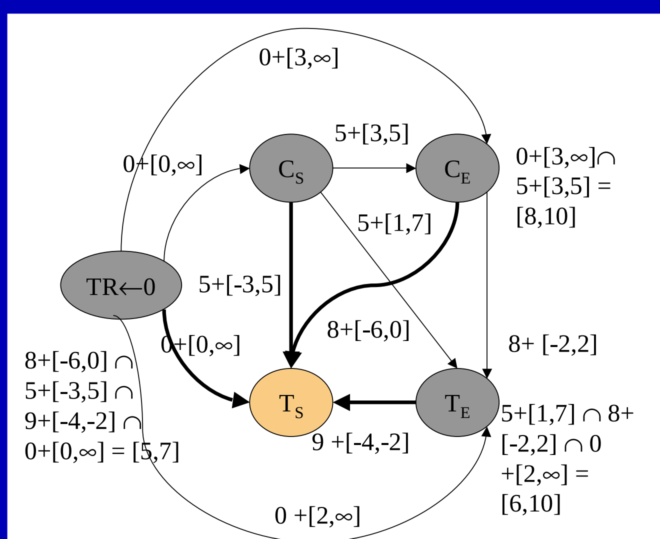
The Dispatch Problem

- Given a (set of) plan(s) with temporal constraints, decide when to execute each action so as to ensure that the constraints are satisfied.

Naïve Dispatch Algorithm

- Use the STP solution algorithm to assign a value to a variable.
- Wait until that time occurs.
- Dispatch the event associated with that variable.

Solving the Breakfast STP



$TR \leftarrow 0$
$C_S \leftarrow 5$
$C_E \leftarrow 8$
$T_E \leftarrow 9$
$T_S \leftarrow 6$

Naïve Dispatch Algorithm

TR $\leftarrow$ 0
Start at 8am

$C_S \leftarrow 5$
Next, start coffee at 8:05

$C_E \leftarrow 8$
Pour the coffee at 8:08

$T_E \leftarrow 9$
Pop the toast at 8:09

$T_S \leftarrow 6$
Start the toast at 8:06

Off-Line Dispatch

- Find a solution to the STP off-line
- Sort the variables in increasing temporal order
- Dispatch as each event as it “comes due”

Find solution TR $\leftarrow$ 0, $C_S \leftarrow 5$, $C_E \leftarrow 8$, $T_E \leftarrow 9$, $T_S \leftarrow 6$
Sort: <TR, C_S, T_S, C_E, T_E>
Then dispatch in order

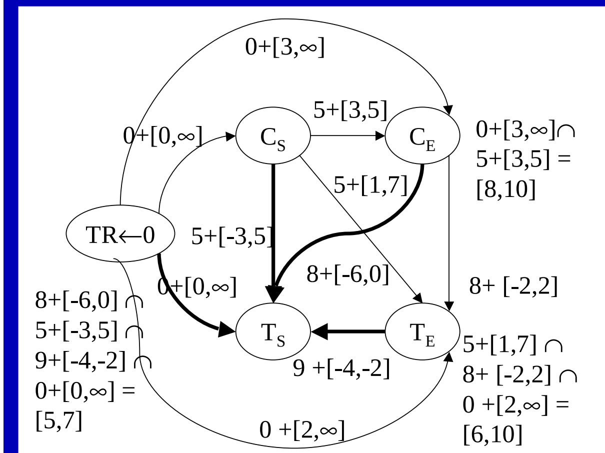
On-Line Dispatch

- Off-line dispatch is inflexible; can't handle "uncontrollable" events
- Key idea for on-line dispatch: only dispatch events that are
  - *Live* (it's currently within their time window), and
  - *Enabled* (all events that are constrained to occur earlier have already been dispatched)
    - Easy to recognize when Y must precede X: $D_{XY} < 0$, i.e., there's a negative edge starting at X

On-Line Dispatch Algorithm

1. Compute the d-graph for the given STP
2. A ← {*x* | *x* has no outgoing negative edges} [*x* is initially enabled]
3. Pick and remove an event *e* from A such that *now* ∈ TW(E)
4. S ← S ∪ {*e*}
5. Dispatch e and set execution-time(e) ← *now*
6. Propagate this assignment *to the neighbors* of *e*
7. A ← A ∪ {*x* | all negative edges starting at *x* have destinations already in S} [*x* is enabled.]
8. Wait until *now* has advanced to some time between the minimum lower bound of a time window for a member of A and the minimum upper bound of a time window for a member of A.
9. Loop to (2) until every event is in S.

On-Line Dispatch of Breakfast



TR ← 0
Start at 8am

C_S ← 5
Next, start coffee at 8:05

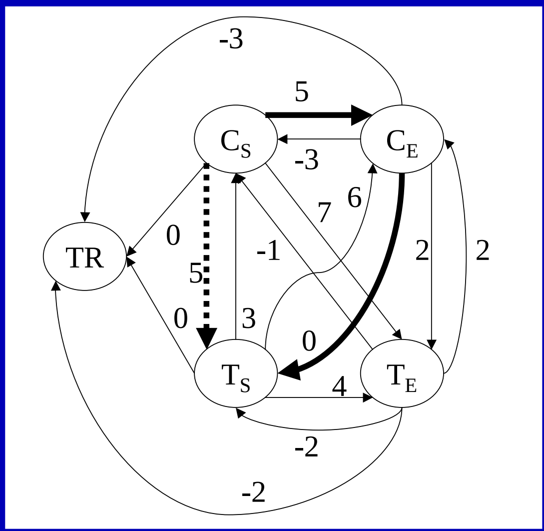
C_E ← 8
Pour the coffee at 8:08

Can't dispatch T_E next, since it isn't enabled!

Improving Efficiency

- Some edges in the d-graph are *dominated*, and can be removed
- Triangle Rule: Edge AC is dominated if there is another node B such that:
  $\{ |AB| + |BC| = |AC| \} \wedge \{ |AB| < 0 \ \vee \ |BC| \geq 0 \}$

A Dominated Edge



Edge C_ST_S is dominated by C_SC_E and C_ET_S

Increasing Efficiency

- Can remove all the dominated edges off-line in $O(n^3)$ time, to create the *minimal equivalent dispatchable (MED)* network
- Dispatch is still $O(n^2)$ since in the worst case no edges may be removed
- But in practice may obtain significant speedup: NASA Remote Agent domain, 40-60% of original edges pruned

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

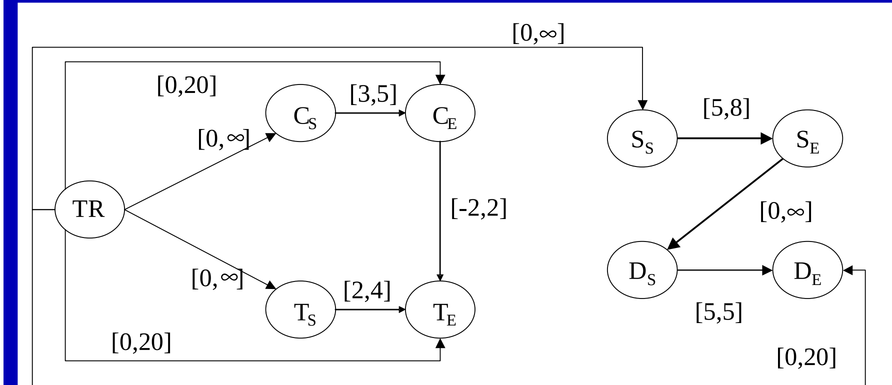
Non-overlap:
$L_E - P_S \leq 0 \vee$
$M_E - L_S \leq 0$

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

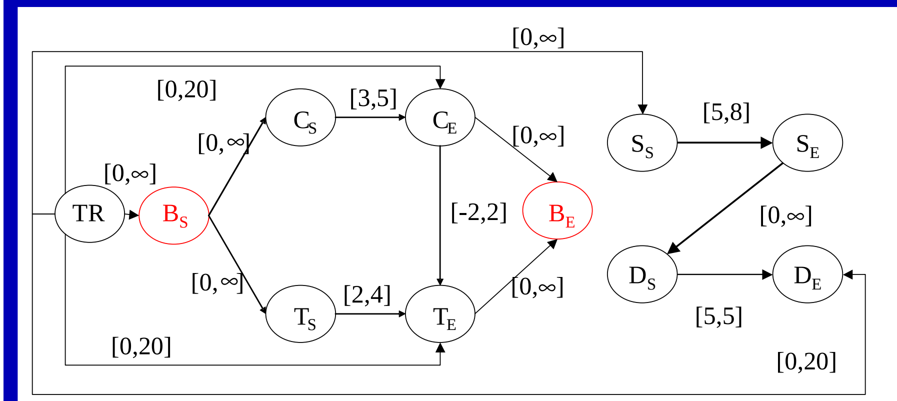
Prepare coffee and toast. Shower and dress.



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

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

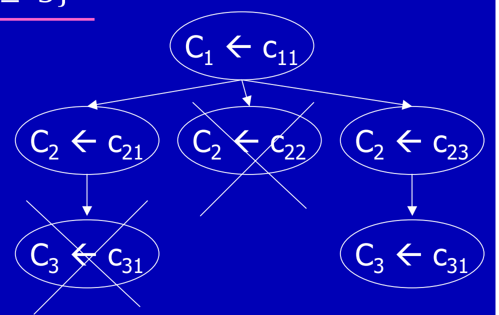
- One-Level Approach
  - Direct assignment of times to DTP variables.
  - Limitations: difficult to deal with infinite domains; produces overconstrained solution
- Two-Level Approach
  - Construct a meta-level CSP
  - Variables: DTP constraints
  - Domains: Disjuncts from DTP constraints.
  - Constraints: 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:
$C_1 \leftarrow c_{11}, C_2 \leftarrow c_{23}, C_3 \leftarrow c_{31}$

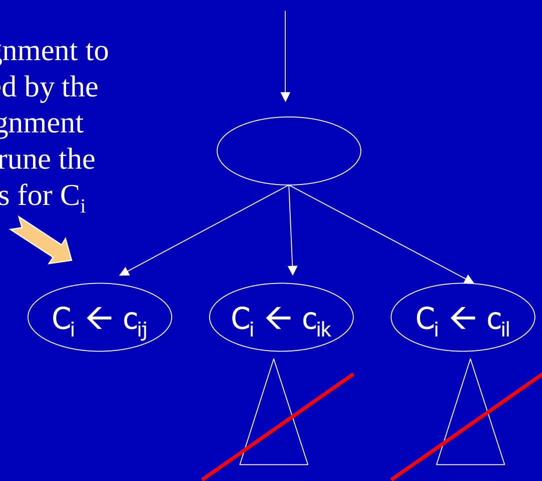
One exact solution:
$\{x = 0, y = 1, z = 2, w = 12\}$

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



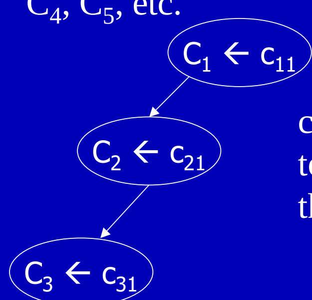
Removal of Subsumed Variables

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

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

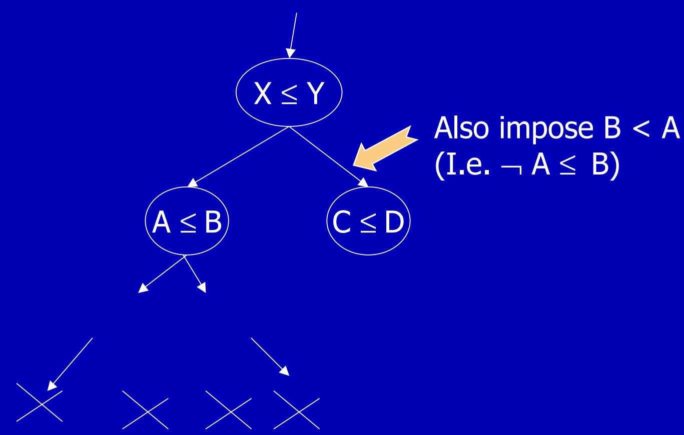
$C_3 : \{c_{31} : y - z \leq 15\} \vee \{c_{32} : z - v \leq 10\} \vee \ldots$

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



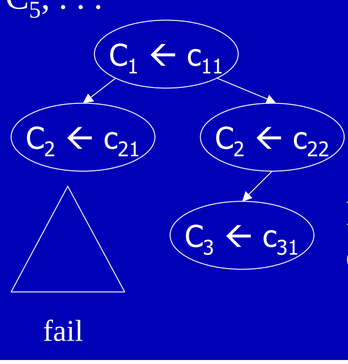
Semantic Branching

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

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

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

$C_4, C_5, \ldots$



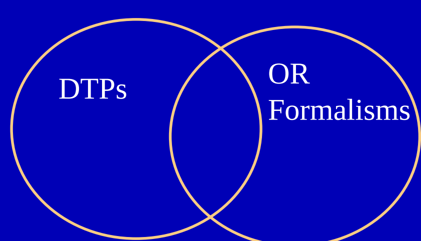
Add $\neg c_{21}$: $x\text{-}z > 3$

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

So, how fast?

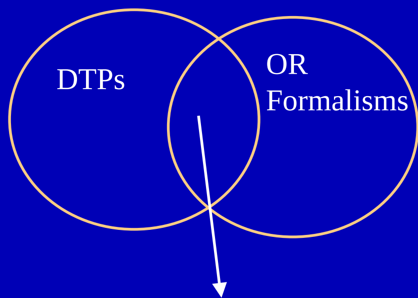
- Current fastest solver, TSAT++, reports:
  - ~10 seconds to solve problems with
    - 35 variables
    - ~210 disjunctive constraints (critical region)
    - Each with 2 disjuncts

DTP Solving and OR Scheduling Formalisms



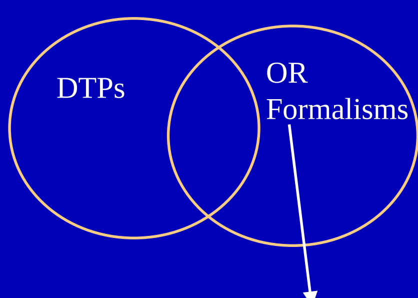
DTPs designed for the needs of planning with temporal constraints

DTP Solving and OR Scheduling Formalisms



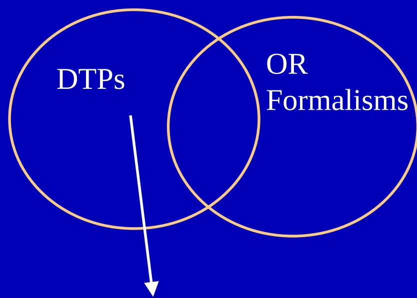
Example: Job Shop Scheduling
Temporal precedence constraints: easy to model with DTPs
Resource constraints: more cumbersome with DTPs

DTP Solving and OR Scheduling Formalisms



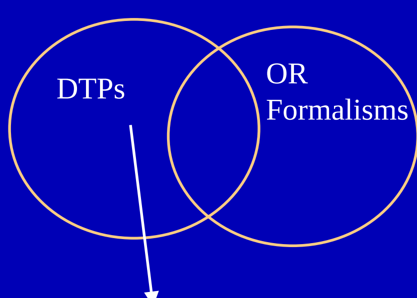
Example: Preemption

DTP Solving and OR Scheduling Formalisms



Example: 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

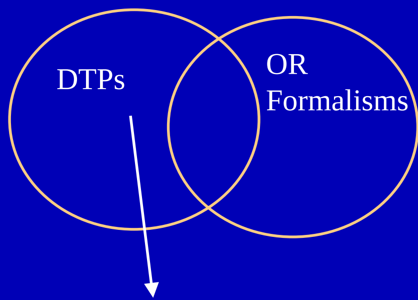


But only DTPs can express general constraints
"If treatment A doesn't last long enough, perform treatment B for a given duration."

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

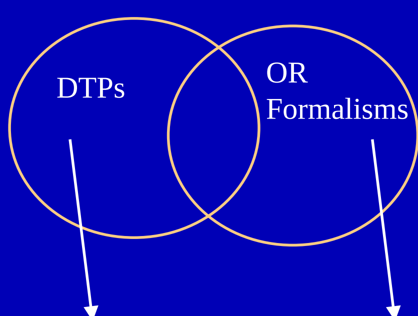
$\equiv \ (A_S - A_E) < -d \vee (B_S - B_E) < -e$

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

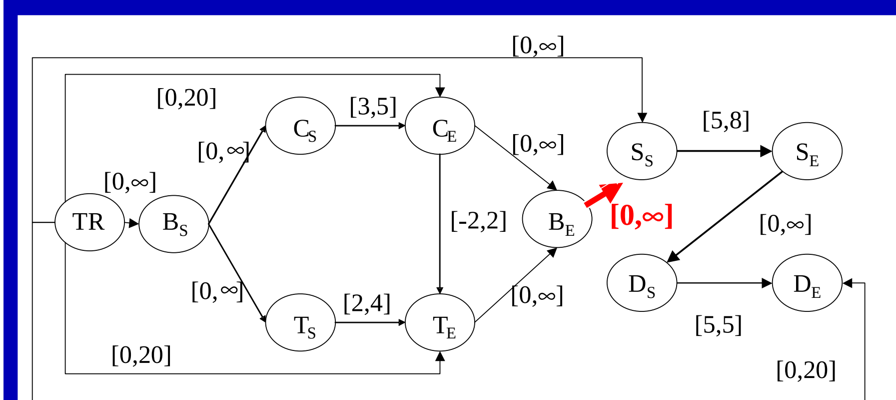
- With total control of the execution process:
- Given a DTP, find a consistent component STP *S*
- Dispatch *S* using STP dispatch algorithm

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, $t \le e - TR \le t$
  - Find a new consistent component STP *S*

The Morning Plan



$B_E - S_S \le 0 \vee D_E - B_S \le 0$

Detect S_s at 8:03

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

A Problem

- Might “miss” a solution

- $X = 2 \vee X = 1$
- $Y = 3 \vee Y = 2$
- $Y > X$

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

<u>Execution Table:</u>

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

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

Enabled events and their time windows

<u>Deadline Formula:</u>

<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 $F = F \wedge (\vee\, x \mid S_x \in \text{MinCover}\ x)$.

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 $F = F \wedge (\vee x \mid S_x \in \text{MinCover } x)$.

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 $S_x = \{i \mid U(x,i) \in U_{MIN}\}$

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

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

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

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

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

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

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

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

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

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

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 $F = F \wedge (\vee\, x \mid S_x \in \text{MinCover}\ x)$.

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

$S_x = \{1,2\}$

$S_y = \{3,4\}$

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

$F = x \vee y$

Larger Deadline Formula

- 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
  - $(x \vee y) \wedge (x \vee z \vee w)$

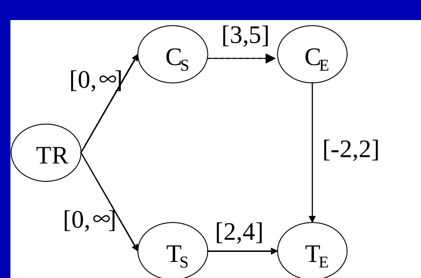
The Dispatch Bottleneck

- 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

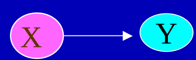
Breakfast Again

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

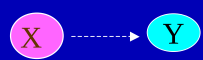


Handling Temporal Uncertainty

- 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!

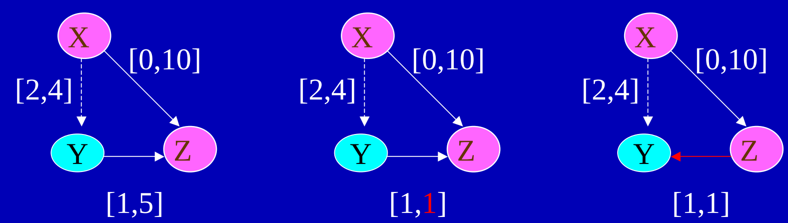
Three Notions of "Solution"

- *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}

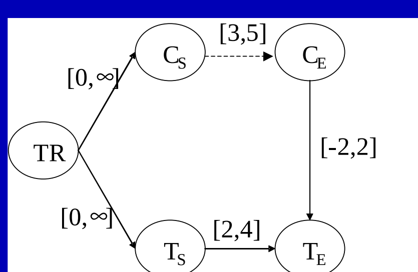
Dynamically Controllable
{X=0, Z = Y + 1}

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

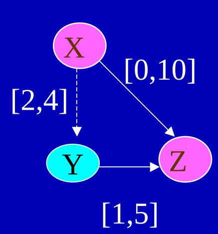
- Controllability: defines policies to determine times for controllable events depending on knowledge of uncontrollable events occurrence
- Dispatchability: identifies effective propagation paths such that knowledge on the execution of an event constrains the possible execution times for other events

Execution Policies

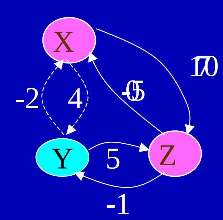
- Controllability definition emphasizes existence of solutions
- At execution time we need policies to make decision as a function of our knowledge
  - Clock time
  - Observation of event occurrence (if possible)
- Like in the case of STPs, provide ways to determine bounds and repropagation methods to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



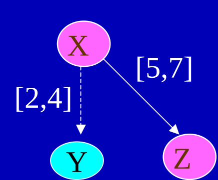
Strongly Controllable



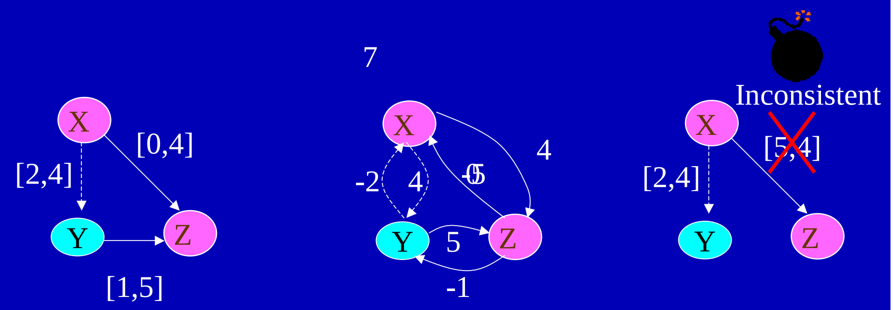
YZ ≤ YX + XZ

Step 1: tighten
XZ = YZ – YX

Step 2: delete YZ



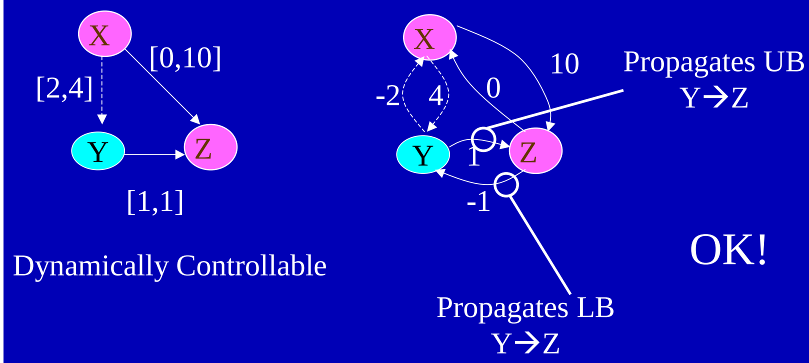
Strongly controllable policies



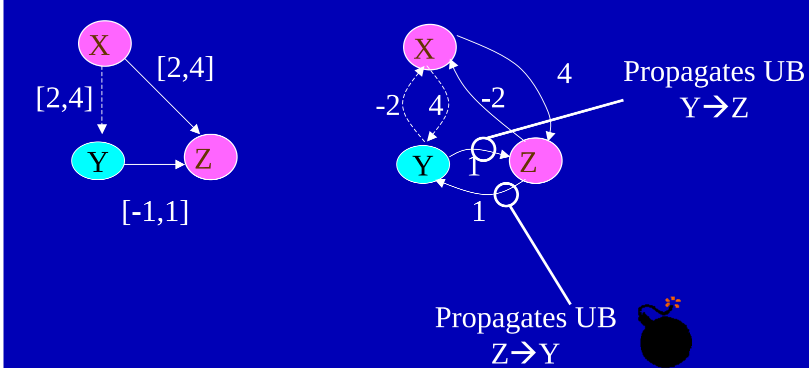
Pseudo-Controllability

- 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) ☹

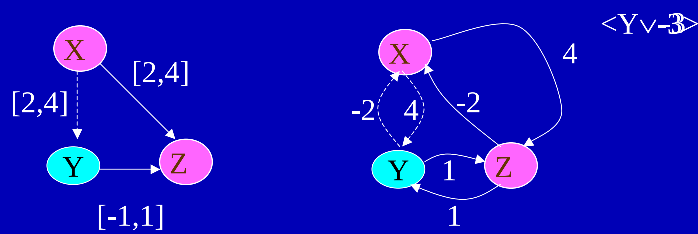
Pseudo-controllable policies



Pseudo-controllable policies



Tightening of controllable links

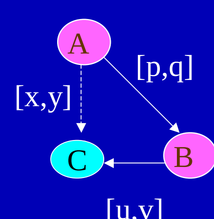


Step 1: tighten
ZX = ZY – XY
(no knowledge of Y occurrence)

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

Computing Dynamic Controllability of an STPU

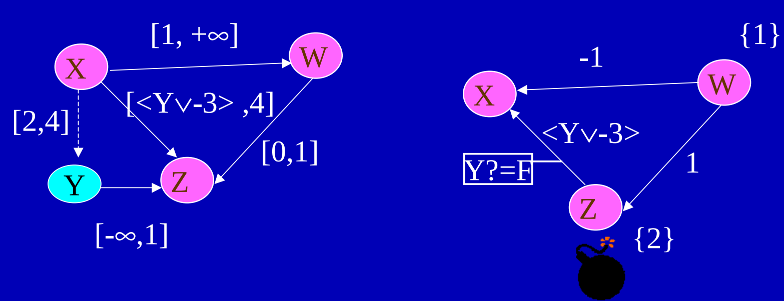
- Use *triangular reductions*
- Case 1: $v < 0$
  - B *follows* C, so d.c.
- Case 2: $u \geq 0$
  - B *precedes* C: tighten AB to [y-v, x-u] to make d.c.
- Case 3: $u < 0$ and $v \geq 0$
  - B is *unordered* w.r.t C: tighten lower bound of AB to (C or y-v) to make d.c.
- Iterate on the entire network



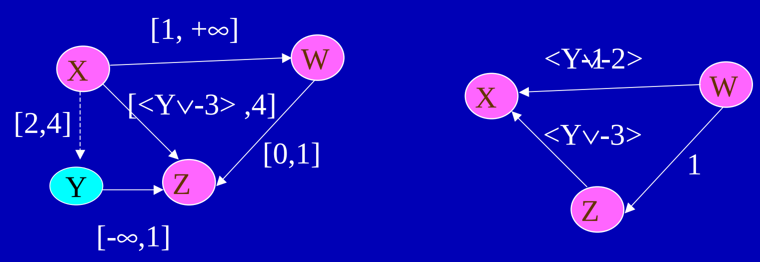
Wait Propagation Rules

- "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

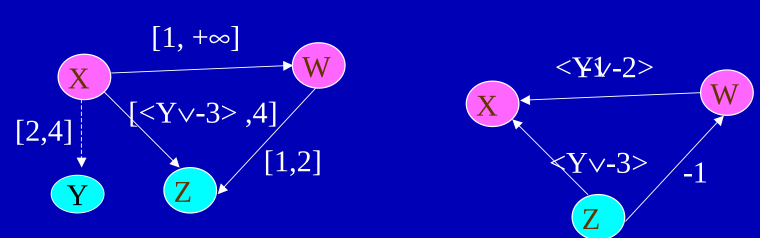
Wait Propagation over Controllable Edges



Wait Propagation over Controllable Edges



Wait Propagation over Uncontrollable Edges



Full Dynamic Controllability Algorithm

```
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"
  }
```

Termination Condition

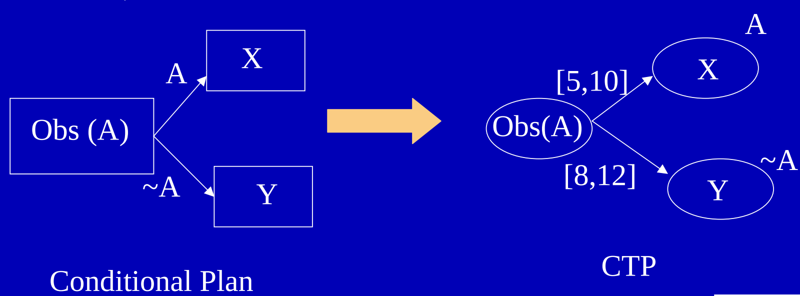
- Without further analysis, the algorithm is pseudo-polynomial
  - Pseudo-controllability: O(NE + N^2log N)
  - Tightening: O(N^3)
  - Number of repetition of cycle: U, number of time units in widest time bound
- Complexity: O(U N^3)
- U could be very large

Cutoff bound

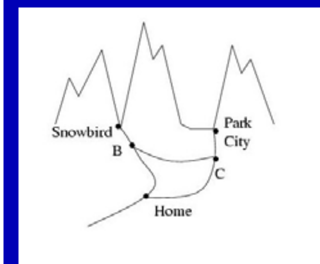
- Since the number of edges is finite, indefinite tightening is due to the existence of propagation cycles
- Cycle traversal must repeat after a maximum number of propagation (as in the Bellman-Ford algorithm for shortest paths
- Cutoff bound for dynamic controllability:
  - O(NK) with K = number of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

Handling Causal Uncertainty

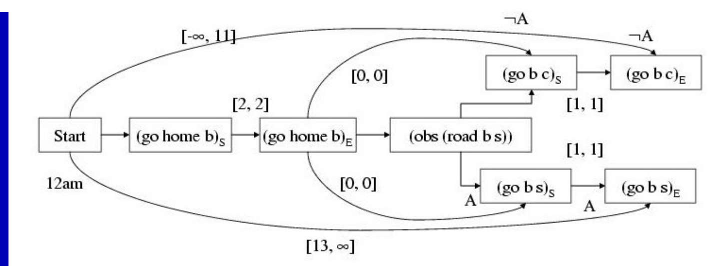
- CTP (e.g., CSTP)
- Label each node—events are executed only if their associated label is true (at a specified observation time)



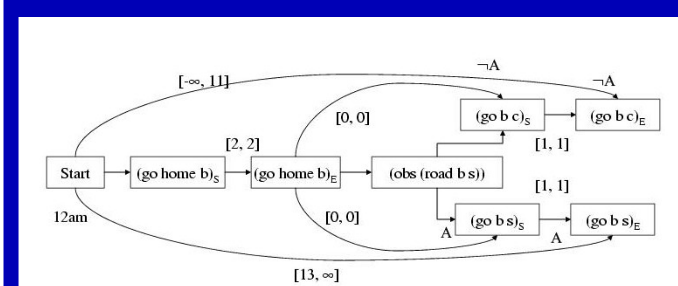
Conditional Plan as CTP



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).

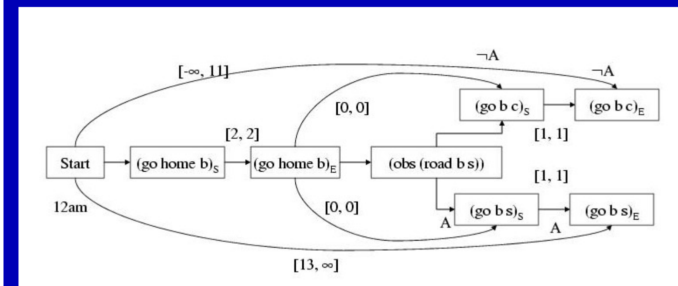


Strong Consistency



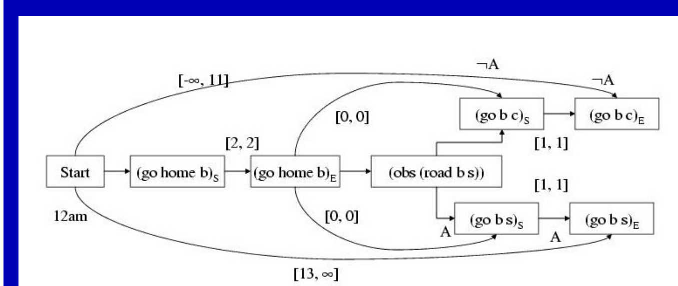
- Not strongly consistent: Must not be at B before 12 (if A is true); must be at B by 10 (if A is false)—and can't observe A until you're at B.

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

Feature	Domain of Values
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 →

  (Action $_1 \wedge$ Preconditions$_1 \wedge$ Constraints$_1$) $\vee$

  (Action $_2 \wedge$ Preconditions$_2 \wedge$ Constraints$_2$) $\vee$

  . . .

  (Action $_n \wedge$ Preconditions$_n \wedge$ 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) ∧

$h_e = c_s \wedge c_e - c_s \leq 120$] ∨

[holds(bathing, b_s, b_e, yes) ∧

$b_e = c_s \wedge c_e - c_s \leq 120$]

Alternative Actions

Example 3

holds(bread, r_s, r_e, toasting) →

holds(toaster-status, t_s, t_e, on) ∧

$t_s = r_s \wedge t_e = r_e$

holds(toaster-contents, c_s, c_e, full) ∧

$c_s \leq r_s \wedge r_e \leq c_e \wedge$

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)] $\rightarrow$
$b_e < t_s \vee t_e < b_s$ ⬅ Mutual exclusion

- Additional mutual exclusion constraints are implicit in uniqueness constraints

Planning Axioms

General Form:

Assertion $\wedge$ Assertion $\wedge$ . . . Assertion $\rightarrow$ ⬅ Head

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

(Assertions $\wedge$ Constraints) $\vee$ ⬅ 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	$\boxed{ready(r_s,r_e)}$	$-2 \leq r_e - t_e \leq 2$
Bread	$\boxed{toast(t_s,t_e)}$	$r_e - TR \leq 500$
Toaster-status		$t_e - TR \leq 500$
Toaster-contents		$d_e - TR \leq 500$
Clean		
Showering		
Bathing		
Dressed	$\boxed{yes(d_s,d_e)}$	

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: $X_1 \wedge ... X_n \rightarrow$ 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

Coffee — $none(n_s,n_e)$ | $brewing(b_s,b_e)$ | $ready(r_s,r_e)$

Bread — $toast(t_s,t_e)$

Toaster-status

Toaster-contents

Clean

Showering

Bathing

Dressed — $yes(d_s,d_e)$

$-2 \le r_e - t_e \le 2$

$r_e - TR \le 500$

$t_e - TR \le 500$

$d_e - TR \le 500$

$b_e = r_s$

$3 \le b_e - b_s \le 5$

$n_e = b_s$

$holds(coffee, r_s, r_e, ready) \rightarrow$

$holds(coffee, b_s, b_e, brewing) \wedge$

$(b_e = r_s) \wedge (3 \le b_e - b_s \le 5)$

$holds(coffee, n_s, n_e, none) \wedge$

$n_e = b_s$

Expanding the Initial Plan II

Coffee — $none(n_s,n_e)$ | $brew(b_s,b_e)$ | $ready(r_s,r_e)$

Bread — $toasting(o_s,o_e)$ | $toast(t_s,t_e)$

Toaster-status

Toaster-contents

Clean — $yes(c_s,c_e)$

Showering — $yes(h_s,h_e)$

Bathing

Dressed — $dressing(g_s,g_e)$ | $yes(d_s,d_e)$

$-2 \le r_e - t_e \le 2$

$r_e - TR \le 500$

$t_e - TR \le 500$

$d_e - TR \le 500$

$b_e = r_s$

$3 \le b_e - b_s \le 5$

$n_e = b_s$

$o_e = t_s$

$g_e = d_s$

$g_s \le c_e$

$h_e = c_s$

$c_e - c_s \le 120$

Causal Links and Uniqueness Conditions

Showering: yes(h_s,h_e)

Clean: yes(c_s,c_e)

no(n_s,n_e)

Exercise: yes(x_s,x_e)

Dressed: dressing(g_s,g_e)

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

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)

$l_s \le b_s$
$b_s \le l_e$
$l_s \le o_s$
$o_s \le l_e$

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

Expand (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

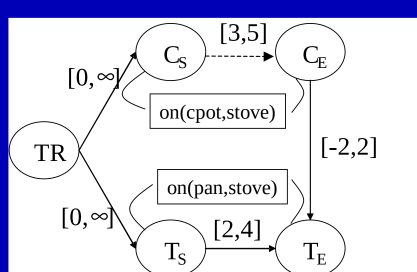
      Add constraints c ∈ d to Constraints

  Expand(Unchecked, Assertions, Constraints, Axioms)

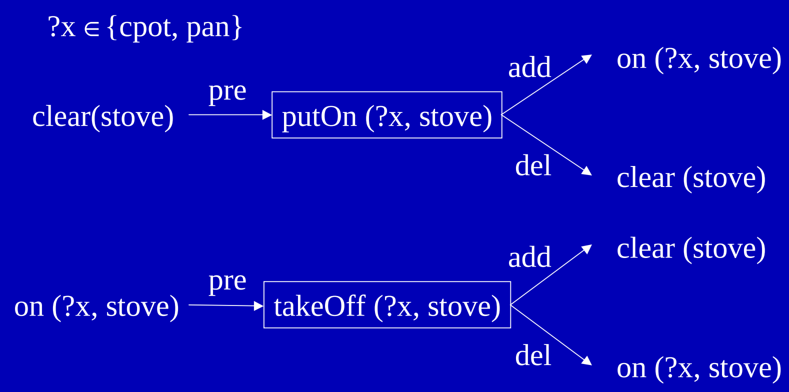
Resource Constraint Reasoning: Scheduling

Breakfast at Yosemite

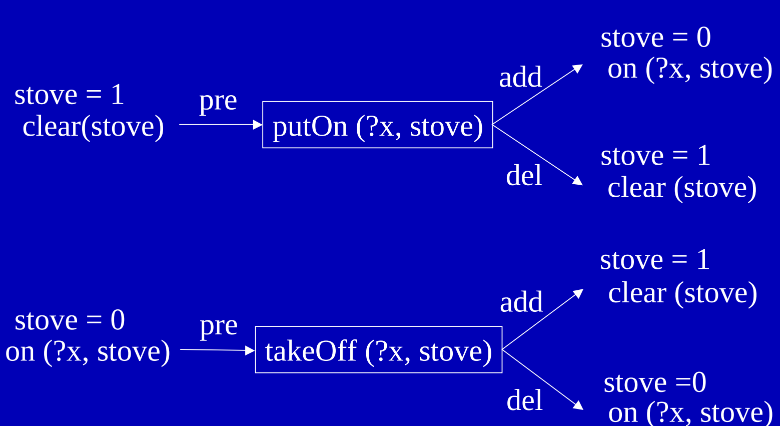
- You are backpacking so you cook the toast on a pan…
- …and you have a stove with just one burner.



Operating the stove
The Planning Perspective

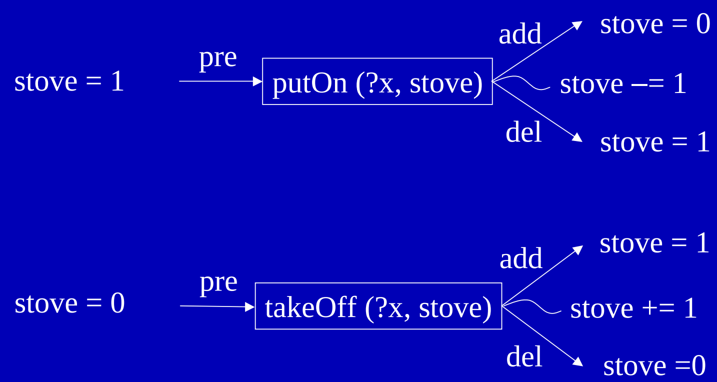


From Planning to Scheduling



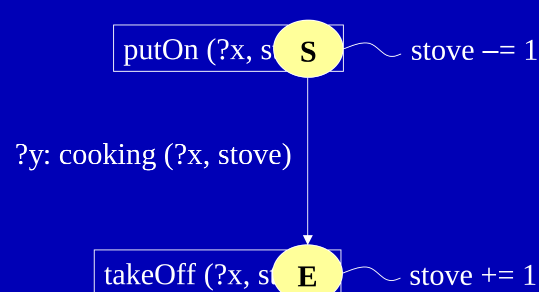
From Planning to Scheduling

$0 \leq \text{stove} \leq 1$



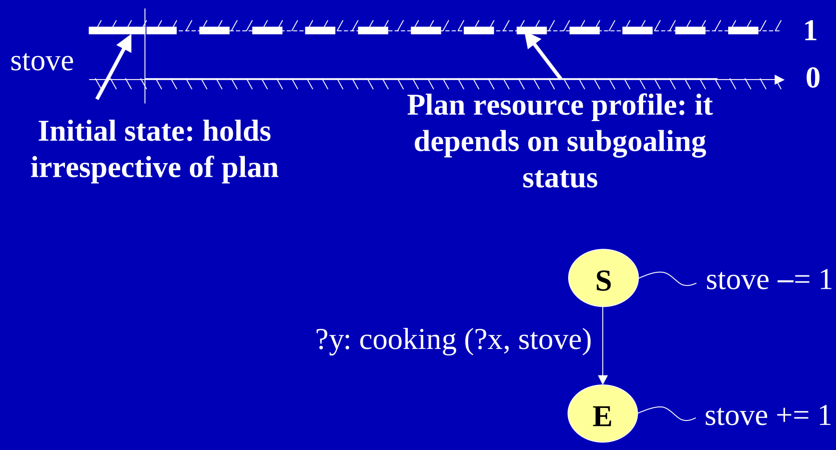
From Planning to Scheduling

$0 \leq \text{stove} \leq 1$



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

Breakfast as Scheduling

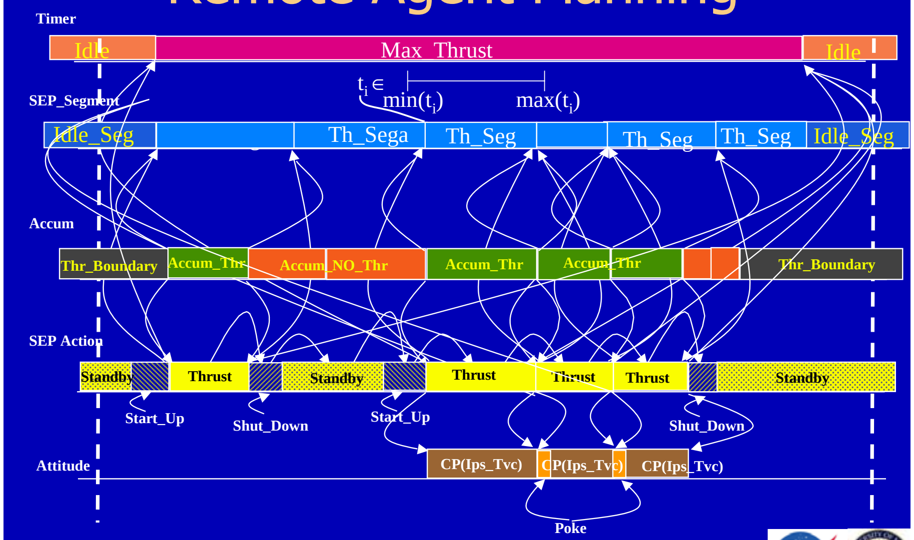


<?y, ?x> ∈ {<coffe, cpot>, <toast, pan>}

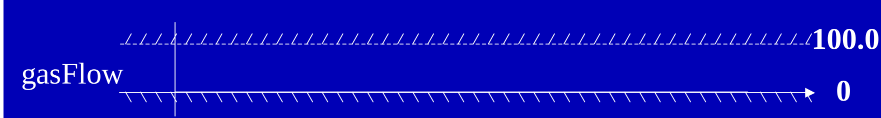
A View of Planning and Scheduling

- 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

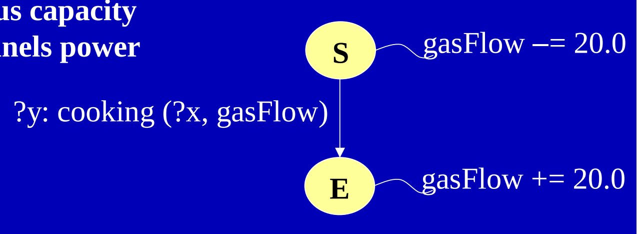
Remote Agent Planning



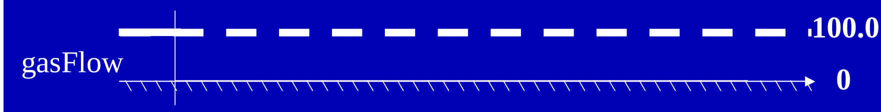
Resource Models



- Discrete/continuous capacity
- Example: solar panels power

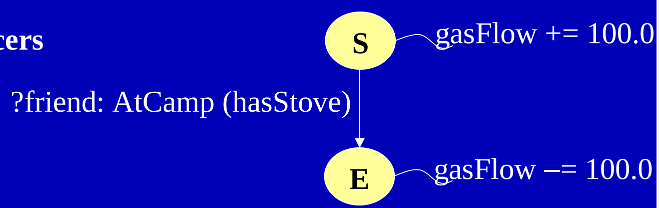


Resource Models

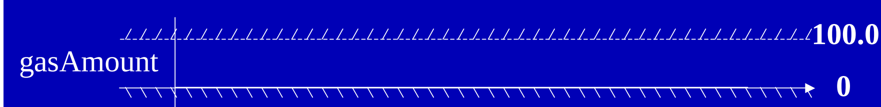


holds (Me, $Plan_s$,$Plan_e$, AtCamp(hasStove)

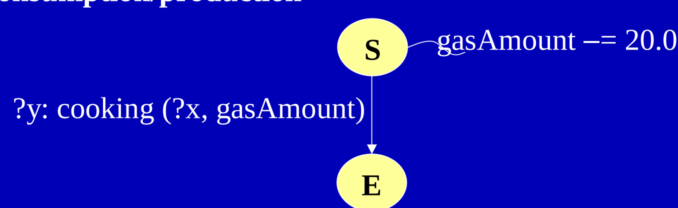
- **Resource producers**



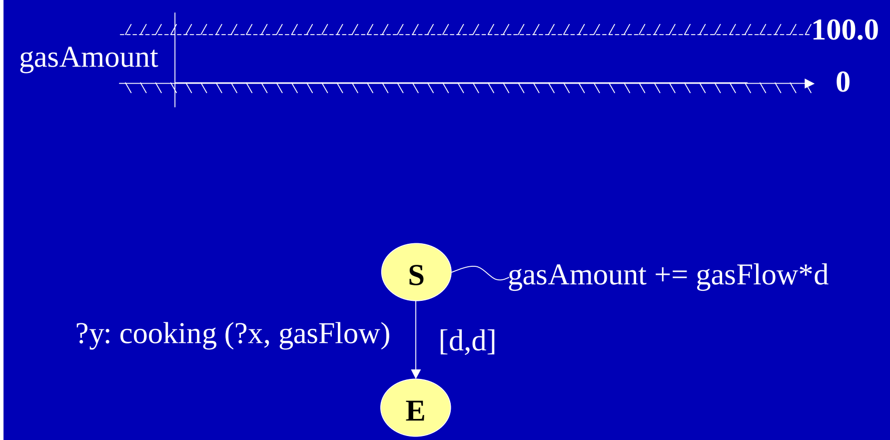
Resource Models



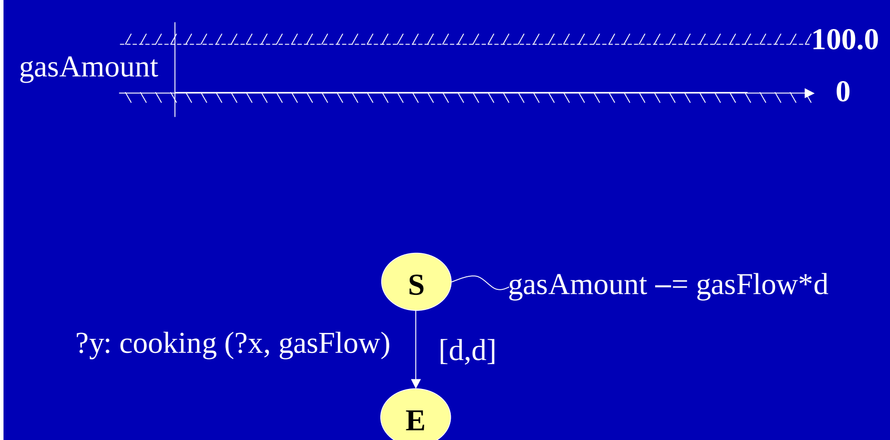
- **Permanent consumption/production**

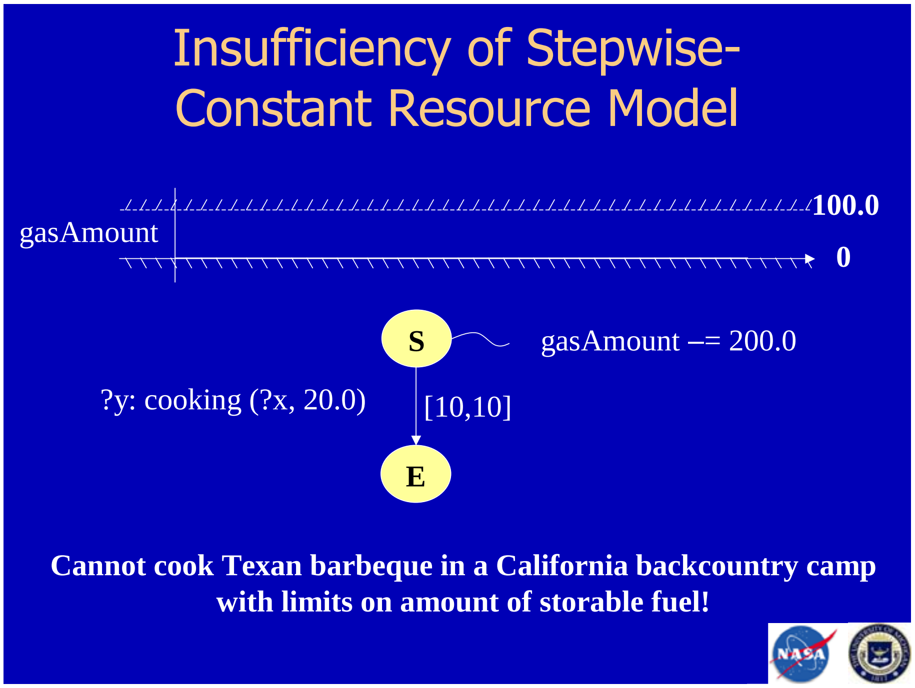


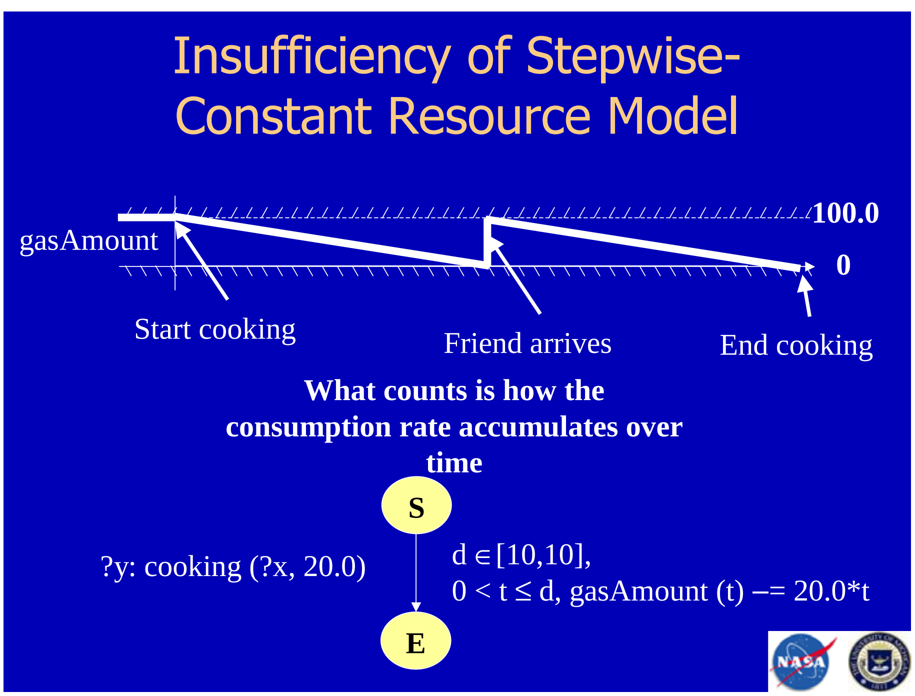
Insufficiency of Stepwise-Constant Resource Model



Insufficiency of Stepwise-Constant Resource Model



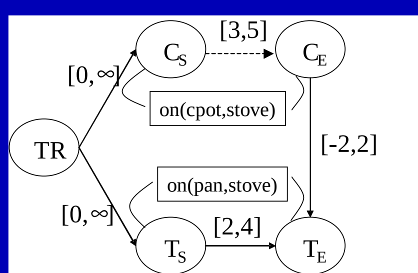

Insufficiency of Stepwise-Constant Resource Model
gasAmount
100.0
0
S
gasAmount –= 200.0
?y: cooking (?x, 20.0)
[10,10]
E
Cannot cook Texan barbeque in a California backcountry camp with limits on amount of storable fuel!


Insufficiency of Stepwise-Constant Resource Model
gasAmount
100.0
0
Start cooking
Friend arrives
End cooking
What counts is how the consumption rate accumulates over time
S
?y: cooking (?x, 20.0)
d ∈ [10,10],
0 < t ≤ d, gasAmount (t) –= 20.0*t
E

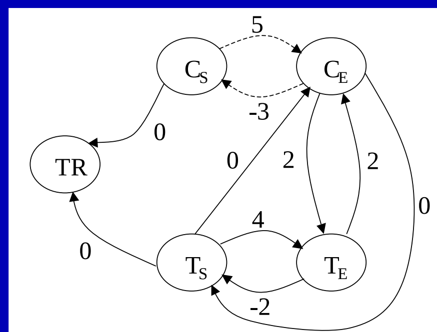
Flexibility in Plans/Schedules

- 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

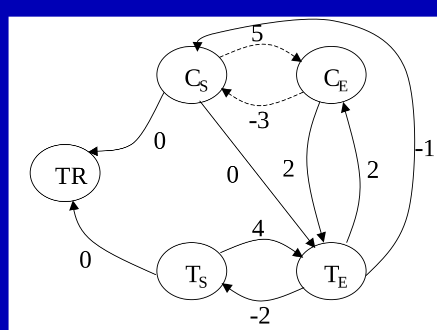


How to build a flexible schedule



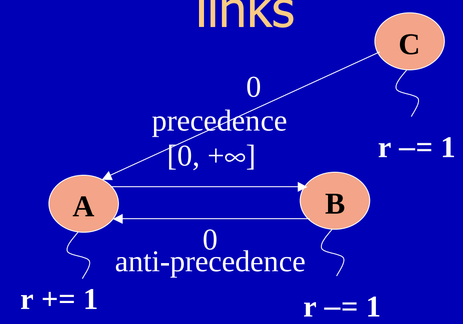
Can we start making the toast after the coffee is brewed? **YES**

How to build a flexible schedule



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

One interpretation of precedence links



- 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

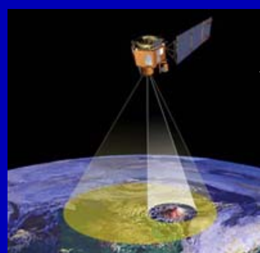
PCP scheduling

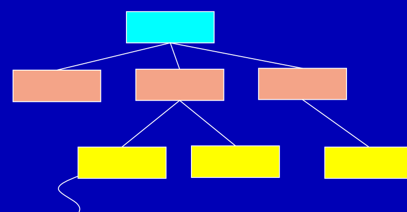
- [Cheung and Smith, 1997] use scratch propagation for unary capacity makespan optimization job-shop scheduling
- Scratch propagation can be done using Dijkstra algorithm from each end time to the start times on the same resource
- Scratch propagation cost: $O(N^2 \log N)$ but can terminate early when 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

[Chien et al. 2005] Automated Sciencecraft Experiment

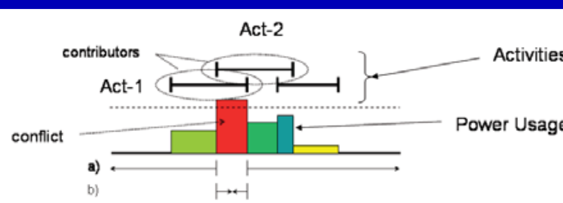
{PowerUp (Imager)} before
{s ∈ [10:00, 13:30], Image(lat, long, Mt.Etna)}





dataBuffer –= 100

Fixed-time scheduling and execution policies



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

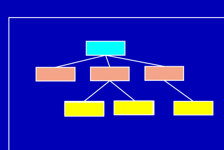
- Use a repair method to eliminate a conflict
- ASE uses a planner, not just a scheduler.
- Hence it is possible to generate new activities or select different task decompositions
- Repair methods
  - move an activity
  - delete an activity
  - add a new activity
  - detailing an activity
  - abstracting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

Add producer of resource. Not handled in classical scheduling

Chose different activity decomposition

From Planning to Execution The ideal situation

Repair plan using same method to generate it



Planner

Executive

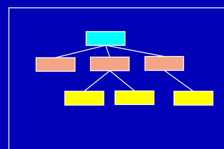
Comparison of Flexible and Fixed Policies (1)

- 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

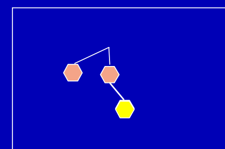
Comparison of Flexible and Fixed Policies (2)

- 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

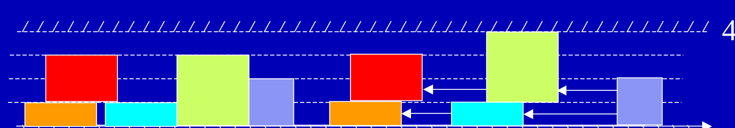


Executive

- 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

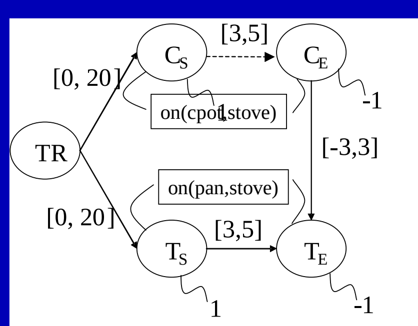
- Simple strategy for single capacity resources: simply keep the ordering constraints and uncommit the times from the fixed values
- Continuous/discrete capacity resources require the introduction of anti-precedence couplings between consumers and producers



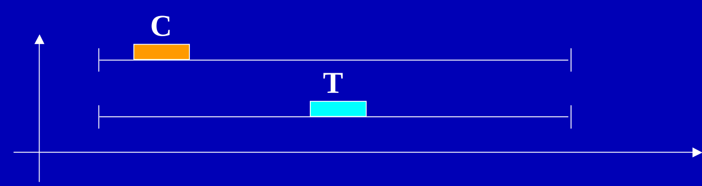
- [Policella et al, 2004] Transform fixed schedule into "chaining form" partial order
- Decompose multiple capacity resource into "virtual" single capacity resources and add couplings on chains

Probabilistic measures of resource contention

Contentious Breakfast



Time bounds and resource conflicts

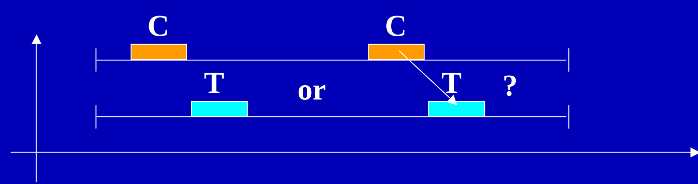


- Without further coordination, C and T are free to collide for the use of the stove
- The inclusion of anti-precedence links ("couplings" of producers to consumers) reduce and eventually eliminate the possibility of conflict

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

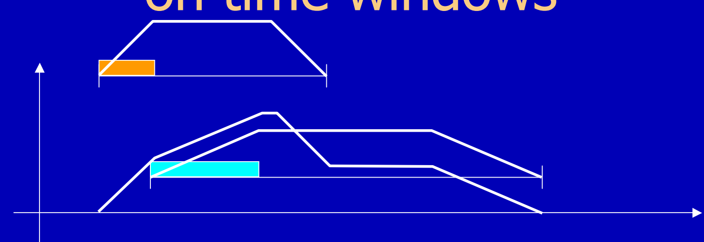
Probabilistic Resource Contention

- 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

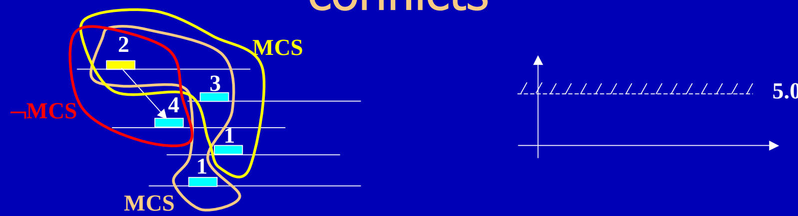
- [Beck & Fox 2000] Assumptions:
  - Fixed durations, consumption at start, same production at end
  - Uniform distribution of start times
  - Time bounds only
- Individual action demand inside the time bound:
  - $d_i(t) = \Sigma_{max(est,\ t\text{-}dur) \leq \tau \leq min(lft,\ t+dur)}\ r_i/(lft -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

Probabilistic contention based on time windows



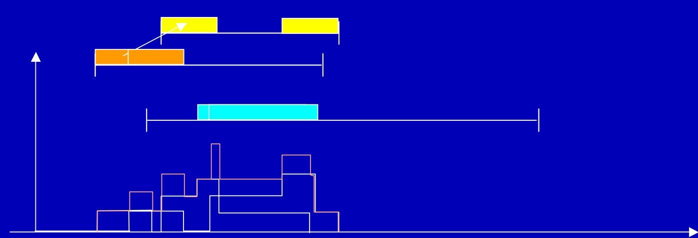
- Aggregate demand = sum demand curves = expected value of instantaneous resource requests
- How to use it
  - Find maximum over all curves → maximum contention
  - Find pair with maximum demand at contention point that are not already ordered

Another way to characterize conflicts



- 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

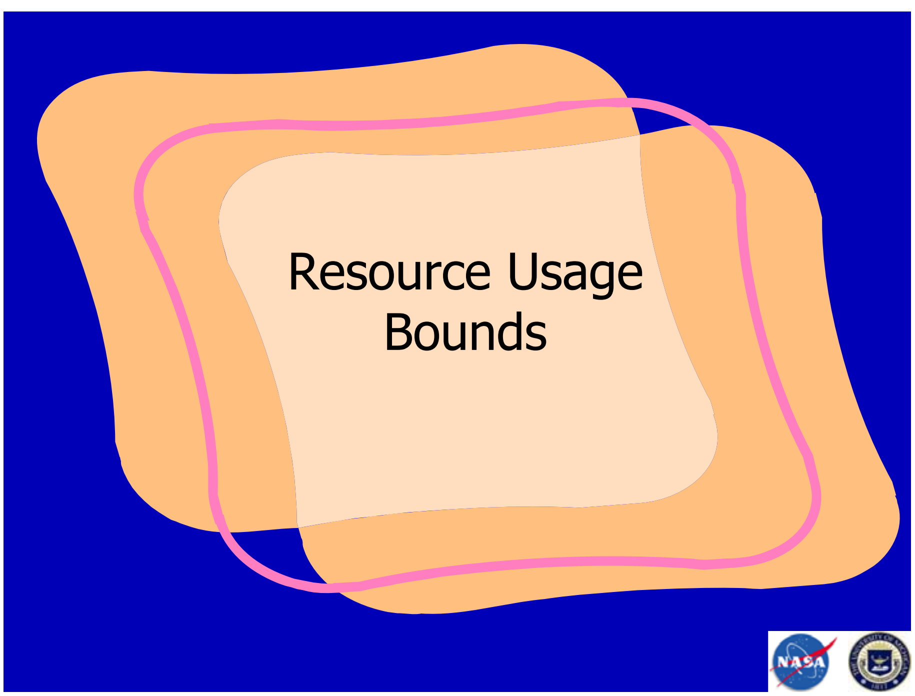
Probabilistic contention using precedence information



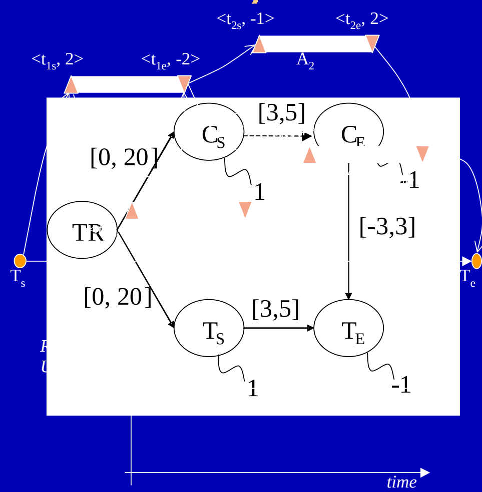
- Monte Carlo resource contention [Muscettola 1994]
- Consider all known temporal constraints
- Simulate a sample of executions ignoring resource contention
- Then compare expected resource request to resource limit to identify conflict areas
- Monte Carlo methods are also used in analysis of plan executions

Comparison of statistical contention measures

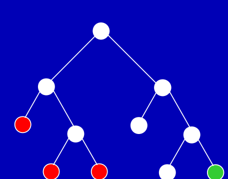
- Monte Carlo simulation is more informed
- Time-window method is less computationally expensive
  - Time windows: O(N) in time and space
  - Monte Carlo: with sample size S
    - O(S E) in time (if network is dispatchable)
    - O(S N) in space
- Monte Carlo method also biases sample depending on stochastic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



From breakfast to infinity and beyond



Search Guidance

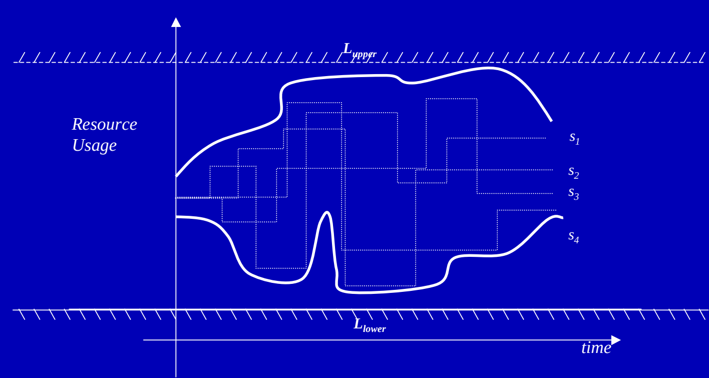


- The ability of detecting early that the flexible plan is resource/time inconsistent can save exponential amount of work
- Same for early detection of a solution

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

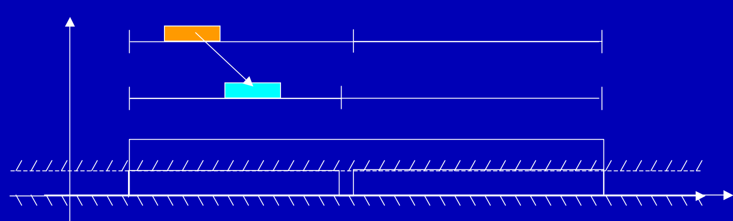


- Case 1: bounds always within limits → solution
- Case 2: bounds at least once outside the limit → inconsistency
- Case 3: otherwise → search

Bounds are costly

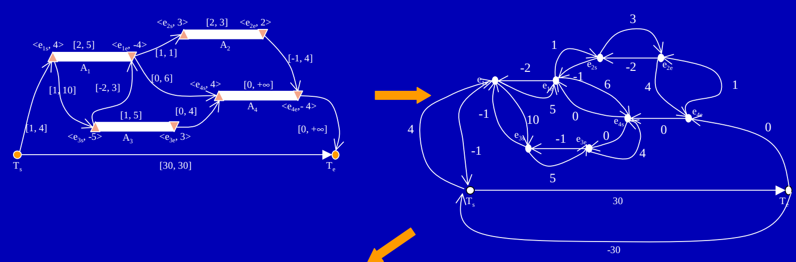
- In summary, bounds try to summarize the status of an exponential number of schedules
- As in the case of probabilistic measures, we can obtain different bounds depending of how much structural information on producer/consumer coupling we use
- The more information, the tighter the bound
- The more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

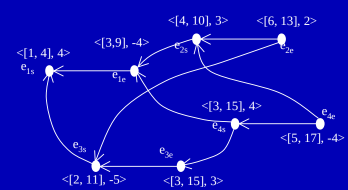
Least informative bounds



- 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

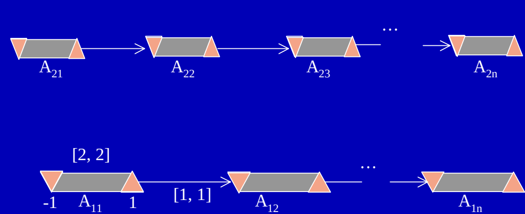




Anti-Precedence Graph

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

Balance Constraint Bounds



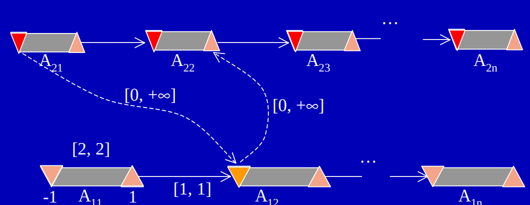
$L_{min, \leq} = -n - 1$ $\qquad$ $L_{min, \geq} = -n$

- 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

- 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]

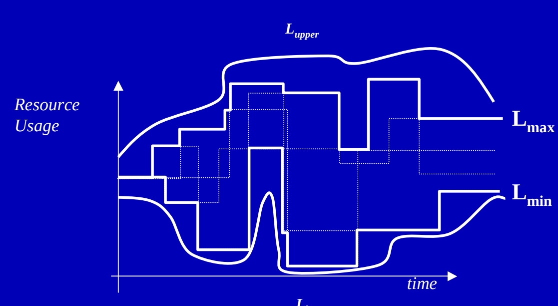
Looseness of Balance Constraint Bound



- 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

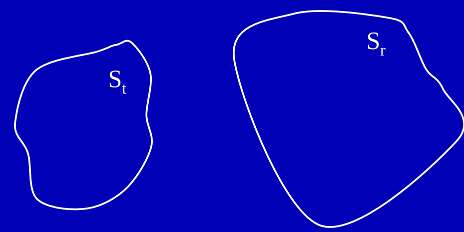
Resource Envelopes

Resource Envelope



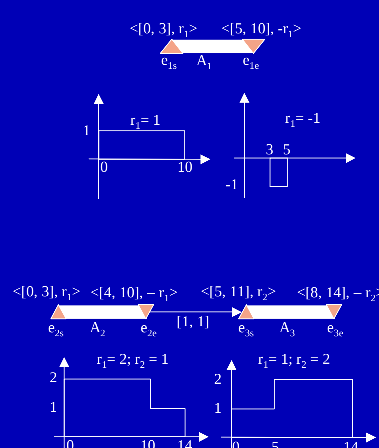
- Manager: “I am tired of half measures. How about giving me the tightest possible bounds?”
- Computer Scientist A: “Hmmm…I don’t know. It looks difficult. Remember the exponential number of schedules?
- Rocket Scientist B: “Aw, no problem. I’ll give you a fast polynomial algorithm for it …”

"WHAT?"

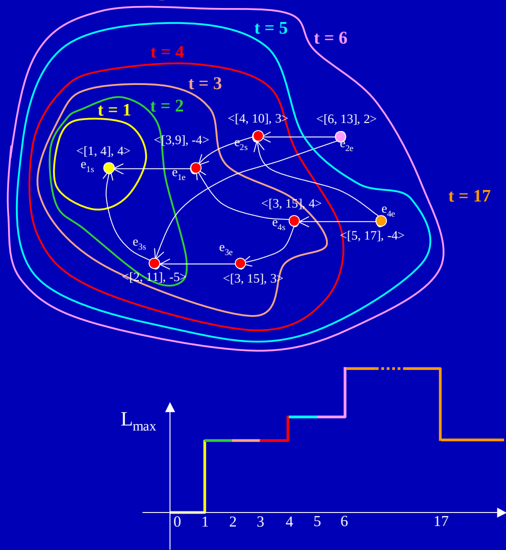


- $\exists s \in S_t \mid s \in S_r$ Scheduling problem NP-hard
- $\forall s \in S_t \mid s \in S_r$ Resource envelope looks hard(er)

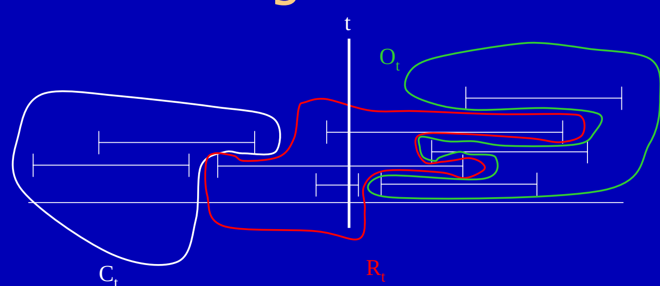
Resource Envelope Method Intuitive Description



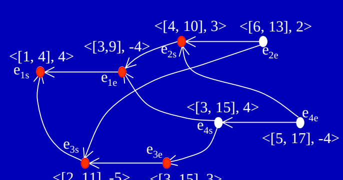
Building a full envelope



Pending Events



P_X = predecessor set of event set X



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

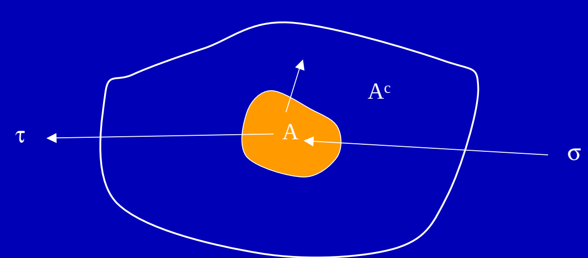
P_{max} = predecessor set of maximum total weight

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?

Maximum flows

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



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

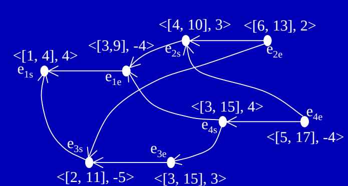
Residual network
For each pair of nodes: $r_f(e_1, e_2) = c(e_1, e_2) - f(e_1, e_2)$

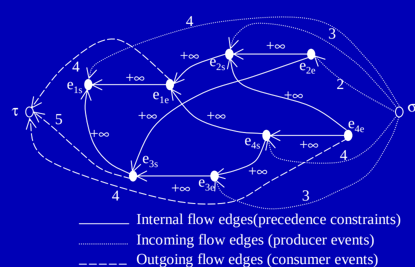
Augmenting path = path from σ to τ with positive residual
No augmenting path = flow is maximum

Maximum Flow Algorithms

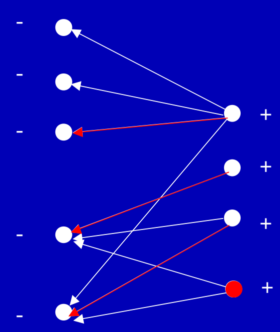
Algorithm	Time Complexity	Complexity Key
Labeling	*O(N E U)*	Total pushable flow
Capacity scaling	*O(NE logU)*	Total pushable flow
Successive shortest paths	*O(N^2E)*	Shortest distance to τ
Generic Preflow-push	*O(N^2E)*	Distance label
FIFO Preflow-push	*O(N^3)*	Distance label

Resource Increment Flow Network





A simple P_{max} selection problem



Maximum Resource-Level Increment Predecessor Set

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

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

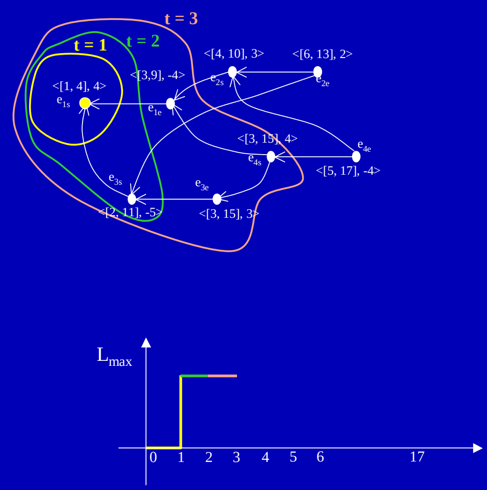
… given a P_{max} is there a temporally consistent schedule and a time t_x such that all events in C_H and P_{max} are schedule at or before t_x and all events in P^c_{max} and O_H are scheduled after t_x?

Theorem 3: Yes!

Maximum Resource Level and Resource Envelope

- 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?

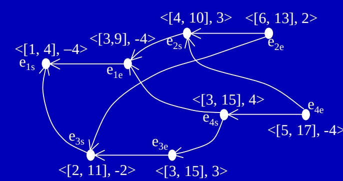
Building a full envelope



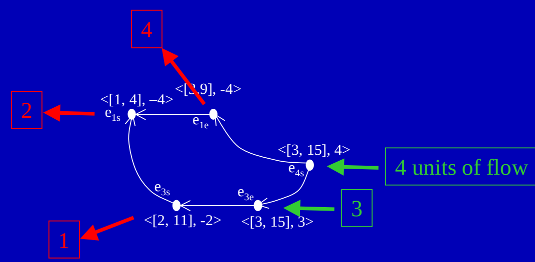
Staged Resource Envelope

- 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?

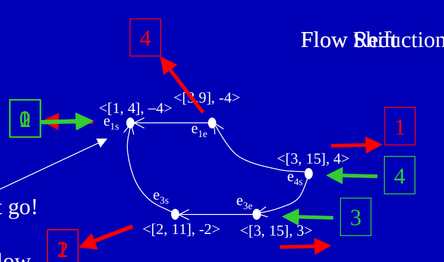


t = 3



Flow Contraction

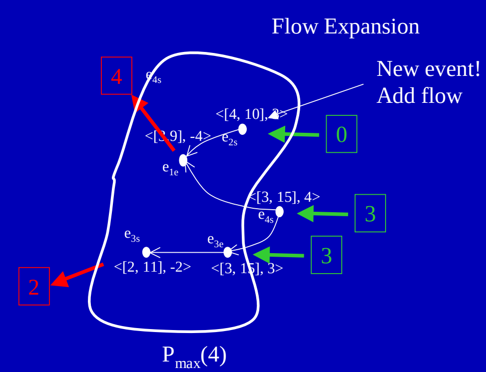
t = 4



This event must go!
It enters C_4
Push back the flow

Flow Expansion

t = 4



Recursive Equation

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

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

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

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

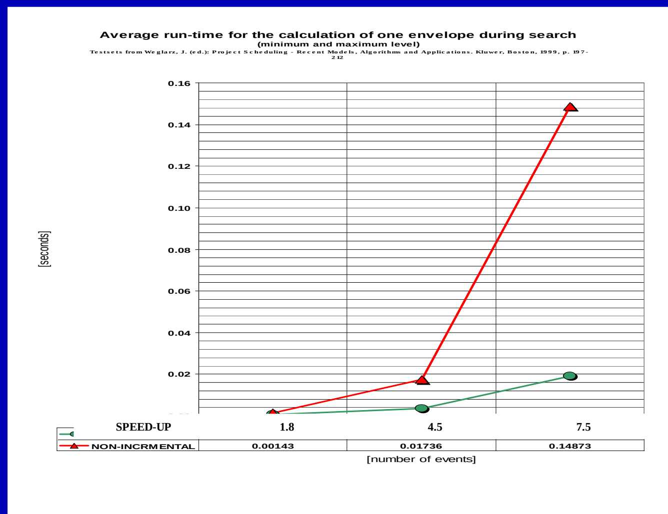
Complexity Analysis

- 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

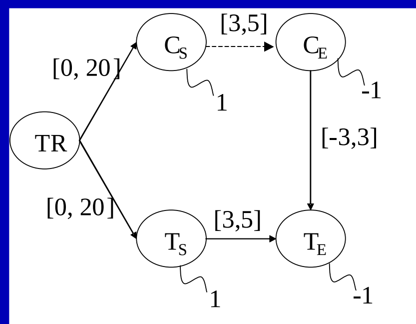


Envelope scheduling so far

- [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**.

Constraint-Satisfaction Processing:

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

Qualitative Models of Time:

- 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).**

Simple and Disjunctive Temporal Problems:

- 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.

Dispatch of Simple Temporal Problems:

- N. Muscettola, P. Morris, and I. Tsamardinos, "Reformulating Temporal Plans for Efficient Execution," *Prof. of the 6th Conference on Principles of Knowledge Representation and Reasoning (KR)*, 1998.
- I. Tsamardinos, P. Morris, and N. Muscettola, "Fast Transformation of Temporal Plans for Efficient Execution," *Prof. of the 15th National Conference on Artificial Intellience (AAAI)*, 1998.
- **M. E. Pollack and I. Tsamardinos, "Efficiently Dispatching Plans Encoded as Simple Temporal Problems," in I. Vlahavas and D. Vrakas, eds., *Intelligent Techniques for Planning*, Idea Group Publishing, 2005.**

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.