Massachusetts Institute of Technology

# Diagnosing Over-subscribed Temporal Problems

16.412, February 25th, 2015

Peng Yu

Model-based Embedded & Robotic Systems

CSAIL

# Reminder



- Thought Exercise 2:
  - Due Friday 2/27.

- Assignment 2:
  - Due Friday 3/13.

- Reading:
  - Peng Yu and Brian Williams, **Continuously Relaxing Over-constrained Conditional Temporal Problems through Generalized Conflict Learning and Resolution,** In *Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI-2013)*, August 2013.

# The Problem



- We human beings tend to ask for more than what we can do.

- Hence our study plans, working schedules, and travel itineraries are often over-subscribed.
  - ‘I want to go to the theater and watch a movie on Friday.’
  - ‘Sorry, you cannot do it because you have to complete the thought exercise for 16.412 first. It will keep you busy until **11:59pm**. ’

# This Lecture

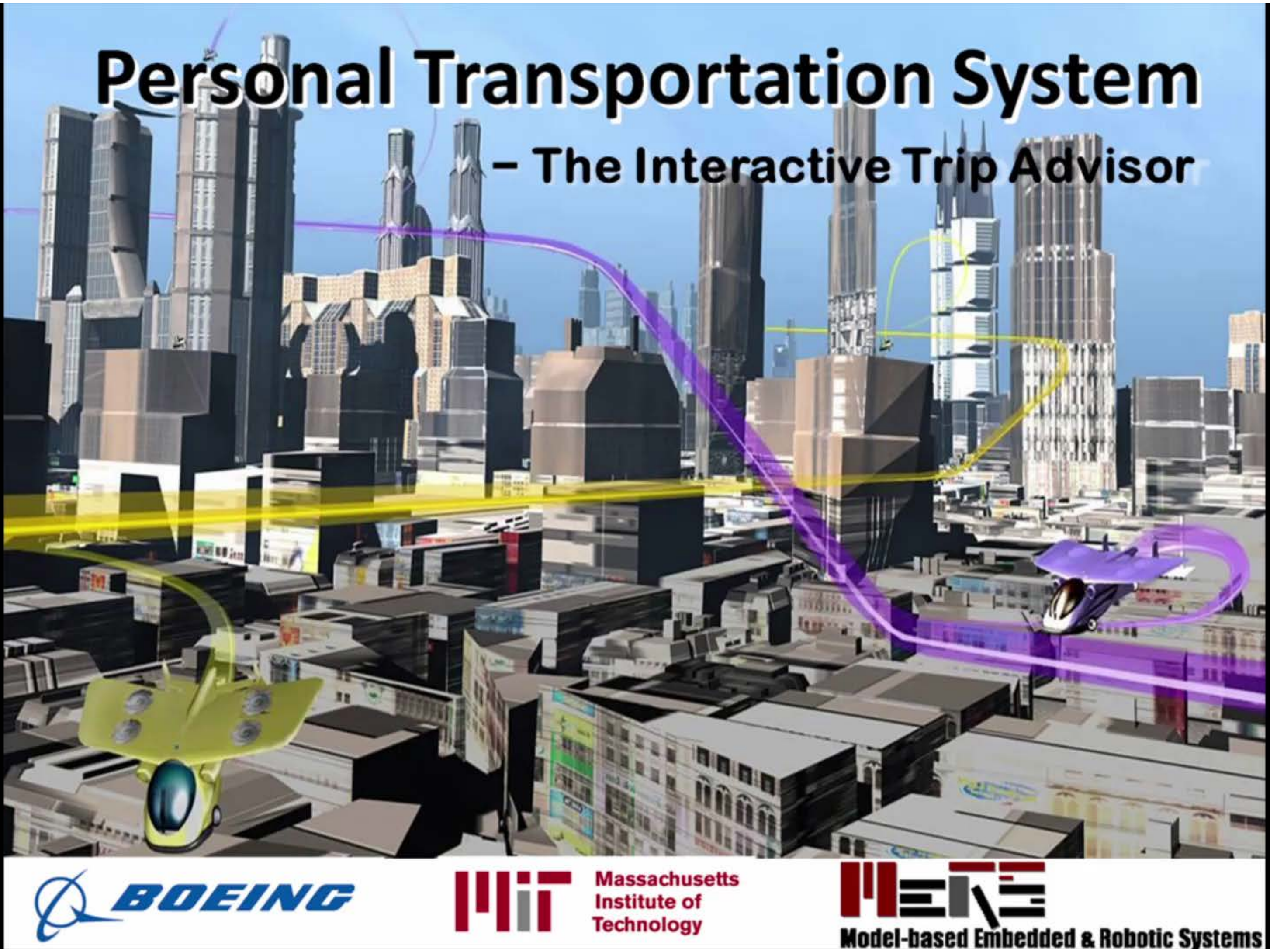


- Detect and resolve such over-subscription, using what we learned in Monday's lecture.
  - The model-based diagnosis methods can be applies to **diagnose** our travel plans.
  - The 'failure modes' are alternative goals (**goal relaxation**) that repair the broken plan.

- This lecture focuses on the diagnosis of temporal problems, which is widely used for modeling real world scheduling problems.

# Build Your Own Travel Advisor





# This Lecture



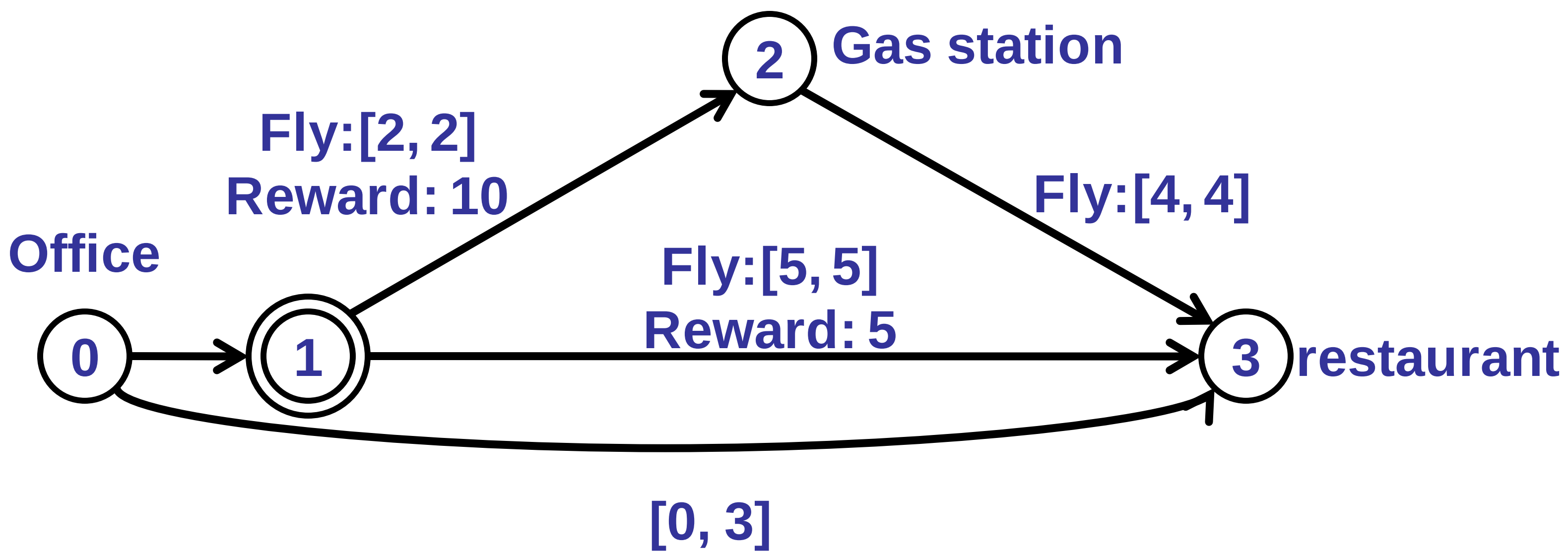
- Conflict Detection
- Discrete Relaxation
- Continuous Relaxation
- Integrate Everything Together

# Detecting Conflicts in Temporal Problems

*'Explain the cause of failure'*

# Review: Temporal Plan Network

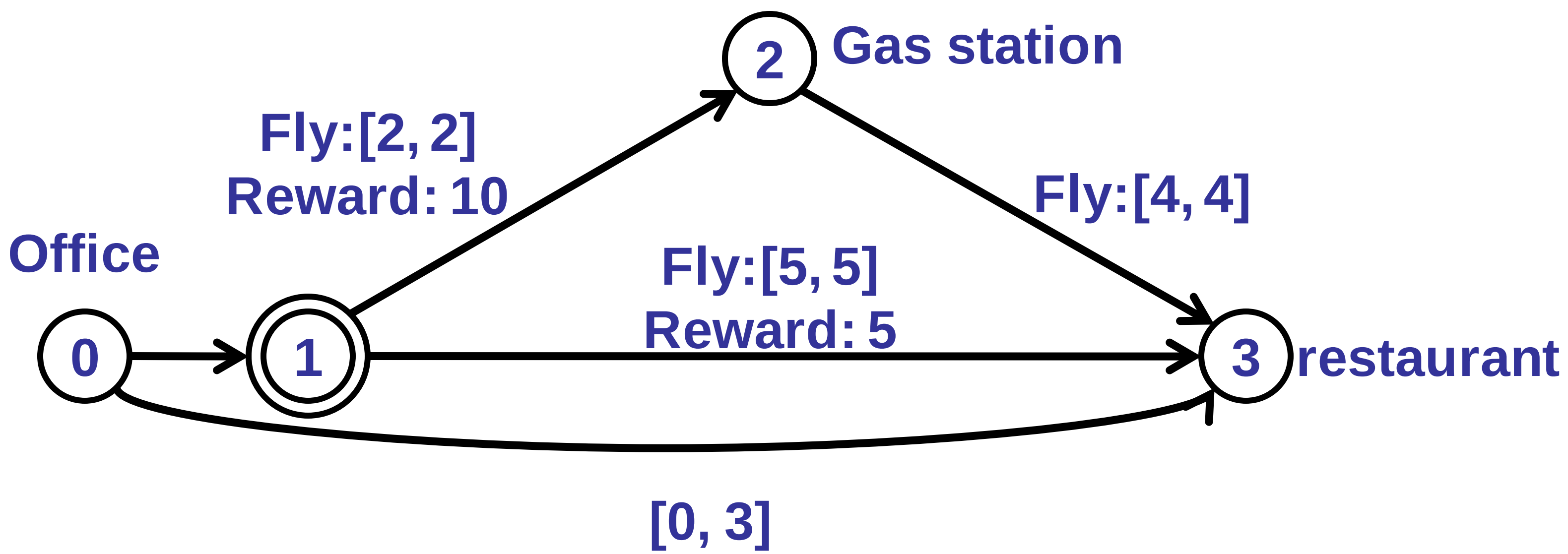
- Augmented from Simple Temporal Networks.
  - Addition of decision nodes.
  - Rewards/costs.



# Review: Solve a TPN

Model-based Embedded & Robotic Systems

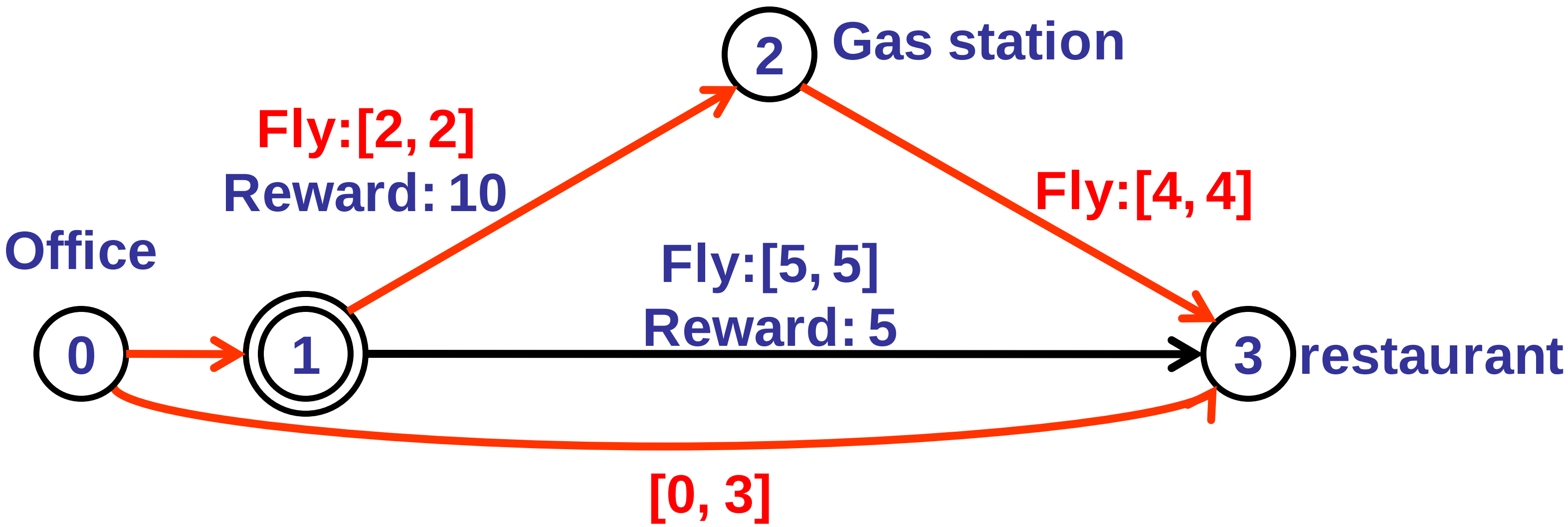
- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency.
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



# Solve a TPN

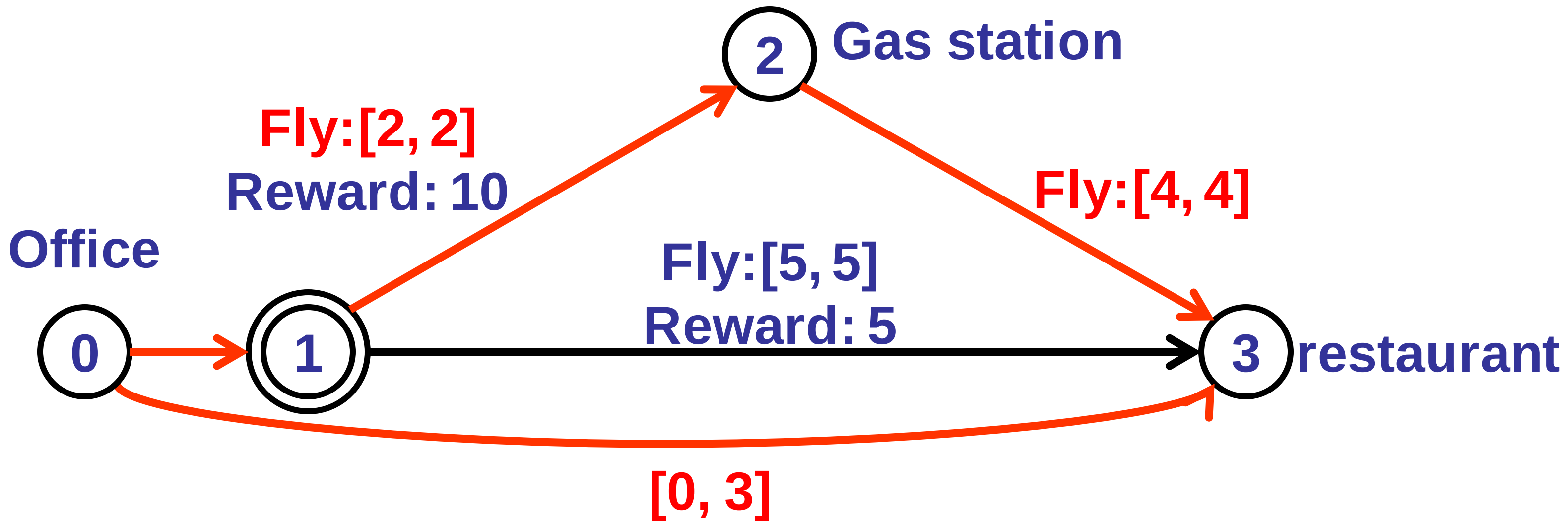


- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency.
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



# Solve a TPN

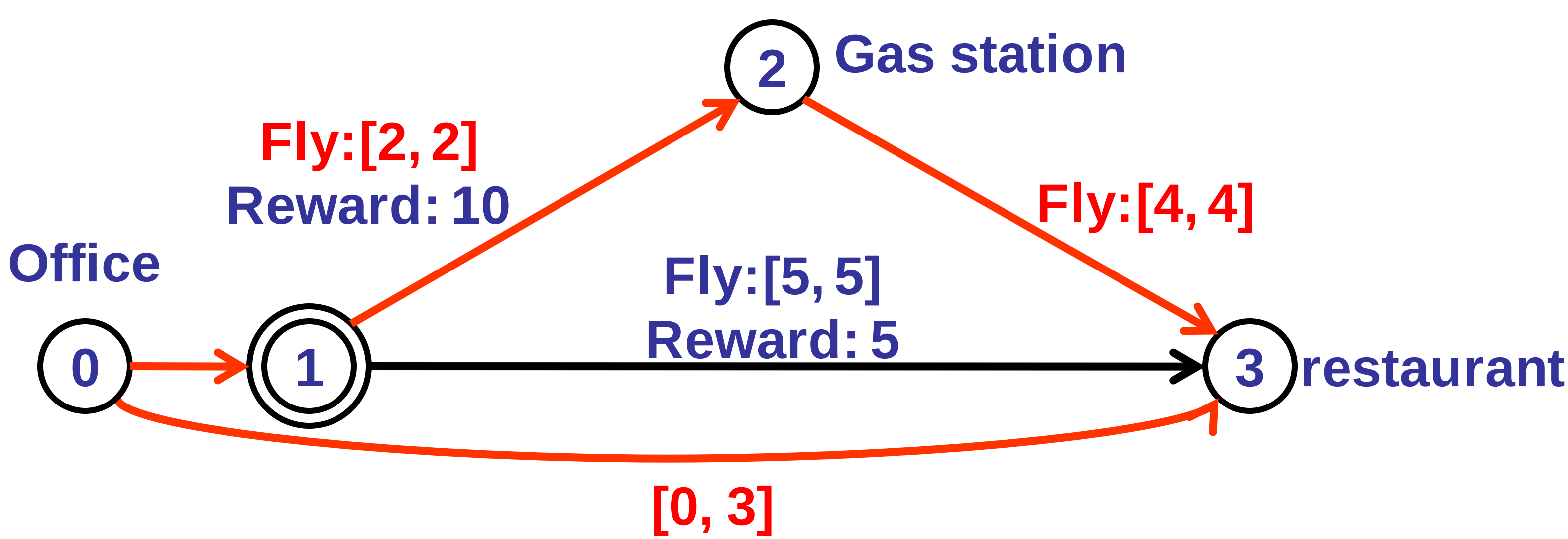
- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency.
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



# Solve a TPN



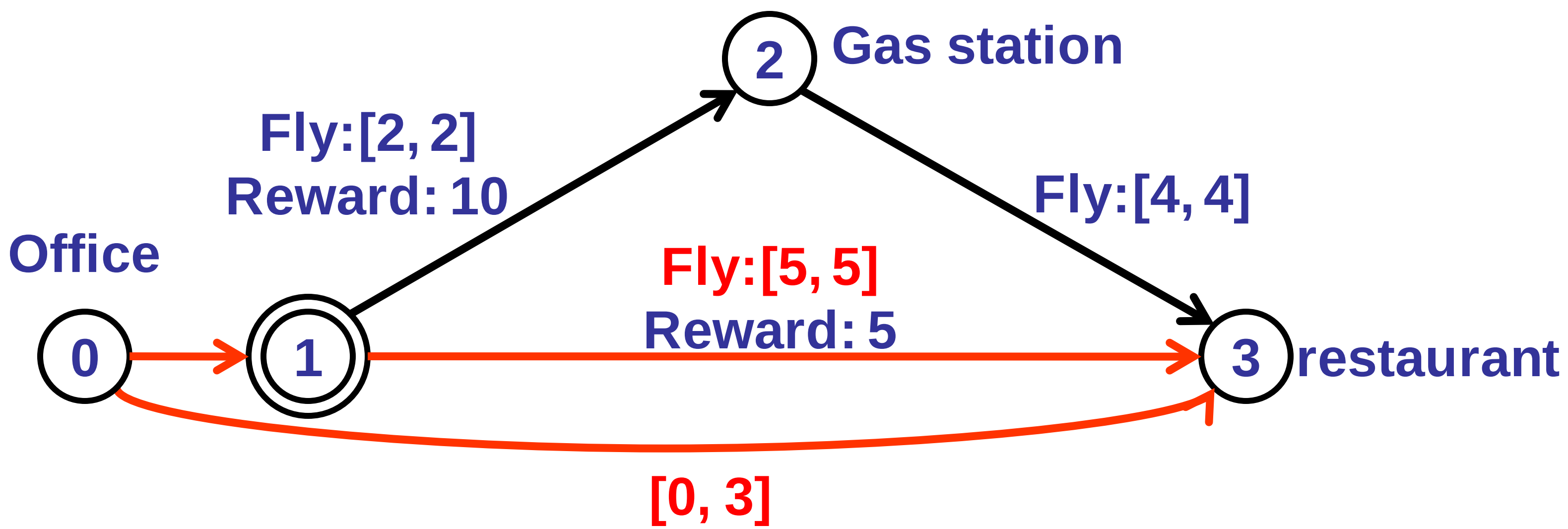
- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency. Not consistent!
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



# Solve a TPN



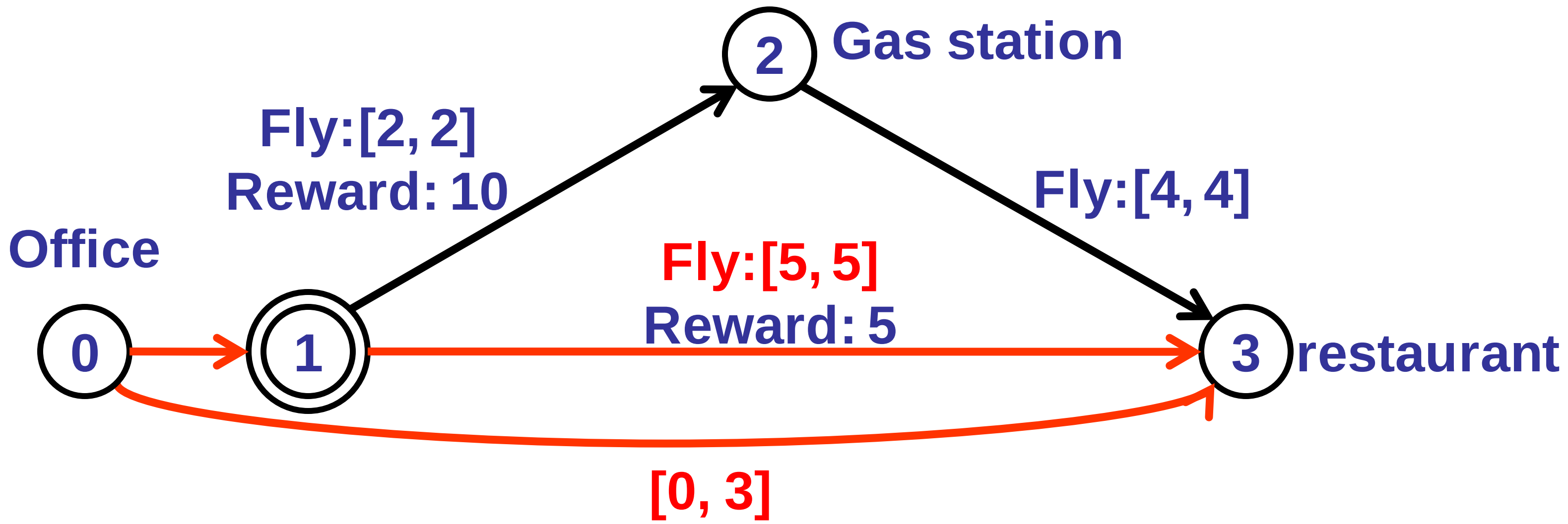
- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency.
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



# Solve a TPN



- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency. Not consistent!
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



# Explain Cause of Failure

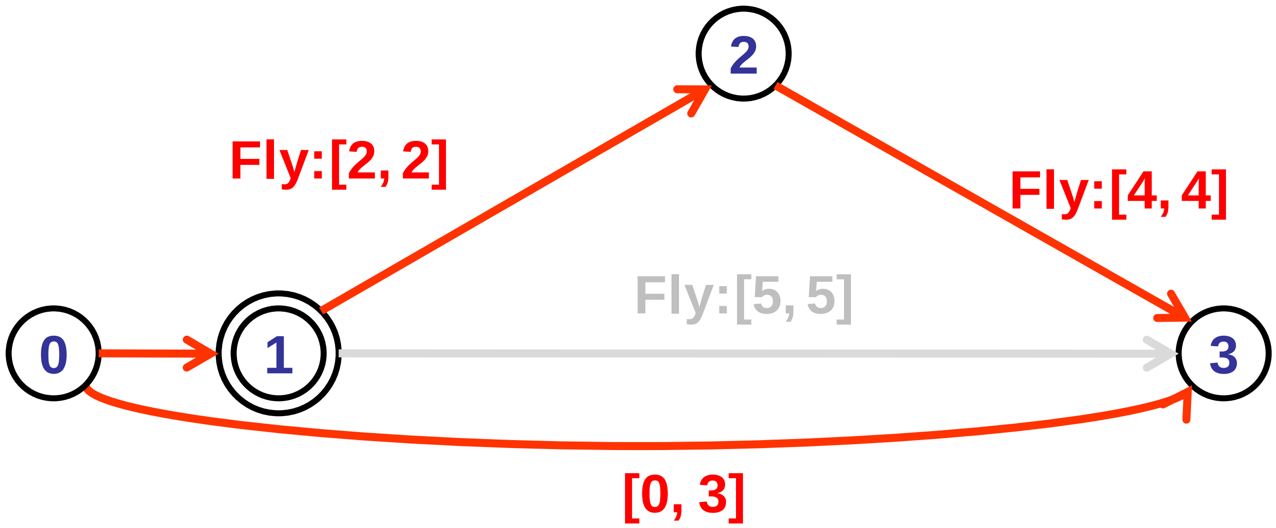


- Similar to hardware diagnoses problem, we use conflicts to explain the cause of infeasibility.
  - Conflict 1: Fly to both ‘gas station’ and ‘restaurant’
  - Conflict 2: Fly to ‘restaurant’ directly
- However, this discrete representation does not preserve useful **temporal** information.
- More often, we use **Continuous Conflict** to describe the conflicts in temporal problems.

# Discrete and Continuous Conflicts



- Continuous conflict: a set of temporal constraints that results in a negative cycle.



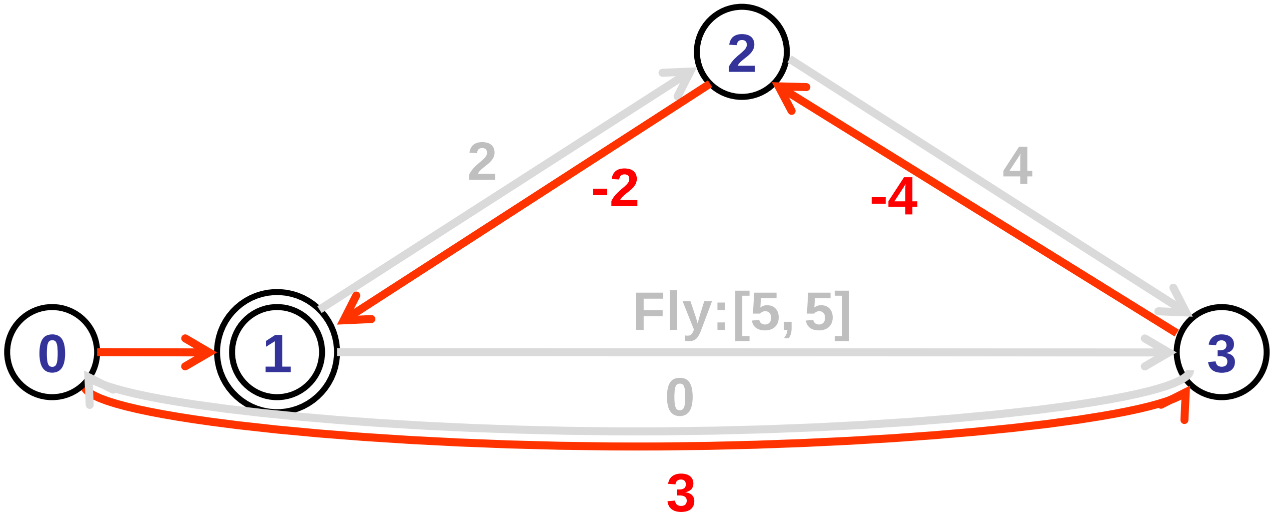

{Fly:[2,2], Fly:[4,4], Arrival[0,3]}

# Discrete and Continuous Conflicts



- More specifically, we can represent a continuous conflict as a linear expression.

$$Arrival_{UB} - Fly_{12LB} - Fly_{23LB} = -3$$

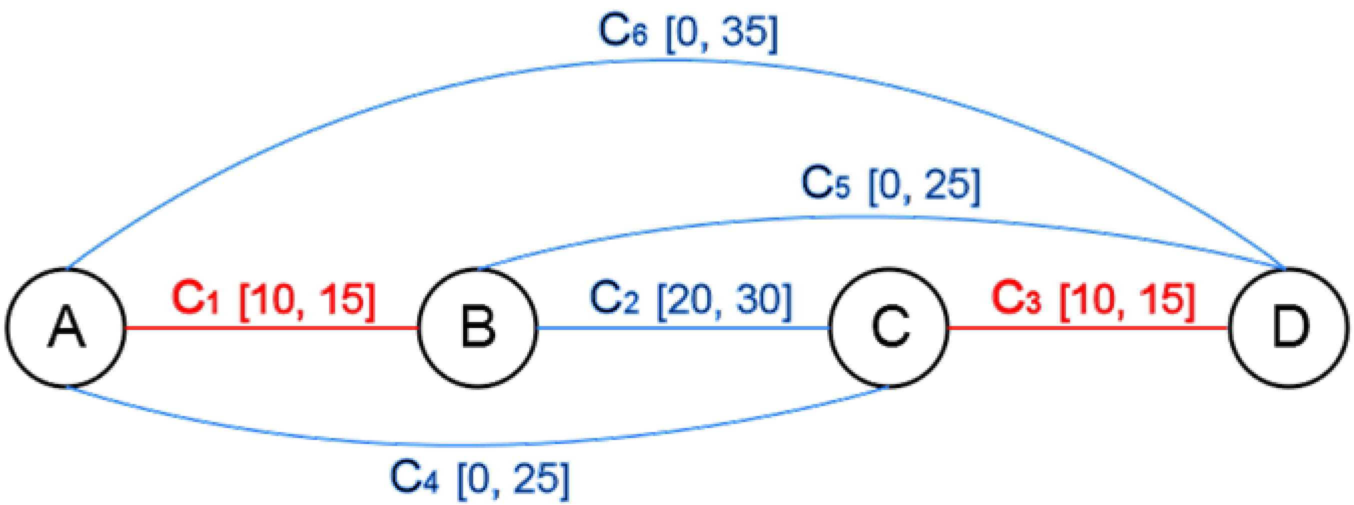



- The more information we encode in the conflict, the more options we have while repairing the problems.

# Exercise



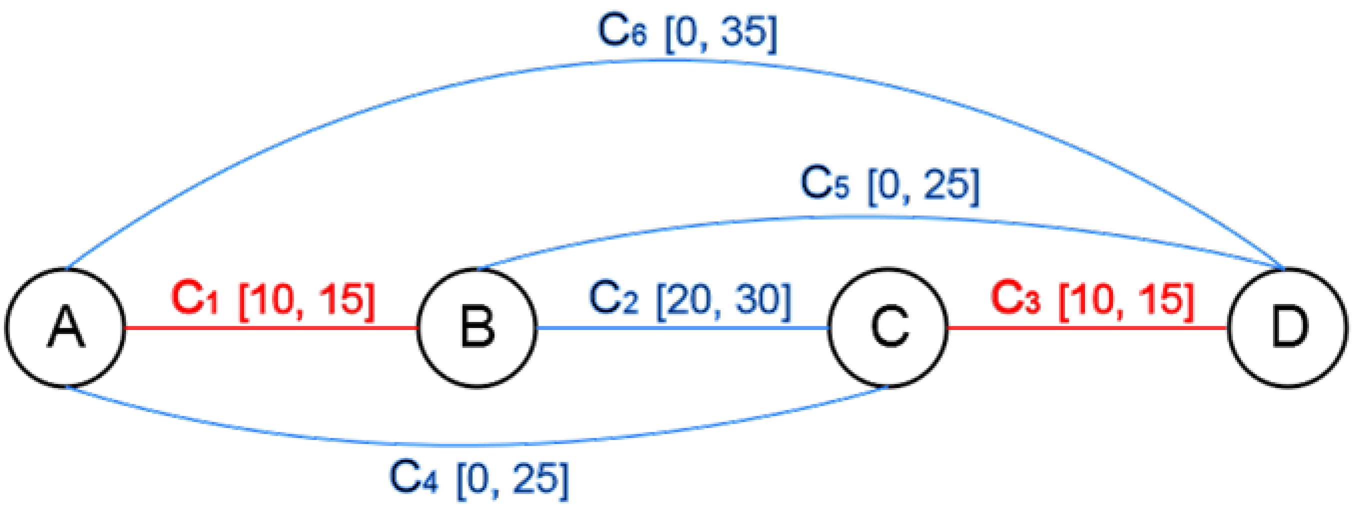
- How many conflicts does this STN have?



# Exercise



- How many conflicts does this STN have?




$$C_{4UB} - C_{1LB} - C_{2LB} = -5$$
$$C_{5UB} - C_{3LB} - C_{2LB} = -5$$
$$C_{6UB} - C_{3LB} - C_{2LB} - C_{1LB} = -5$$

# What’s next?



- Return the conflicts to the user.
- Let the user figure out the problem and input a new set of requirements.

- OR

- “I have a feasible plan if you skip all stops.”
- “If you relax your arrival constraint by only 2 minutes, I can find a solution for you.”

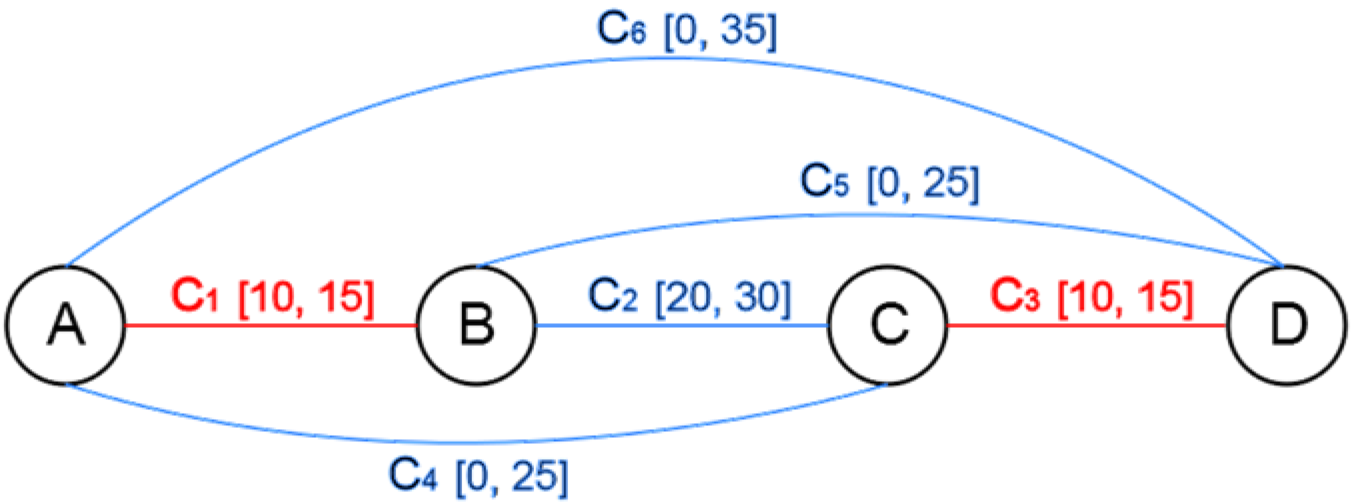
# Discrete Relaxations for Temporal Problems

*‘Repair broken problems through* ***suspending*** *constraints’*

# Problem Statement: Input



- Temporal Constraint Relaxation Problems:
  - Events.
    - With different time of occurrence.
  - Constraints.
    - Base: C1, C3.
    - Relaxable: C2, C4, C5, C6.



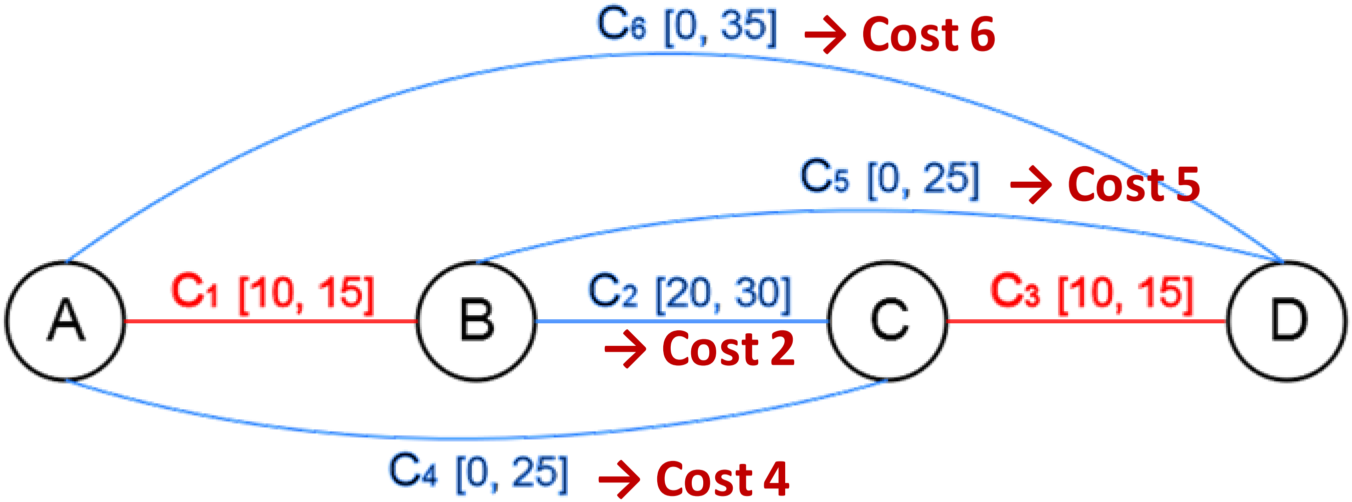
# Problem Statement: Preferences



- Discrete relaxation cost:

$$f : C_i \rightarrow \mathbb{R}$$

  – Activated when constraints are suspended.
    - Suspend C2: Cost 2.
    - Suspend C4, C6: Cost 10.

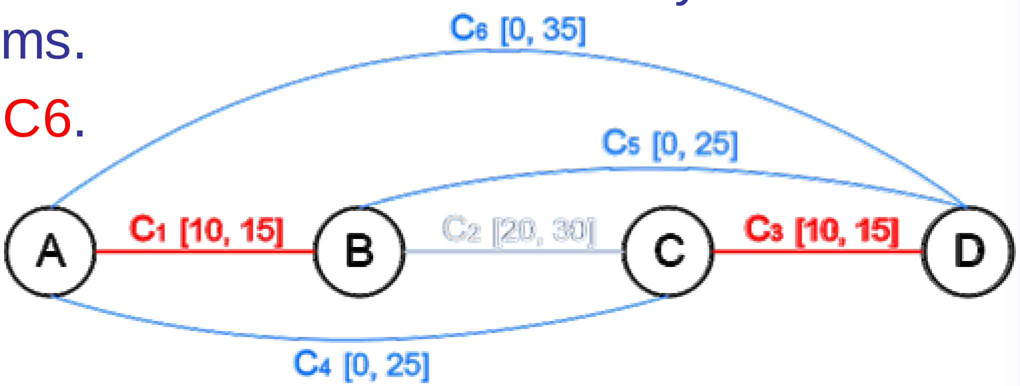


# Output: Minimal relaxation



- Relaxation:
  - A set of relaxed constraints that resolve the inconsistency of over constrained temporal problems.
    - Suspending C4, C5 and C6.
    - Suspending C2, C6.
    - Suspending C2.

    … …
  - All supersets of a relaxation are valid relaxations, too.




- Minimal relaxation:
  - A valid relaxation.
  - None of its proper subsets are valid relaxations.
    - {C2=**sus**} vs. {C2=**sus**, C6=**sus**} .

# Best-First

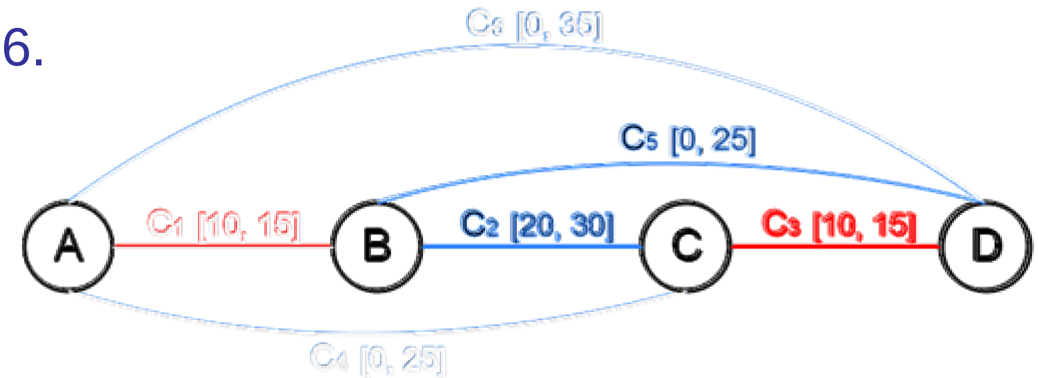


- Cost of relaxations:
  - Suspending constraints will incur costs:
  - {C2=**sus**, C5=**sus**} (7) vs. {C2=**sus**, C4=**sus**} (6).

- Candidate relaxations with lower cost will be returned first.

# Conflict-Directed

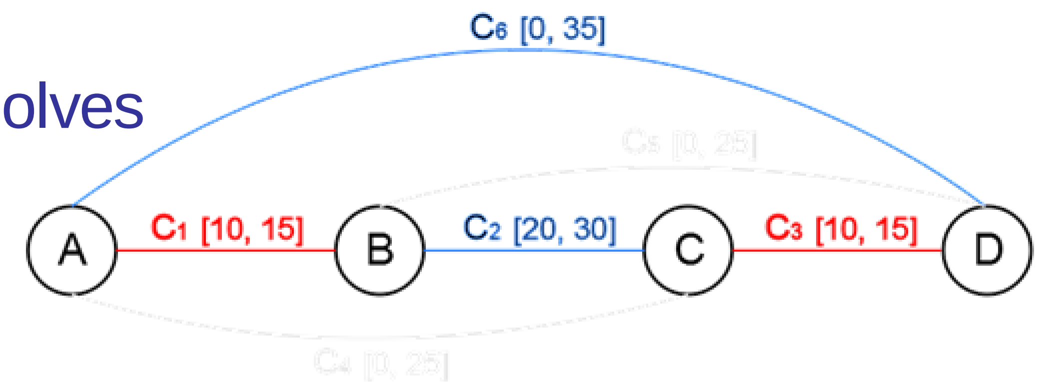
- A set of inconsistent active constraints.
  - C1=**act**,C2=**act**,C3=**act**,C6=**act**.
  - C1, C2, C3, C4, C5, C6.
  - C2, C3, C5.
  - … …




- Minimal conflicts.
  - A conflict whose proper subsets are not in conflict.
    - C2=**act**, C3=**act**, C5=**act**.
    - C1=**act**, C2=**act**, C3=**act**, C6=**act**.
    - C1=**act**, C2=**act**, C4=**act**.

# Conflicts and Relaxations

- Relaxing one constraint can resolve a minimal conflict.
  - Suspending C6 resolves {C1=**act**, C2=**act**, C3=**act**, C6=**act**}.



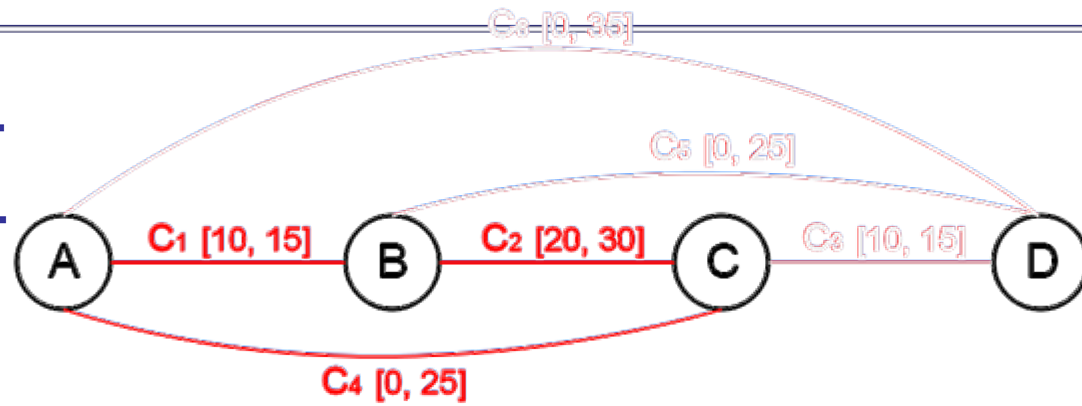

  - A relaxation suspends at least one constraint in each minimal conflict (a covering set).

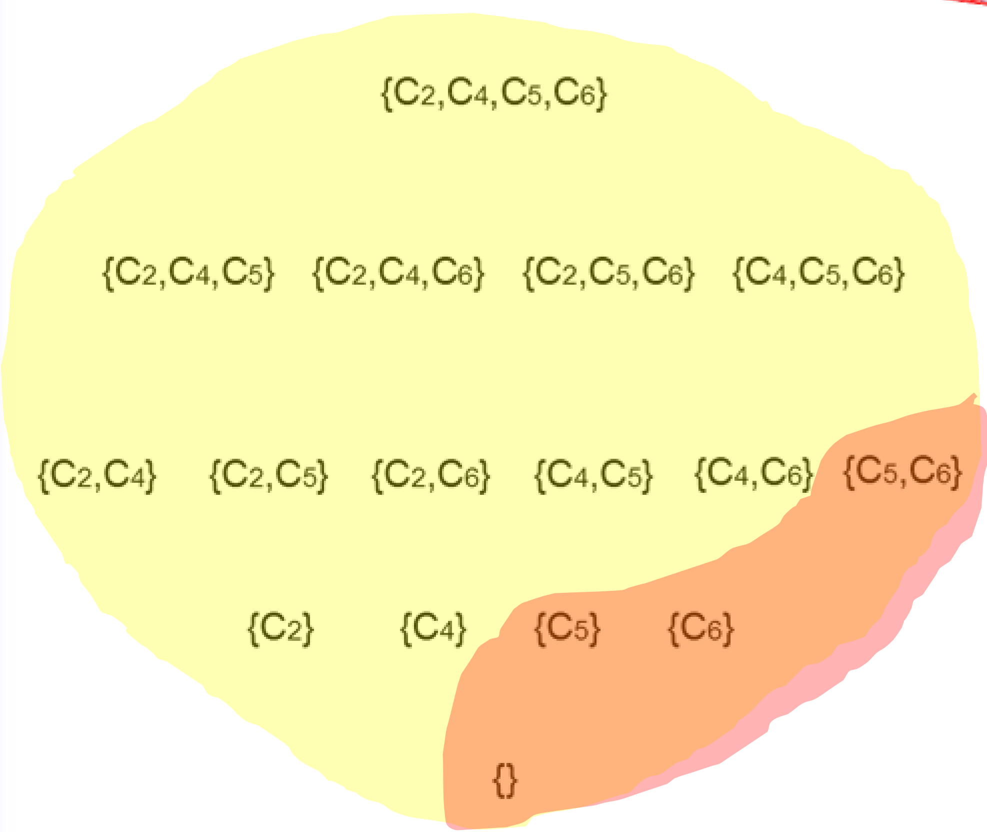
| | |
| --- | --- |
| **{C2=act, C3=act, C5=act}** | **{C2=sus, C1=sus, C4=sus}** |
| **{C1=act, C2=act, C3=act, C6=act}** | **{C2=sus}** |
| **{C1=act, C2=act, C4=act}** | **… …** |

# Enumerate Minimal Relaxations



- 4 relaxable constraints.
- 16 possible relaxations.
- Q = {C2=**sus**},{C4=**sus**}



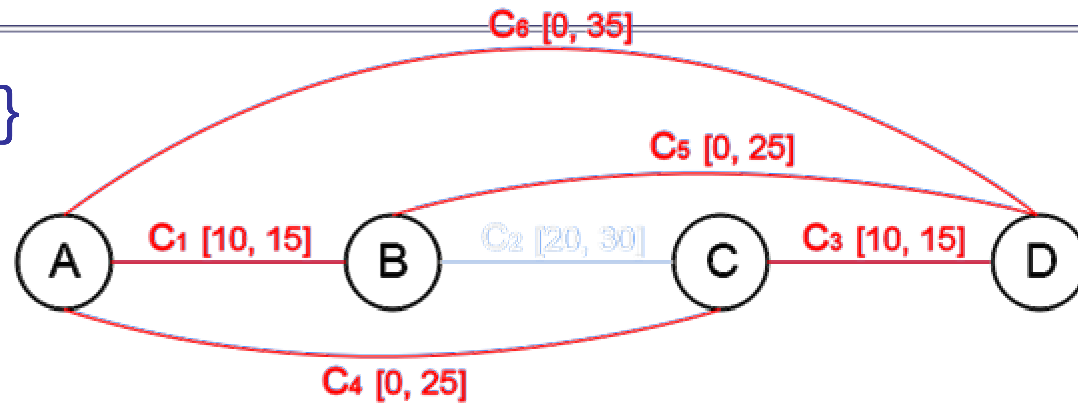



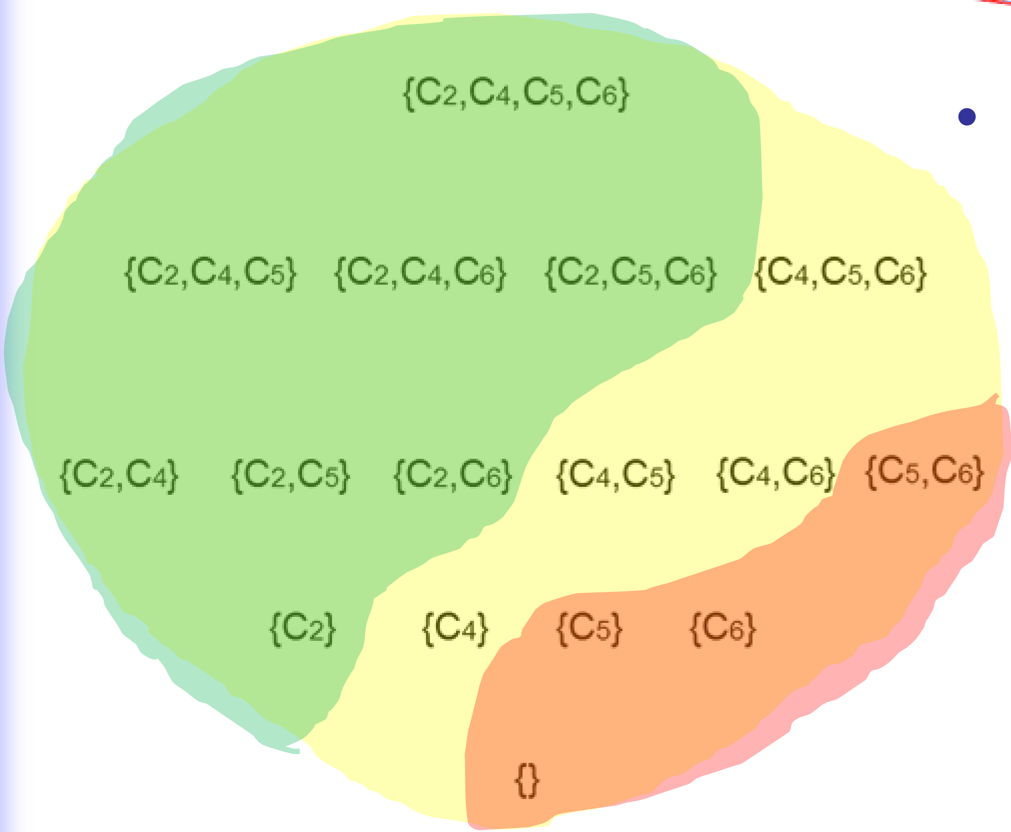

- Start with {}.
  - Covers all relaxations.

  Test {C2=act,C4=act,C5=act,C6=act}.
  - Inconsistent!
  - Extract minimal conflict {C1=act,C2=act,C4=act}.
  - Candidate minimal relaxations: {C2=sus},{C4=sus}.

# Enumerate Minimal Relaxations

- Q = {C2=sus},{C4=sus}



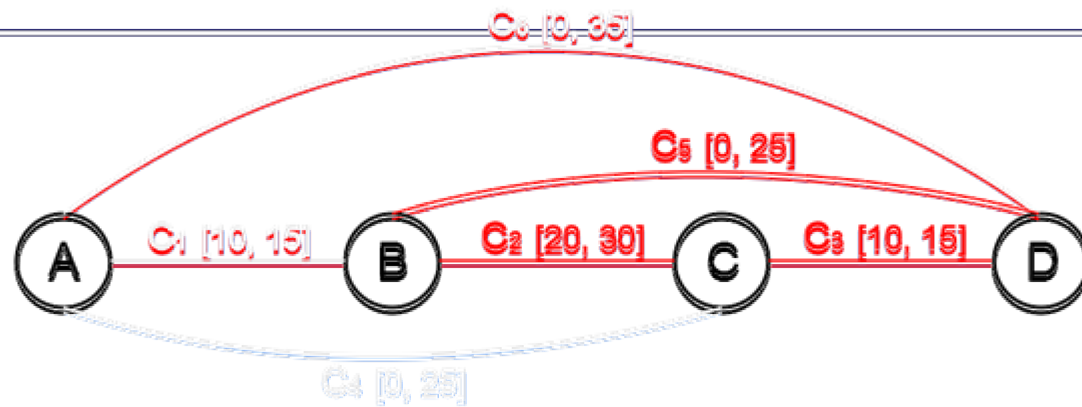



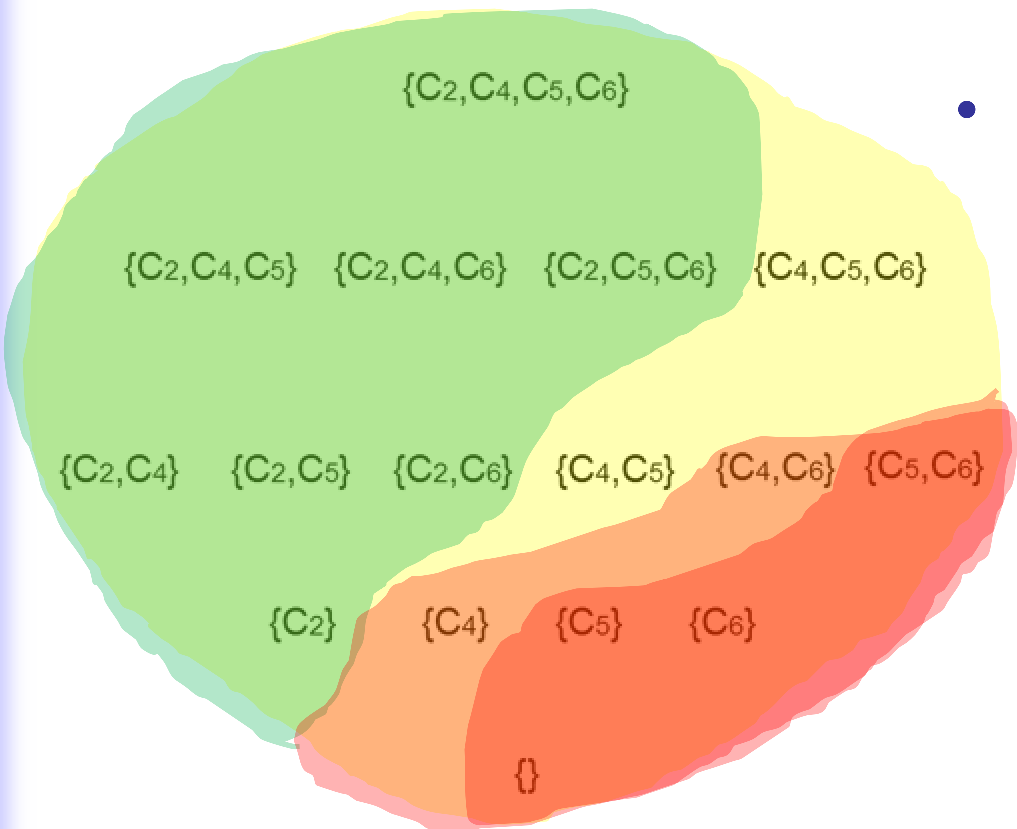

- Check {C2=sus}
  - Test {C2=sus,C4=act,C5=act,C6=act}.
  - Consistent!
  - All supersets of {C2=sus} are valid relaxations.

# Enumerate Minimal Relaxations

Model-based Embedded & Robotic Systems

- Q = {C4=sus}
  {C4=sus,C5=sus}



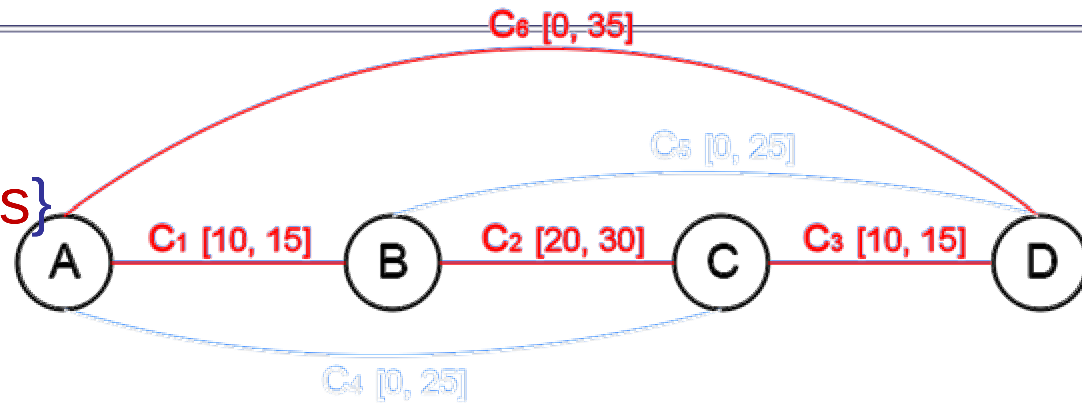

- Check {C4=sus}
  - Test {C2=act,C4=sus,C5=act,C6=act}.
  - Inconsistent!
  - Extract minimal conflicts {C2=act,C3=act,C5=act}, {C1=act,C2=act,C4=act}.
  - Candidate minimal relaxations: {C4=sus, C5=sus}.

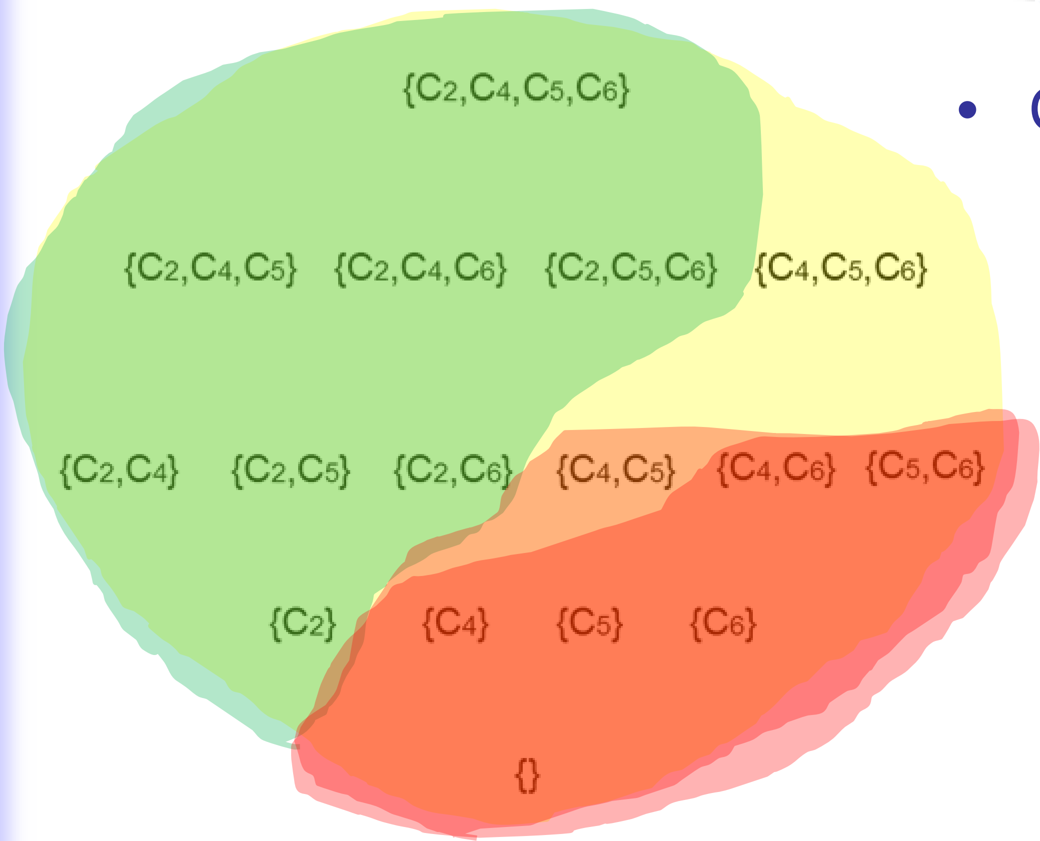
# Enumerate Minimal Relaxations



- Q = {C4=sus,C5=sus}

{C4=sus, C5=sus, C6=sus}

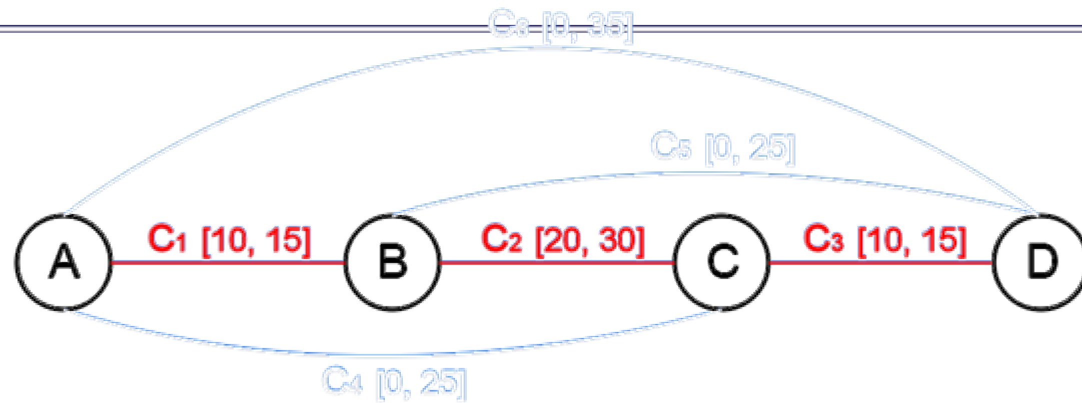


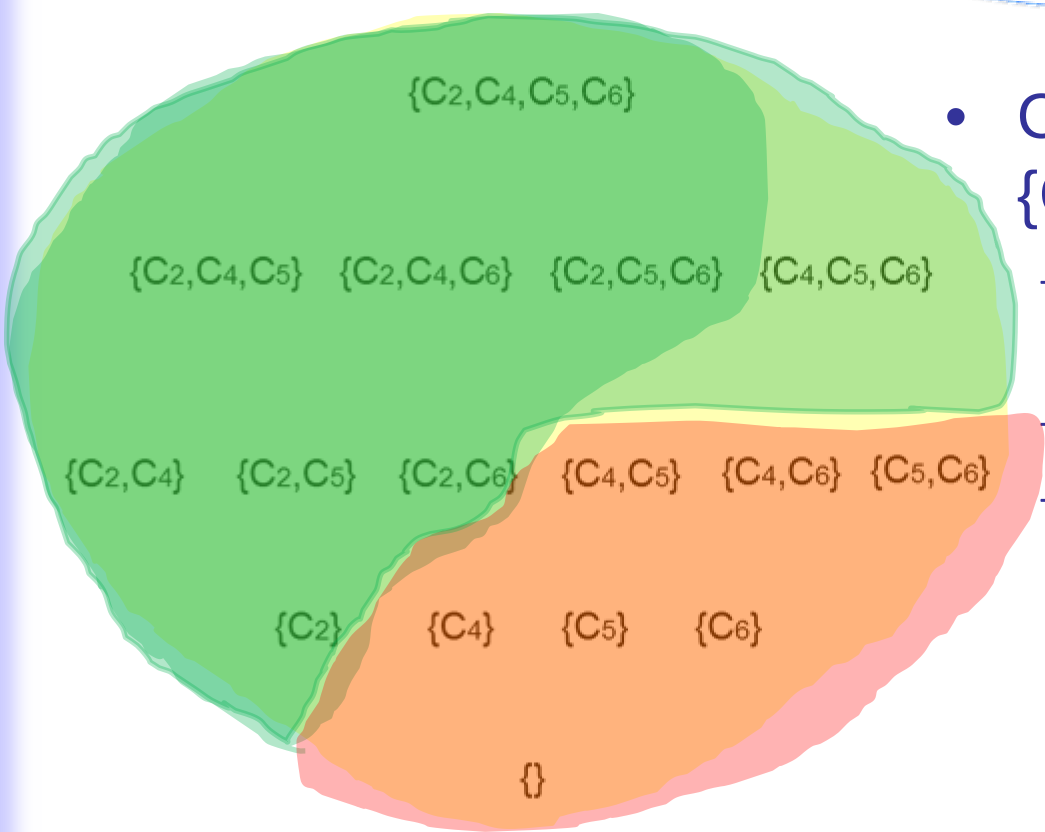




- Check {C4=sus,C5=sus}
  - Test {C2=act,C4=sus,C5=sus,C6=act}.
  - Inconsistent!
  - Extract conflicts {C1=act, C2=act, C3=act, C6=act}.
  - Candidate minimal relaxations: {C4=sus, C5=sus, C6=sus}.

# Enumerate Minimal Relaxations

- Q = {C4=sus, C5=sus, C6=sus}




- Check {C4=sus,C5=sus,C6=sus}
  - Test {C2=act,C4=sus,C5=sus,C6=sus}.
  - Consistent!
  - No more candidate relaxations left, search complete!

# Problem With Discrete Relaxation



- Discrete relaxations often make modifications that are too aggressive to the problems.
  - *'Do not stop at the gas station or the restaurant'.*

- Usually, we would expect better suggestions that do not make unnecessary changes, in order to minimize the perturbations to our goals.

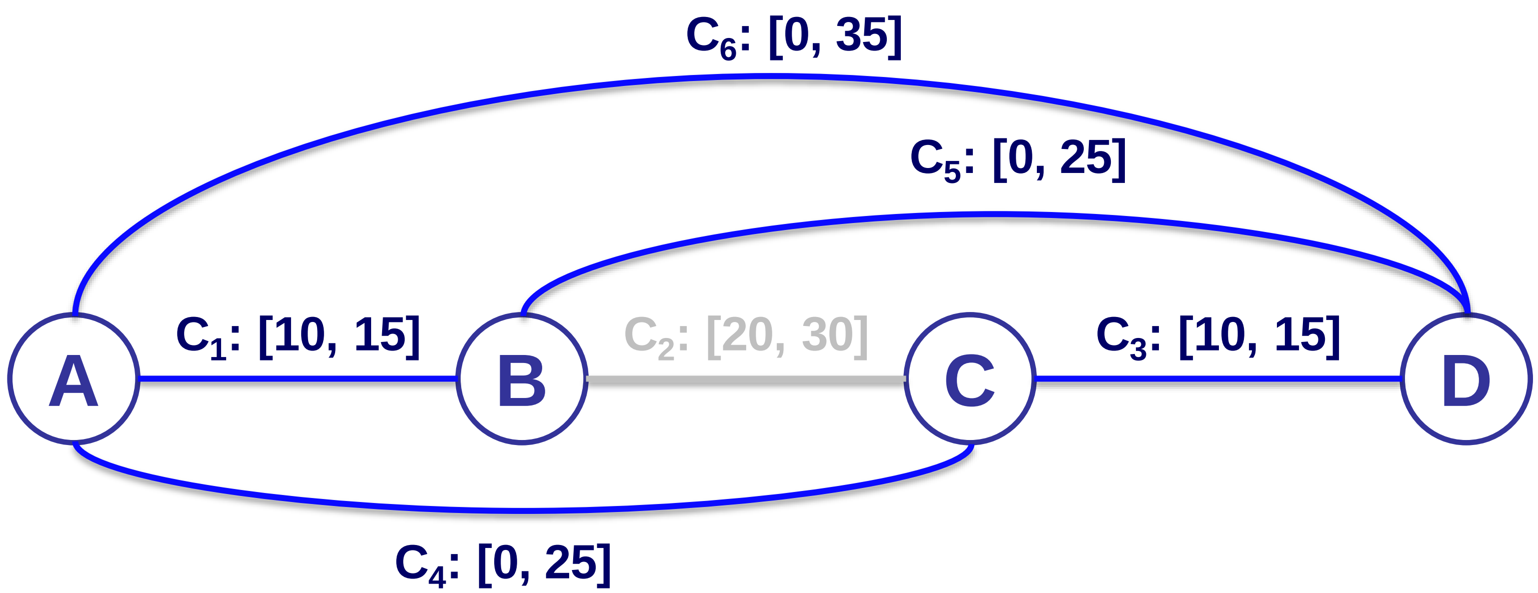
# Continuous Relaxations for Temporal Problems

*'Repair broken problems through* ***weakening*** *constraints'*

# Discrete Relaxation

Model-based Embedded & Robotic Systems

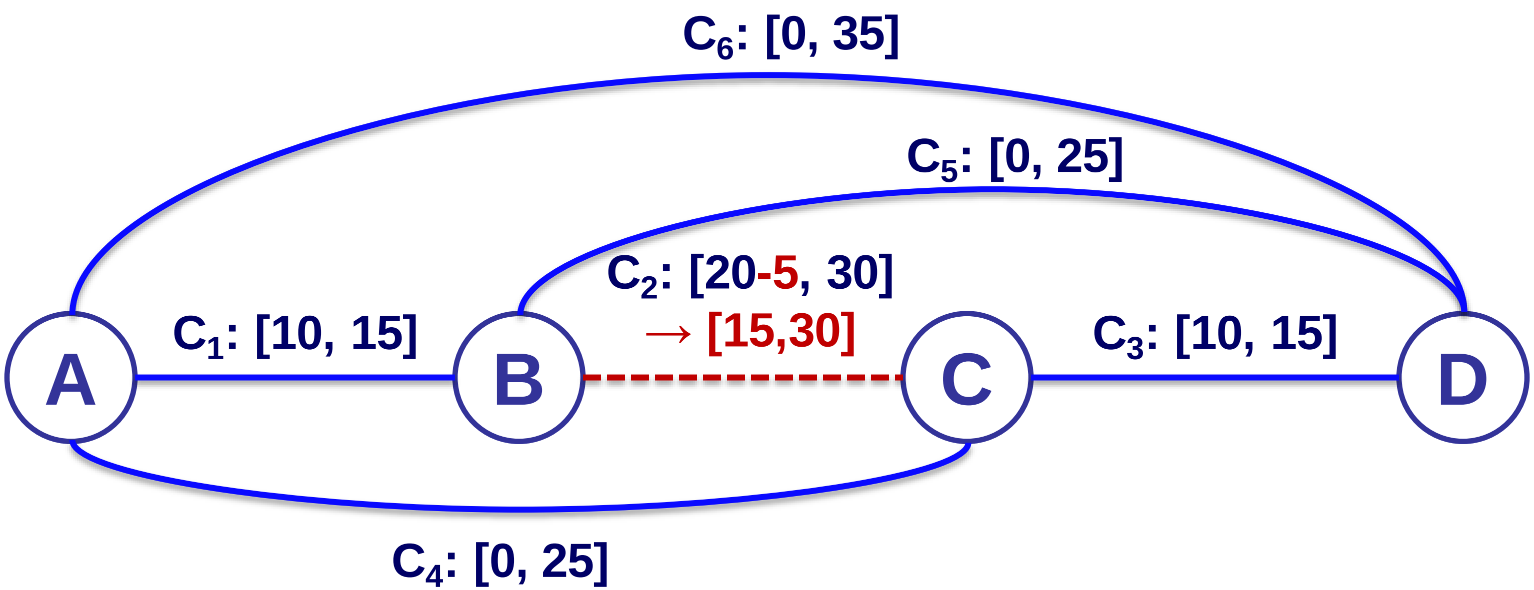
- Identify a minimal set of constraints that have to be dropped.



# Continuous relaxation



- Introduce slack variables and preference functions over the deviations of temporal bounds.
  - $\Delta C_{1LB}$, $\Delta C_{1UB}$, $\Delta C_{2LB}$, $\Delta C_{2UB}$,…

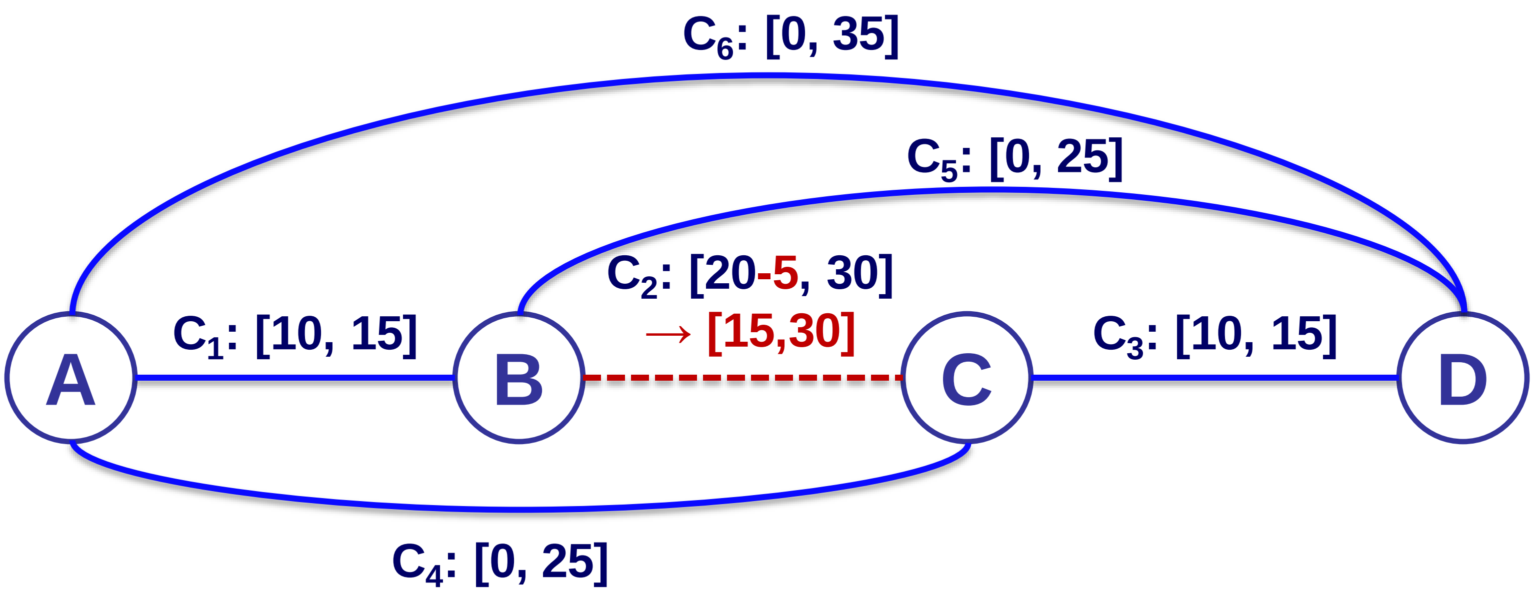



- Given an over-constrained temporal problem, find the most preferred continuous relaxation.

# Preferences

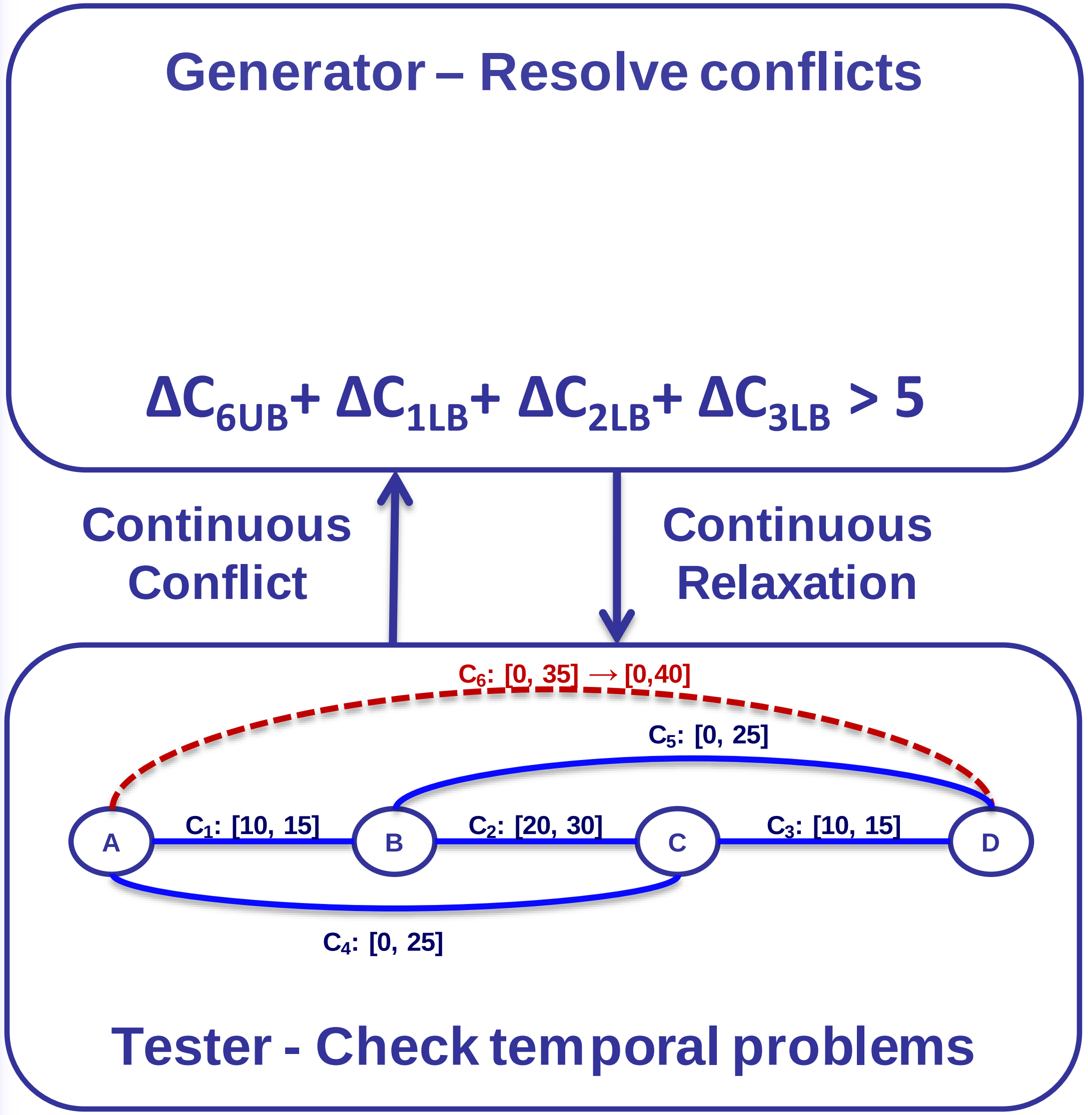


- We define linear cost functions over the deviations from the original temporal bounds.
  - $\Delta C_{1LB}$: \$1/minutes; $\Delta C_{2LB}$: \$1.5/minutes
  - Minimize($\Delta C_{1LB}+1.5\Delta C_{2LB}+\Delta C_{3LB}+\Delta C_{4UB}+3\Delta C_{5UB}+3\Delta C_{5UB}$)



# Approach – Generate and Test



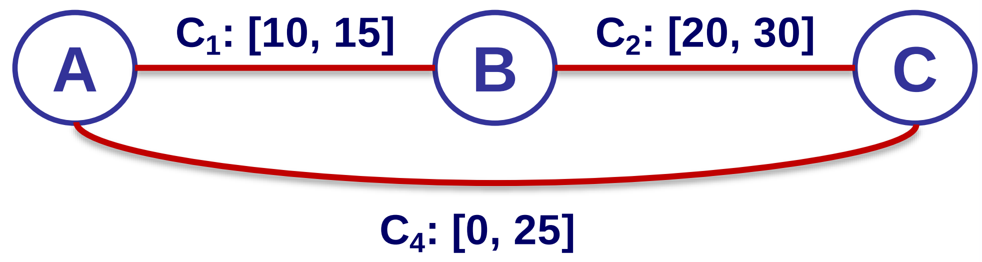



- We take a generate and test approach.
- The **generator** computes new continuous relaxations to resolve known conflicts.
- The **tester** checks the consistency of the relaxed problem, and extract new conflicts.

# From Continuous Conflicts to Constraints



- A conflict composes of an inconsistent set of temporal constraints, such as $\{C_1, C_2, C_4\}$.



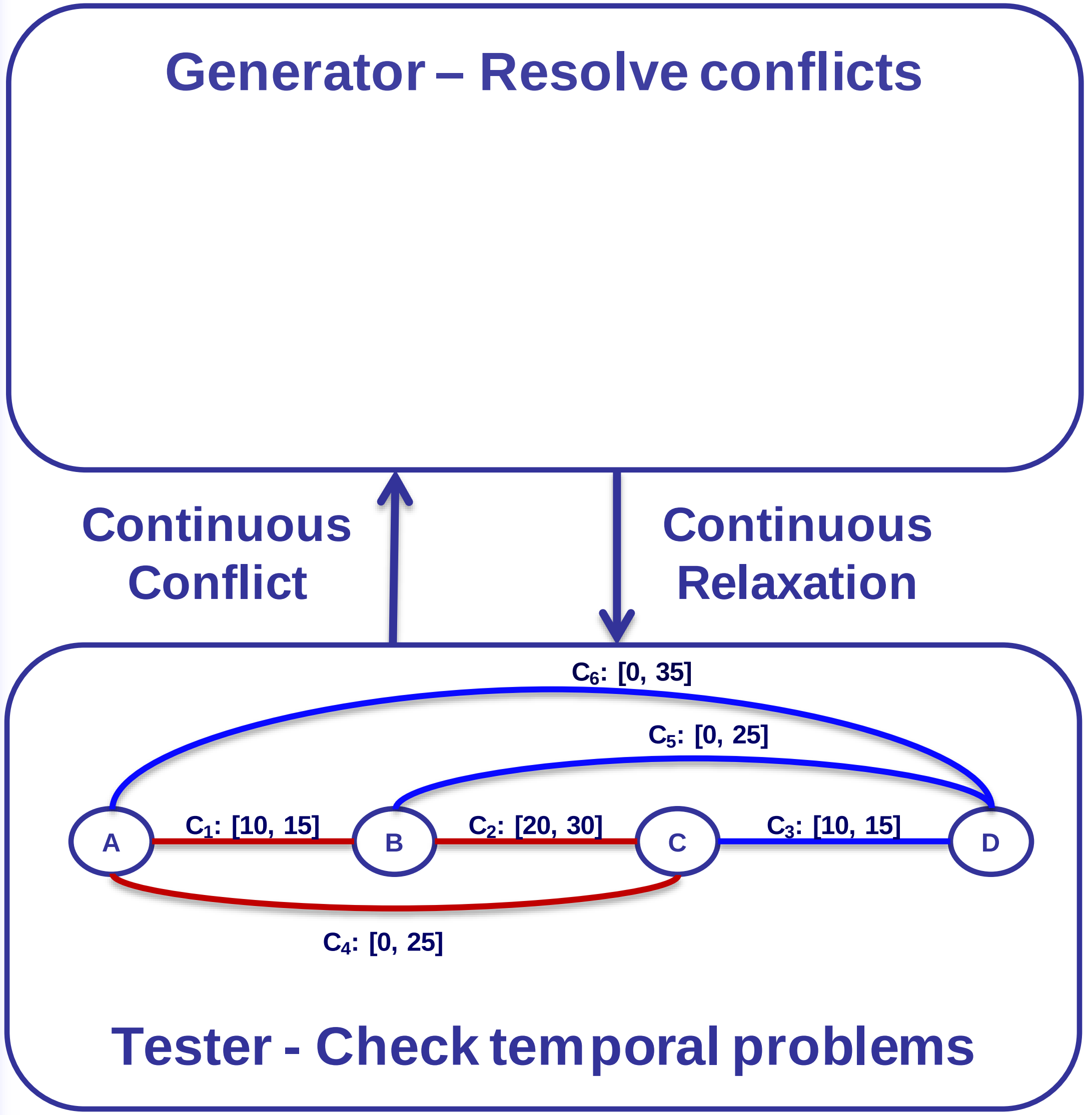

$$C_{4UB} - C_{2LB} - C_{1LB} = -5$$

- We reformulate the continuous conflicts into a set of linear inequalities.

$$\Delta C_{4UB} + \Delta C_{2LB} + \Delta C_{1LB} \geq 5$$

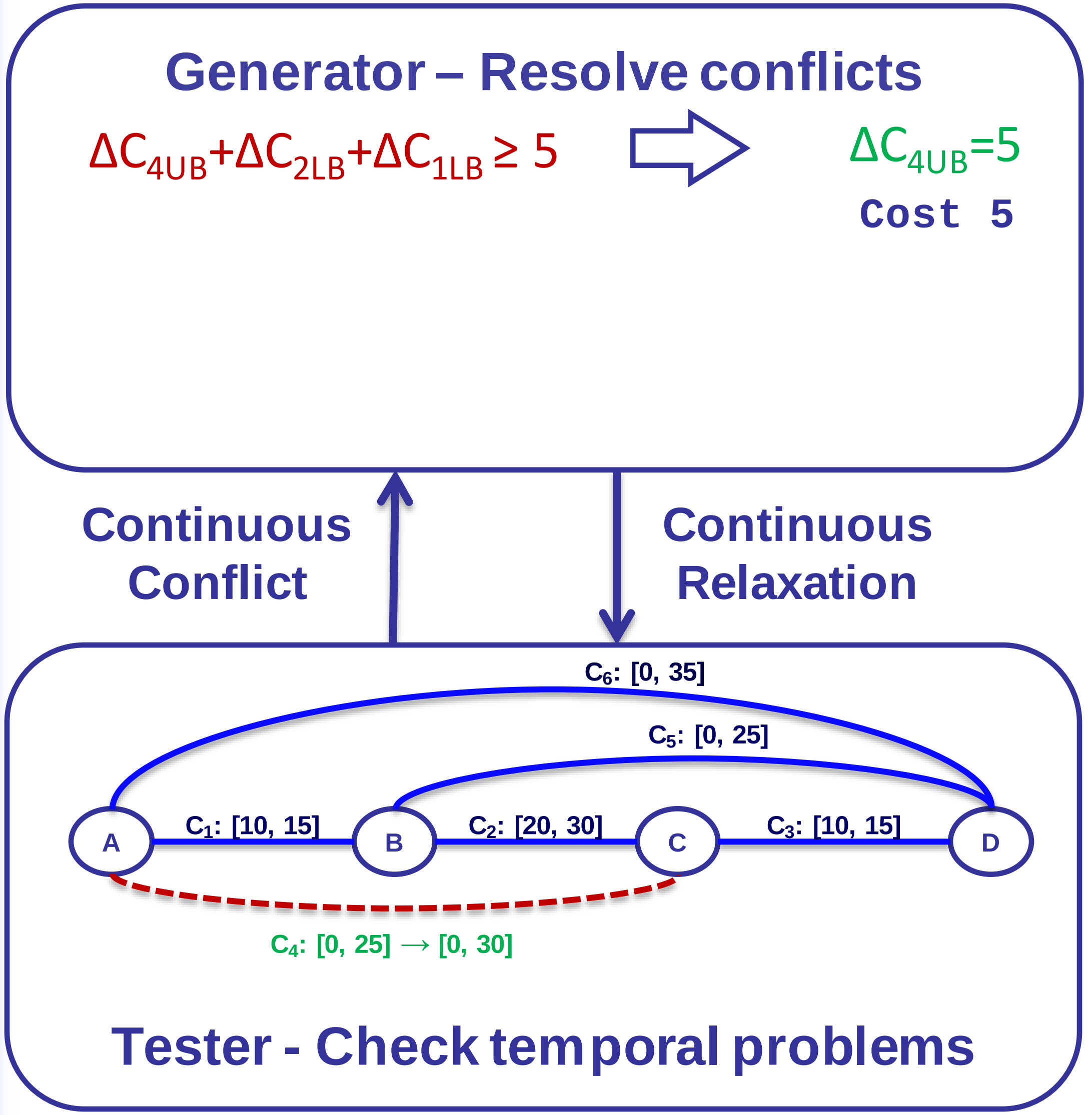
*"This is the minimal amount of deviation required to resolve this conflict."*

# Example – Test




- We start with no known conflicts and zero relaxation.
- The tester detects a conflict.

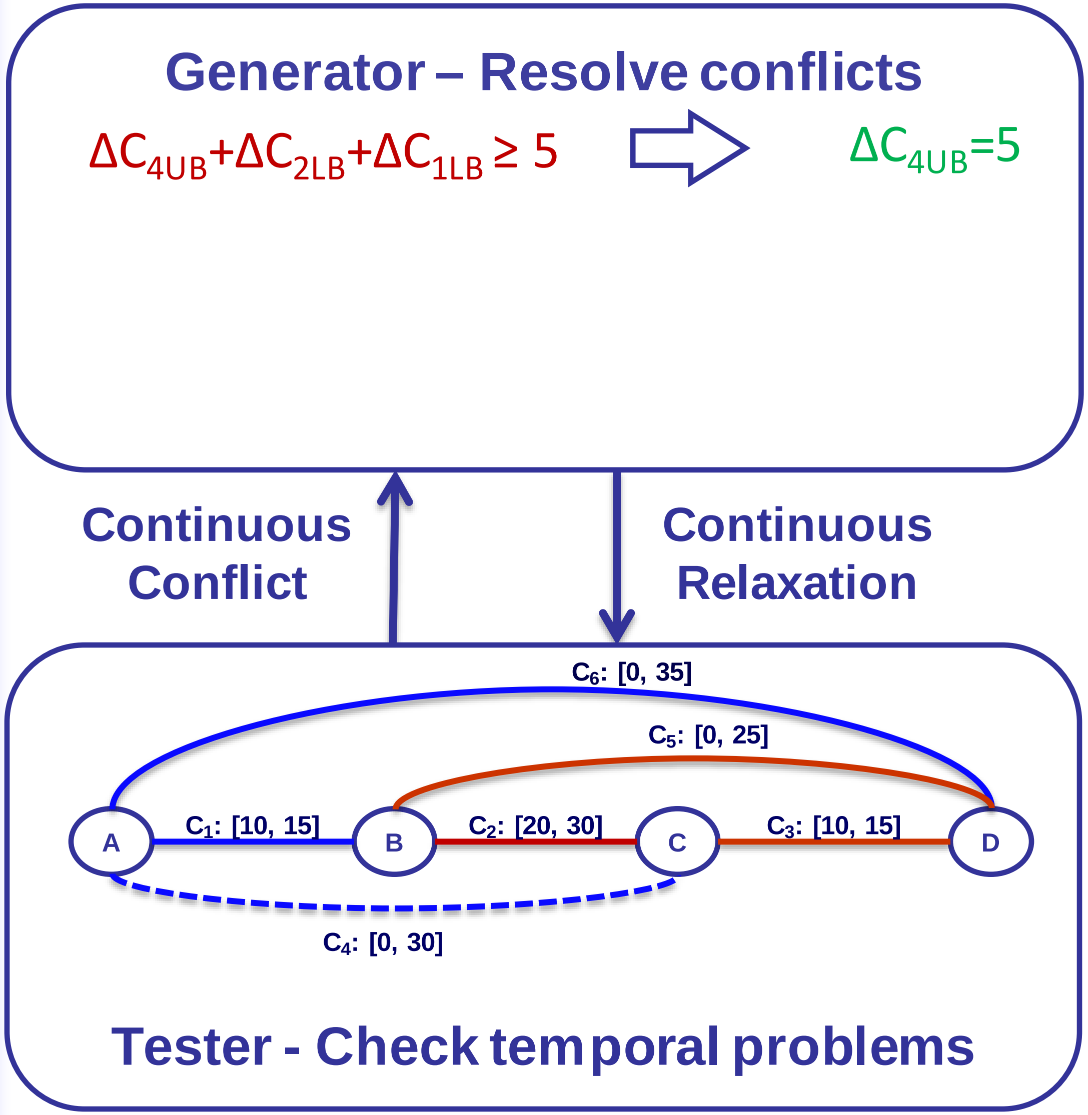
# Example – Generate Relaxation




- A continuous conflict was returned to the generator.
- It computes the best continuous relaxation that resolves the conflict.

# Example – Test






- Round 2: the generator passes the continuous relaxation to the tester.
- The tester detects a new conflict.

# Example – Generate Relaxation

**Generator – Resolve conflicts**

$\Delta C_{4UB}+\Delta C_{2LB}+\Delta C_{1LB} \geq 5$

$\Delta C_{5UB}+\Delta C_{2LB}+\Delta C_{3LB} \geq 5$

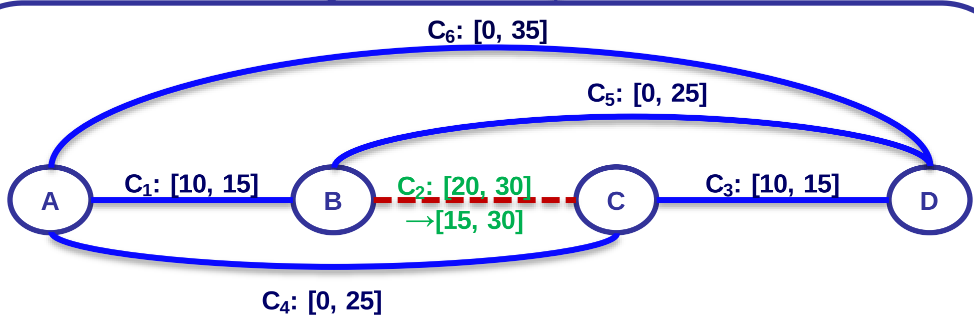
⇒

$\Delta C_{2UB}=5$

`Cost 7.5`

**Continuous Conflict**

**Continuous Relaxation**

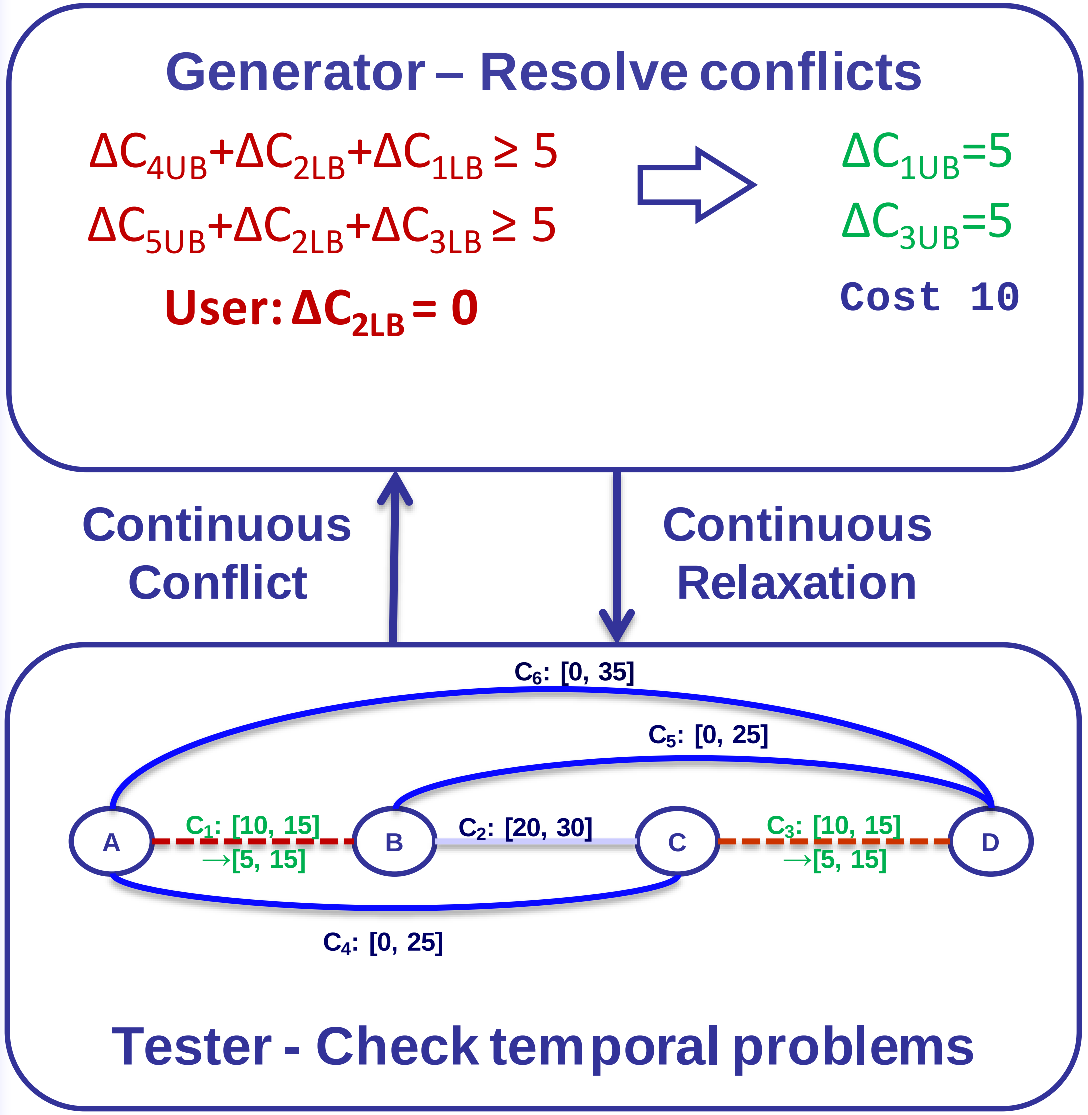



**Tester - Check temporal problems**

- A new continuous conflict was sent to the generator.
- It re-computes the best relaxation for both conflicts.
- This new relaxation passes the consistency check!

# Example – Incorporate User Response






- *User: "No, $C_2$ should be at least 20."*
- The user's input is recorded as a new continuous conflict.
- The generator re-computes the relaxation and asks the user again.

# Summary

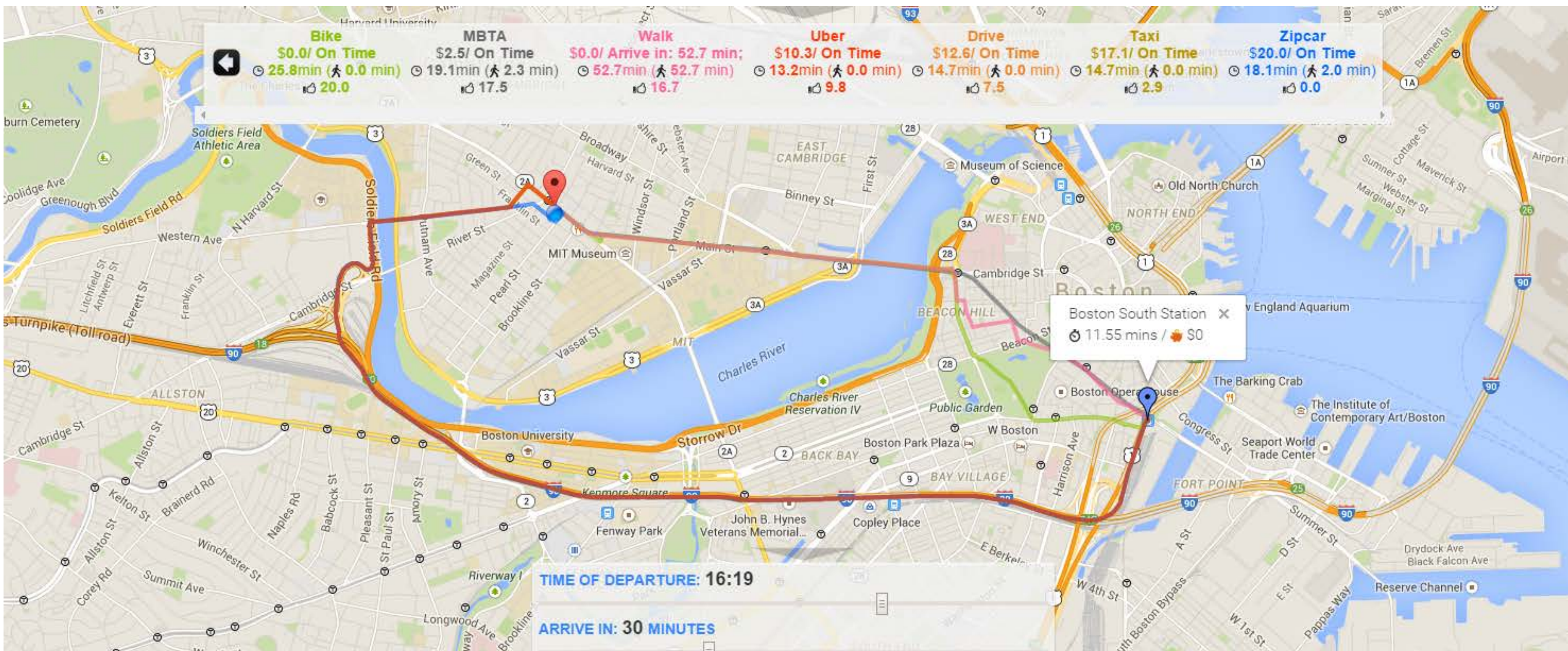


- Make trade-offs between constraints in the temporal problem (goals) to repair broken temporal problems.

- Detect and resolve over-subscription using model-based diagnosis approach.
  - The diagnosis methods can be applies to **diagnose** inconsistent set of temporal constraints.
  - The ‘failure modes’ are alternative goals (**goal relaxation**) that repair the broken plan.

# In the thought exercise

- Plan a trip to some points of interest in Boston, given some timing constraints.
- Please find the optimal trade-off between the mode of transportation and the time of arrival.



# Integrate Relaxation with Decision Making

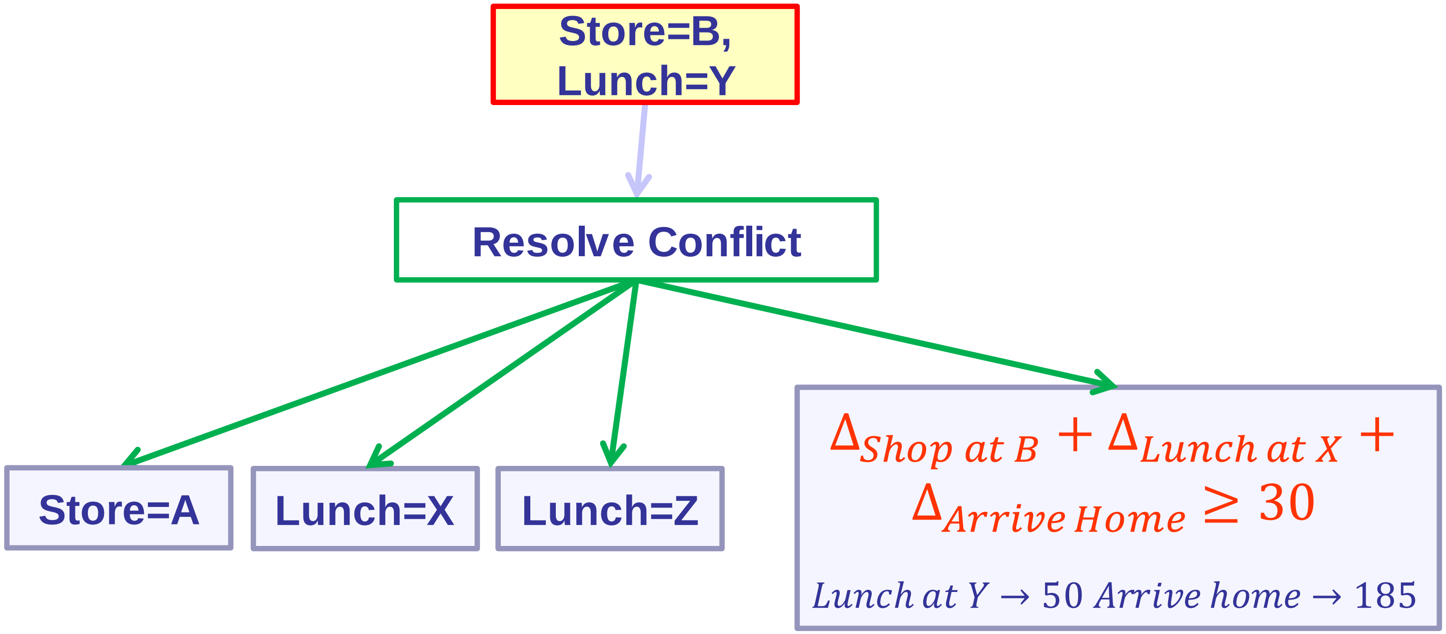
# Relaxation as part of Decision Making



- We have been working with temporal problems without choices so far.
- For problems with choices, we make decisions and compute relaxations simultaneously
- While incorporate user responses on the fly
  - ‘This cannot take that long’
  - ‘I do not want to go to restaurant A any more’

# Discrete and Continuous Relaxation

- We resolve a conflict using both continuous and discrete relaxations.




- The utility of the continuous relaxation is computed using the grounded solution of the lowest cost.

# Procedure: Enumerate Minimal Relaxation