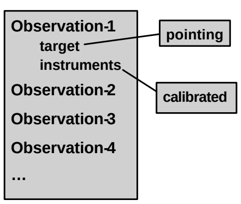
Temporal Plan Execution: Dynamic Scheduling and Simple Temporal Networks

Brian C. Williams
16.412J/6.834J
March 7th, 2005

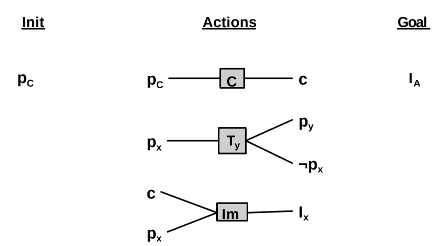
Outline

- Review: Constraint-based Interval Planning
- Simple Temporal Networks
- Temporal Consistency and Scheduling
- Execution Under Uncertainty

Simple Spacecraft Problem



Example

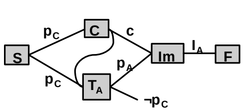


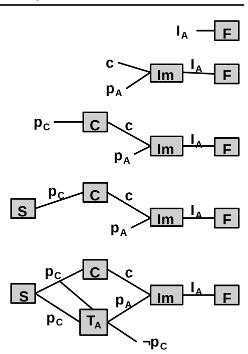
16.410/13: Solved using Graph-based Planners (Blum & Furst)

Partial Order Causal Link Planning

(SNLP, UCPOP)

1. Select an open condition
2. Choose an op that can achieve it
   - Link to an existing instance
   - Add a new instance
3. Resolve threats



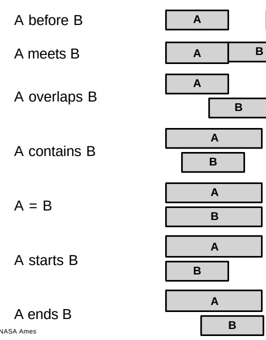


Needed Extensions

- **Time**
- **Resources**
- **Utility**
- **Uncertainty**

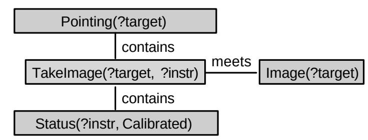
Based on slides by Dave Smith, NASA Ames

Representing Timing: Qualitative Temporal Relations [Allen AAAI83]

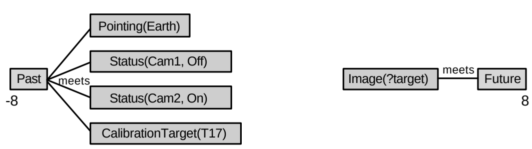


TakeImage Pictorially

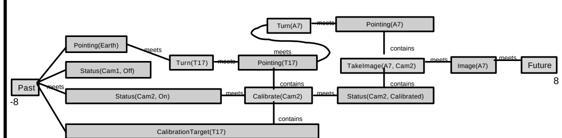
TakeImage (?target, ?instr)	
contained-by	Status(?instr, Calibrated)
contained-by	Pointing(?target)
meets	Image(?target)



A Temporal Planning Problem



A Consistent Complete Temporal Plan



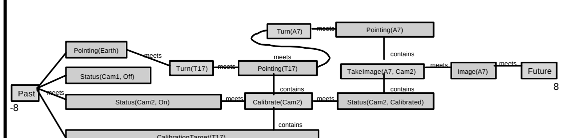
CBI Planning Algorithm

Choose:

- introduce an action & instantiate constraints
- coalesce propositions

Propagate temporal constraints

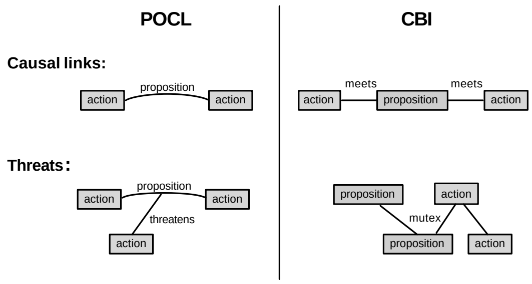
A Consistent Complete Temporal Plan



Planner Must:

- Check schedulability of candidate plans for correctness.
- Schedule the activities of a complete plan in order to execute.

Relation to Causal Links & Threats



Examples of CBI Planners

Zeno (Penberthy)	intervals, no CSP
Trains (Allen)	
Descartes (Joslin)	extreme least commitment
IxTeT (Ghallab)	functional rep.
HSTS (Muscettola)	functional rep., activities
EUROPA (Jonsson)	functional rep., activities
Kirk (Williams)	HTN

Outline

- Review: Constraint-based Interval Planning
- Simple Temporal Networks
- Temporal Consistency and Scheduling
- Execution Under Uncertainty

Qualitative Temporal Constraints Maybe Expressed as Inequalities

(Vilain, Kautz 86)

- x before y $X^+ < Y^-$
- x meets y $X^+ = Y^-$
- x overlaps y $(Y^- < X^+)$ & $(X^- < Y^+)$
- x during y $(Y^- < X^-)$ & $(X^+ < Y^+)$
- x starts y $(X^- = Y^-)$ & $(X^+ < Y^+)$
- x finishes y $(X^- < Y^-)$ & $(X^+ = Y^+)$
- x equals y $(X^- = Y^-)$ & $(X^+ = Y^+)$

Inequalities may be expressed as binary interval relations:

$Y^- - X^+ < [0, +\text{inf}]$

Generalize to include metric constraints:

$Y^- - X^+ < [\text{lb}, \text{ub}]$

Metric Time: Temporal CSPS

(Dechter, Meiri, Pearl 91)

"Bread should be eaten within a day of baking."

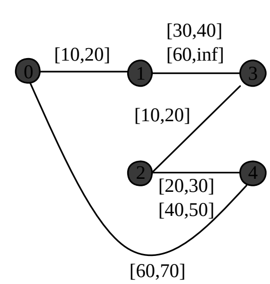
? $0 \le [T^+(\text{baking}) - T^-(\text{eating})] \le 1$ day

$< X_i, T_i, T_{ij} >$

- X_i continuous variables
- T_i, T_{ij} interval constraints
  $\{I_1, \ldots, I_n\}$ where $I_i = [a_i, b_i]$

– $T_i = (a_i \le X_i \le b_i)$ or . . . or $(a_i \le X_i \le b_i)$
– $T_{ij} = (a_1 \le X_i - X_j \le b_1)$ or ... or $(a_n \le X_i - X_j \le b_n)$

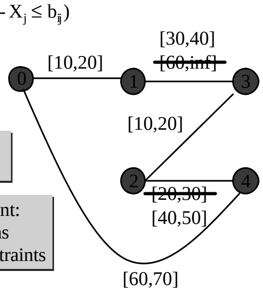
TCSP Are Visualized Using Directed Constraint Graphs



Simple Temporal Networks (STNs)
(Dechter, Meiri, Pearl 91)

At most one interval per constraint

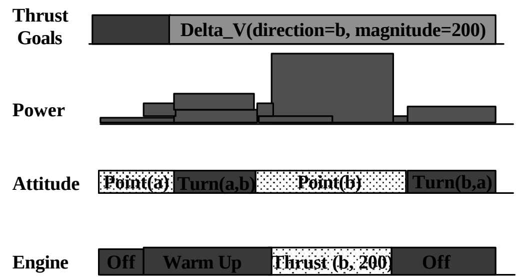
- $T_{ij} = (a_{ij} \le X_i - X_j \le b_{ij})$



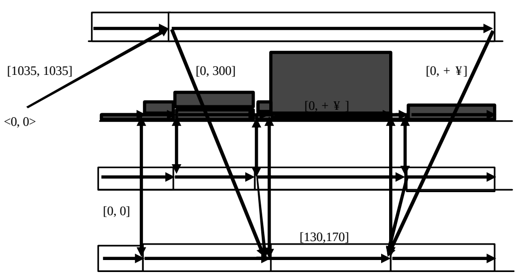
Can't represent:
• Disjoint activities

Sufficient to represent:
• most Allen relations
• simple metric constraints

A Temporal Plan Forms an STN



A Temporal Plan Forms an STN



Outline

- Review: Constraint-based Interval Planning
- Simple Temporal Networks
- Temporal Consistency and Scheduling
- Execution Under Uncertainty

TCSP Queries
(Dechter, Meiri, Pearl, AIJ91)

- Is the TCSP consistent? *Planning*
- What are the feasible times for each X_i? *Planning*
- What are the feasible durations between each X_i and X_j? *Planning*

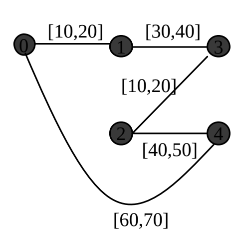
- What is a consistent set of times? *Scheduling*
- What are the earliest possible times? *Scheduling*
- What are the latest possible times?

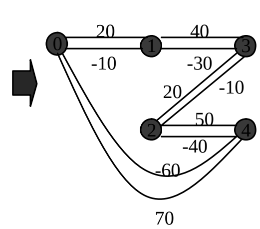
To Query an STN, Map to a Distance Graph $G_d = < V,E_d >$

- Edge encodes an upper bound on distance to target from source.
- Negative edges are lower bounds.

$T_{ij} = (a_{ij} \le X_j - X_i \le b_{ij})$

$X_j - X_i \le b_{ij}$
$X_i - X_j \le - a_{ij}$





G_d Induces Constraints

- Path constraint: $i_0 = i, i_1 = \ldots, i_k = j$

$$X_j - X_i \le \sum_{j=1}^{k} a_{i_{j-1}, i_j}$$

? Conjoined path constraints result in the shortest path as bound:

$$X_j - X_i \le d_{ij}$$

where d_{ij} is the shortest path from i to j

Conjoined Paths Computed using All Pairs Shortest Path

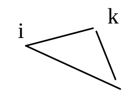
(e.g., Floyd-Warshall, Johnson)

1. for i := 1 to n do d_{ii} — 0;
2. for i, j := 1 to n do d_{ij} — a_{ij};

Initialize distances

3. for k := 1 to n do
4. for i, j := 1 to n do
5. d_{ij} — min{d_{ij}, d_{ik} + d_{kj}};

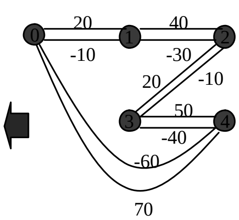
Take minimum distance over all triangles



Shortest Paths of G_d

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph



Map To STN Minimum Network

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

	0	1	2	3	4
0	[0]	[10,20]	[40,50]	[20,30]	[60,70]
1	[-20,-10]	[0]	[30,40]	[10,20]	[50,60]
2	[-50,-40]	[-40,-30]	[0]	[-20,-10]	[20,30]
3	[-30,-20]	[-20,-10]	[10,20]	[0]	[40,50]
4	[-70,-60]	[-60,-50]	[-30,-20]	[-50,-40]	[0]

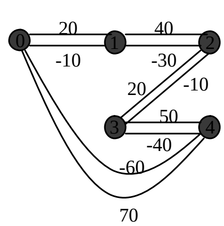
STN minimum network

Schedulability: Plan Consistency

No negative cycles: $-5 > T_A - T_A = 0$

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph



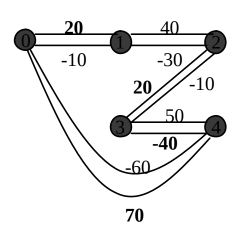
Scheduling: Latest Solution

Node 0 is the reference.

$S_1 = (d_{01}, \ldots, d_{0n})$

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph



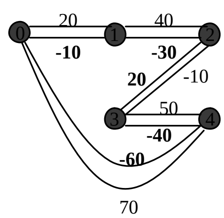
Scheduling: Earliest Solution

Node 0 is the reference.

$S_1 = (-d_{10}, \ldots, -d_{n0})$

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph



Scheduling: Window of Feasible Values

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

Earliest Times d-graph

Latest Times

- X_1 in [10, 20]
- X_2 in [40, 50]
- X_3 in [20, 30]
- X_4 in [60, 70]

Scheduling without Search: Solution by Decomposition

- Can assign variables in any order, without backtracking.

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

- Select value for 1

  → 15 [10,20]

Solution by Decomposition

- Can assign variables in any order, without backtracking.

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

- Select value for 1

  → 15 [10,20]

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of Y using all selected X: $Y \in X + |XY|$

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

- Select value for 1

  → 15
- Select value for 2, consistent with 1

  → 45 [40,50], 15+[30,40]

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of Y using all selected X: $Y \in X + |XY|$

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

- Select value for 1

  → 15
- Select value for 2, consistent with 1

  → 45 [45,50]

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of Y using all selected X: Y ∈ X + |XY|

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

- Select value for 1
  → 15
- Select value for 2, consistent with 1
  → 45 [45,50]

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of Y using all selected X: Y ∈ X + |XY|

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

- Select value for 1
  → 15
- Select value for 2, consistent with 1
  → 45
- Select value for 3, consistent with 1 & 2
  → 30 [20,30], 15+[10,20],45+[-20,-10]

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of Y using all selected X: Y ∈ X + |XY|

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

- Select value for 1
  → 15
- Select value for 2, consistent with 1
  → 45
- Select value for 3, consistent with 1 & 2
  → 30 [25,30]

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of Y using all selected X: Y ∈ X + |XY|

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

- Select value for 1
  → 15
- Select value for 2, consistent with 1
  → 45
- Select value for 3, consistent with 1 & 2
  → 30 [25,30]

Solution by Decomposition

- Can assign variables in any order, without backtracking.
- Tighten bound of Y using all selected X: Y ∈ X + |XY|

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

d-graph

⇨ **$O(N^2)$**

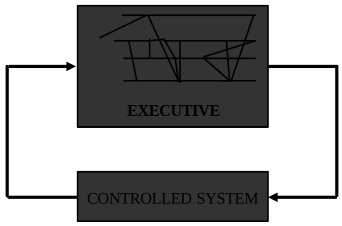
- Select value for 1
  → 15
- Select value for 2, consistent with 1
  → 45
- Select value for 3, consistent with 1 & 2
  → 30
- Select value for 4, consistent with 1,2 & 3

Outline

- Review: Constraint-based Interval Planning
- Simple Temporal Networks
- Temporal Consistency and Scheduling
- Execution Under Uncertainty

Executing Flexible Temporal Plans
[Muscettola, Morris, Pell et al.]

Handling delays and fluctuations in task duration:
• Least commitment temporal plans leave room to adapt.



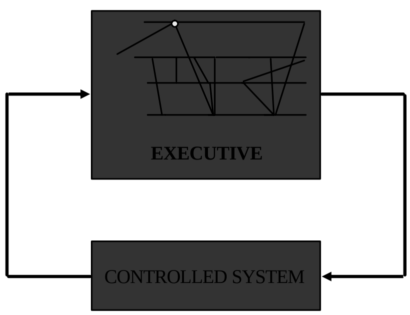
Flexible execution adapts through dynamic scheduling.
• Assigns time to event when executed.

Issues in Flexible Execution

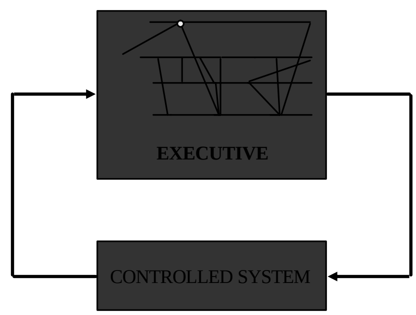
1. How do we minimize execution latency?

2. How do we schedule at execution time?

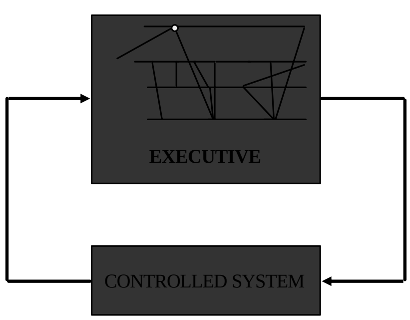
Time Propagation Can Be Costly



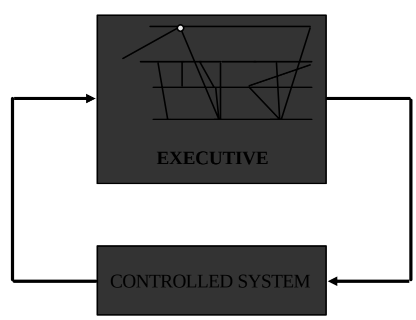
Time Propagation Can Be Costly



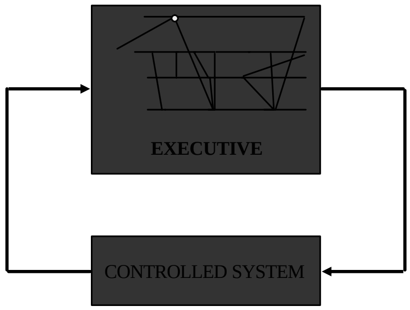
Time Propagation Can Be Costly



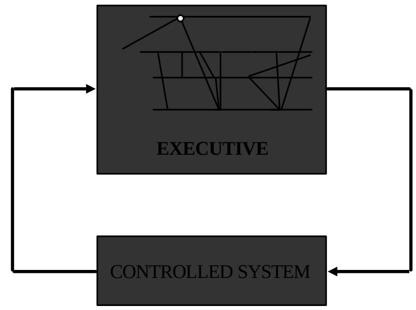
Time Propagation Can Be Costly



Time Propagation Can Be Costly



Time Propagation Can Be Costly

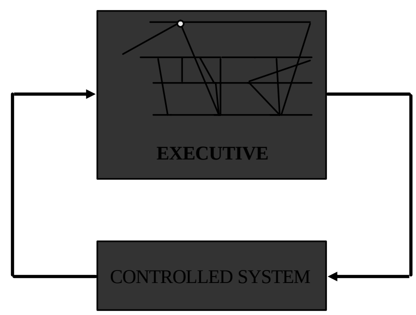


Issues in Flexible Execution

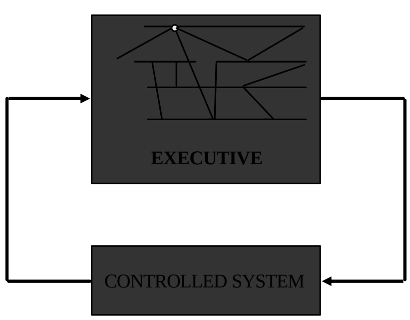
1. How do we minimize execution latency?
   ➔ Propagate through a small set of neighboring constraints.

2. How do we schedule at execution time?

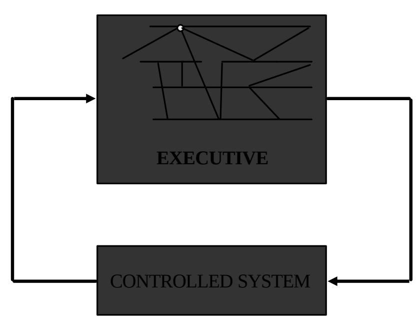
Compile to Efficient Network



Compile to Efficient Network



Compile to Efficient Network



Issues in Flexible Execution

1. How do we minimize execution latency?
   ➔ Propagate through a small set of neighboring constraints.

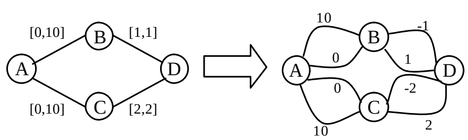
2. How do we schedule at execution time?

Issues in Flexible Execution

1. How do we minimize execution latency?
   ➔ Propagate through a small set of neighboring constraints.

2. How do we schedule at execution time?
   ➔ Through decomposition?

Dynamic Scheduling by Decomposition

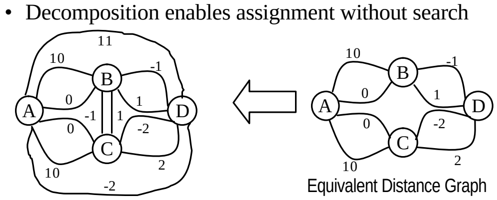


Simple Example

Equivalent Distance Graph Representation

Dynamic Scheduling by Decomposition

- Compute APSP graph
- Decomposition enables assignment without search

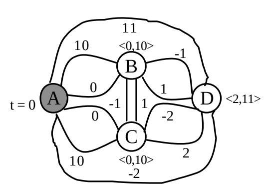


Equivalent All Pairs Shortest Paths (APSP) Representation

Equivalent Distance Graph Representation

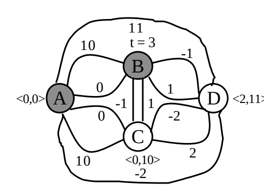
Assignment by Decomposition

- Select executable timepoint and assign
- Propagate assignment to neighbors



Assignment by Decomposition

- Select executable timepoint and assign
- Propagate assignment to neighbors



Assignment by Decomposition

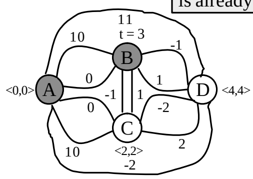
- Select executable timepoint and assign
- Propagate assignment to neighbors

But C now has to be executed at t =2, which is already in the past!

Solution:

• Assignments must monotonically increase in value.

➔ First execute all APSP neighbors with negative delays.



Dispatching Execution Controller

Execute an event when enabled and active

- Enabled - APSP Predecessors are completed
  - Predecessor – a destination of a negative edge that starts at event.
- Active - Current time within bound of task.

Dispatching Execution Controller

Initially:
- E = Time points w/o predecessors
- S = {}

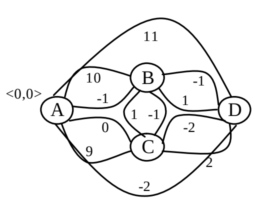
Repeat:
1. Wait until current_time has advanced st
   a. Some TP in E is active
   b. All time points in E are still enabled.
2. Set TP's execution time to current_time.
3. Add TP to S.
4. Propagate time of execution to TP's APSP immediate neighbors.
5. Add to A, all immediate neighbors that became enabled.
   a. TPx enabled if all negative edges starting at TPx have their destination in S.

Propagation is Focused

- Propagate forward along positive edges to tighten upper bounds.
  - forward prop along negative edges is useless.
- Propagate backward along negative edges to tighten lower bounds.
  - Backward prop along positive edges useless.

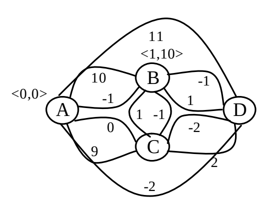
Propagation Example

S = {A}



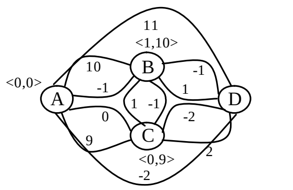
Propagation Example

S = {A}



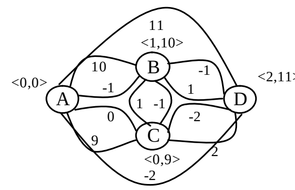
Propagation Example

S = {A}



Propagation Example

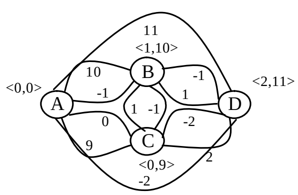
S = {A}



Propagation Example

S = {A}

E = { C }



Reducing Execution Latency

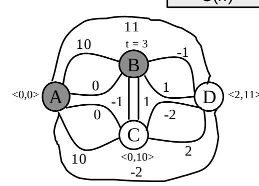
Filtering:

- some edges are redundant
- remove redundant edges

Execution time is:

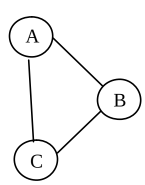
- worst case O(n)
- best case

  O(n)



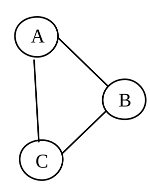
Edge Domination

- BC upper-dominates AC if in every consistent execution, $T_B + b(B,C) \leq T_A + b(A,C)$
  - The thread running through A-B-C is **always** just as fast or faster than the thread running through A-C

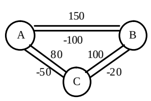


Edge Domination

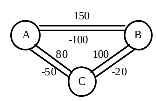
- AB lower-dominates AC if in every consistent execution, $T_B - b(A,B) \geq T_C - b(A,C)$
  - Enablement of node A is always determined by thread running through A-B-C



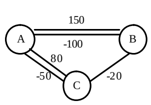
- Edge Dominance
  - Eliminate edge that is redundant due to the triangle inequality AB + BC = AC



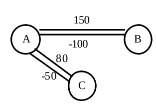
- Edge Dominance
  - Eliminate edge that is redundant due to the triangle inequality AB + BC = AC



- Edge Dominance
  - Eliminate edge that is redundant due to the triangle inequality AB + BC = AC

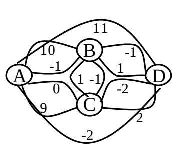


- Edge Dominance
  - Eliminate edge that is redundant due to the triangle inequality AB + BC = AC



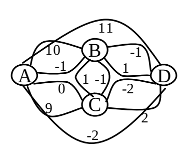
An Example of Edge Filtering

- Start off with the APSP network



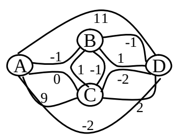
An Example of Edge Filtering

- Start at A-B-C triangle



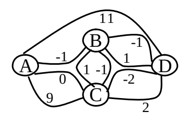
An Example of Edge Filtering

• Look at B-D-C triangle



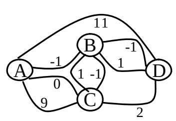
An Example of Edge Filtering

• Look at B-D-C triangle



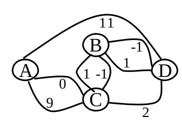
An Example of Edge Filtering

• Look at D-A-B triangle



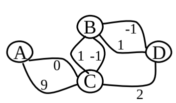
An Example of Edge Filtering

• Look at D-A-C triangle



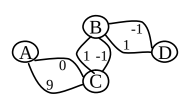
An Example of Edge Filtering

• Look at B-C-D triangle



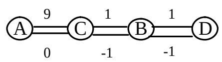
An Example of Edge Filtering

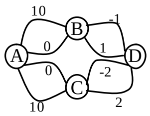
• Look at B-C-D triangle



An Example of Edge Filtering

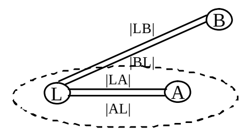
- Resulting network has less edges than the original





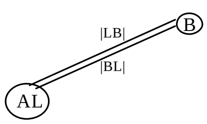
Additional Filtering

- Node Contraction
  - Collapse two events with fixed time between them



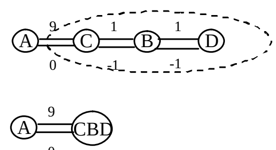
Additional Filtering

- Node Contraction
  - Collapse two events with fixed time between them



An Example of Node Contraction

- Resulting network has less edges than the original



Avoiding Intermediate Graph Explosion

Problem:

- APSP consumes $O(n^2)$ space.

Solution:

- Interleave process of APSP construction with edge elimination
  - Never have to build whole APSP graph