# Temporal Problem Relaxation Through Conflict-Directed Diagnosis

Peng Yu

Model-based Embedded & Robotic Systems Lab
Research Supervisor: Professor Brian Williams

January 30th, 2012

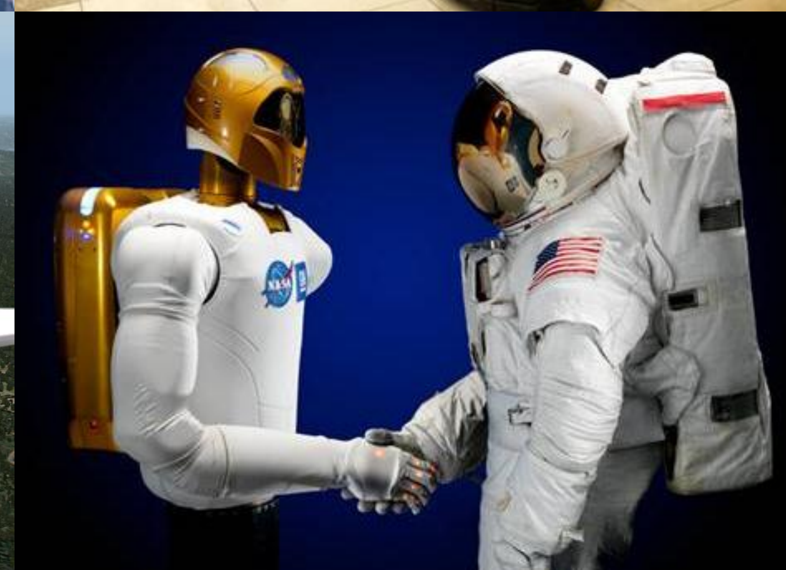
# Outline

- Motivation
- Previous Works
- Problem Definition and Approach
- Experimental Results
- Conclusion

# Motivation

- Robot-Human cooperation is very common nowadays.

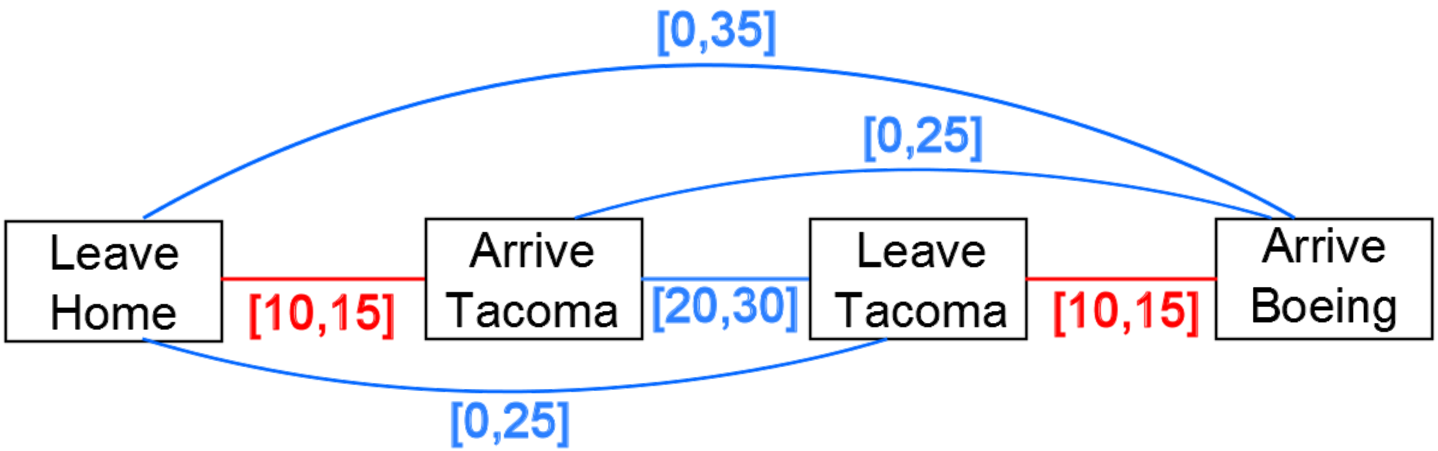


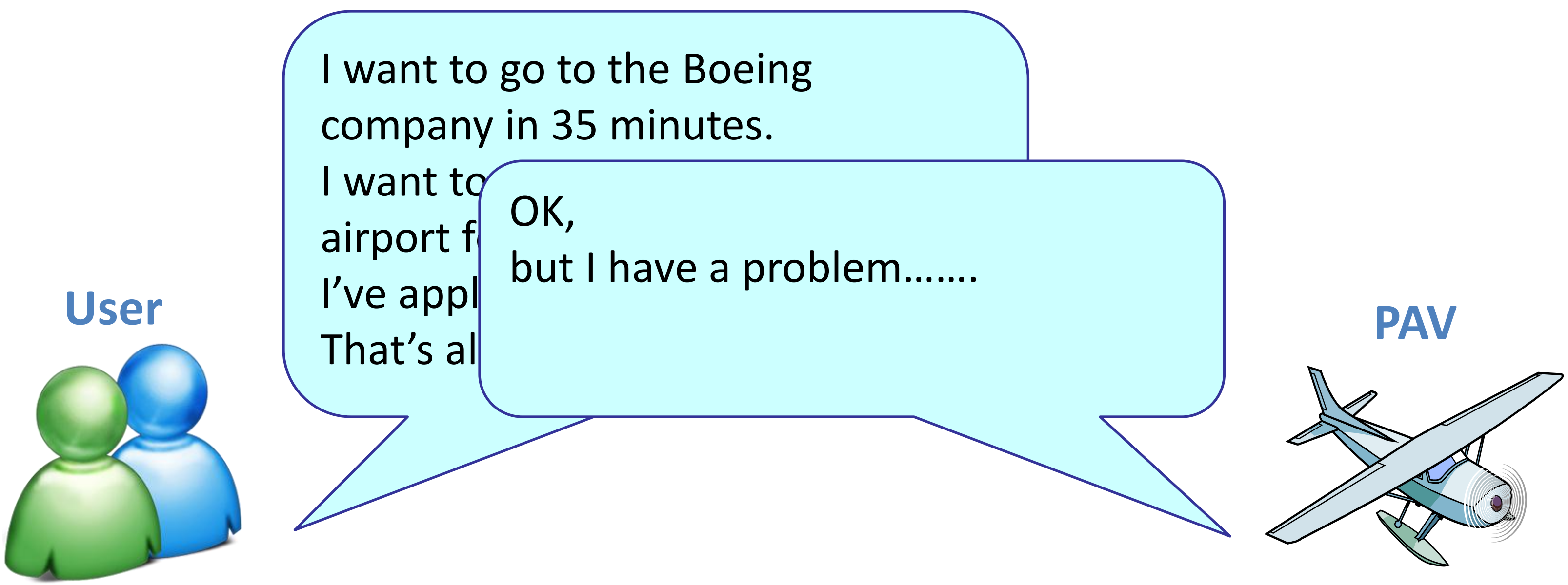



# Personal Transportation System

# Motivation

- Robot-Human cooperation is very common nowadays.



# Motivation

- Robot-Human cooperation is very common nowadays.
- However, inconsistencies are commonly observed during operations.
- Early systems only **signal failures** or **show many resolutions**.

I found 64 solutions:
1. Don’t stop at Tacoma.
2. Don’t go to work.

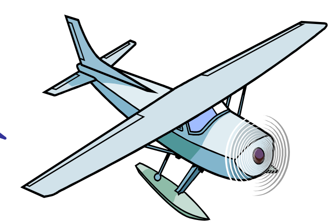
… …

64. ……

Which one do you choose?

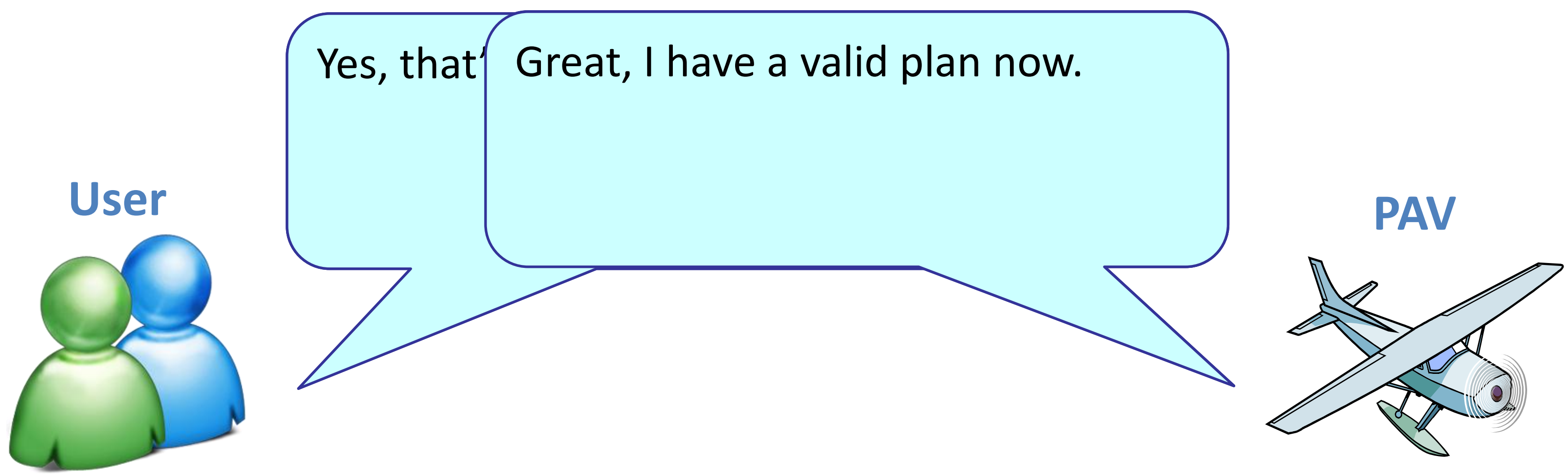
User

PAV



# Conflict-Directed Temporal Relaxation Algorithm

- Objective: Restore consistency of over-constrained plans.
- New features:
  - Provide a few **compact** and **preferred** solutions.
  - **Continuously** relax temporal constraints.
  - Accept both quantitative and **qualitative** preferences.

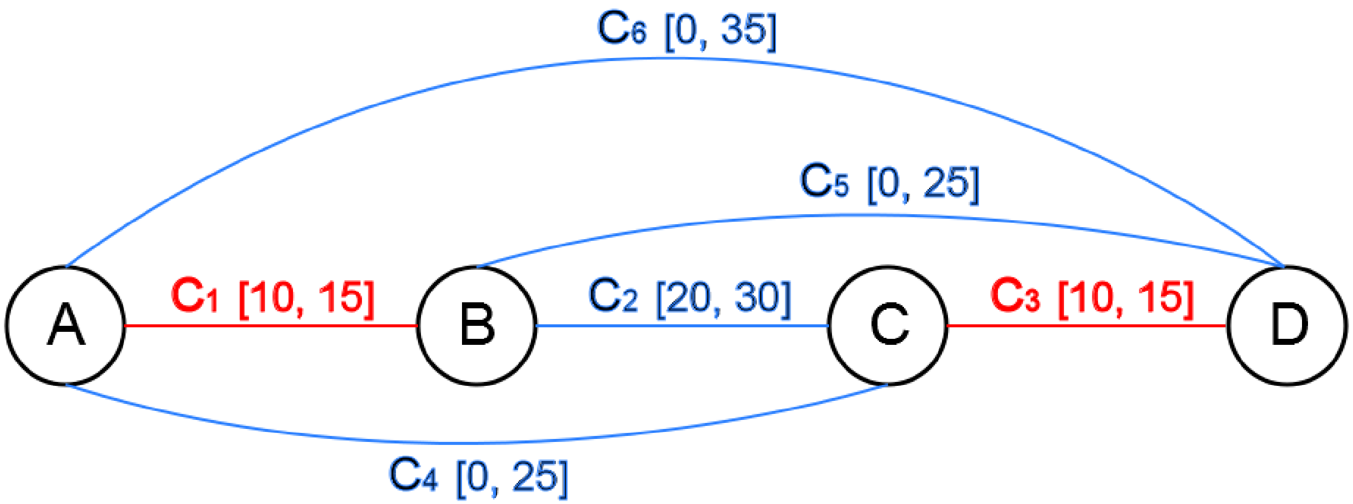


# Previous Work On Over-constrained Problems

| | Discrete Variables | Continuous Variables | User Preferences | Compact Results | Discrete Relaxations | Continuous Relaxations |
|---|---|---|---|---|---|---|
| de Kleer & Williams,1987 | ● | | | ● | ◌ | |
| Freuder & Wallace, 1992 | ● | | | | ● | |
| Williams & Ragno, 2003 | ● | | ● | | ● | |
| Bailey & Stucky, 2005 | ● | | | ● | ● | |
| Beaumont, et.al, 2001 | | ● | | | ● | |
| Moffitt & Pollack, 2005 | | ● | | ● | ● | |
| Peintner, et.al, 2005 | | ● | ● | | ● | |
| CTR | | ● | ● | ● | ● | ● |

# Problem Statement: Input

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

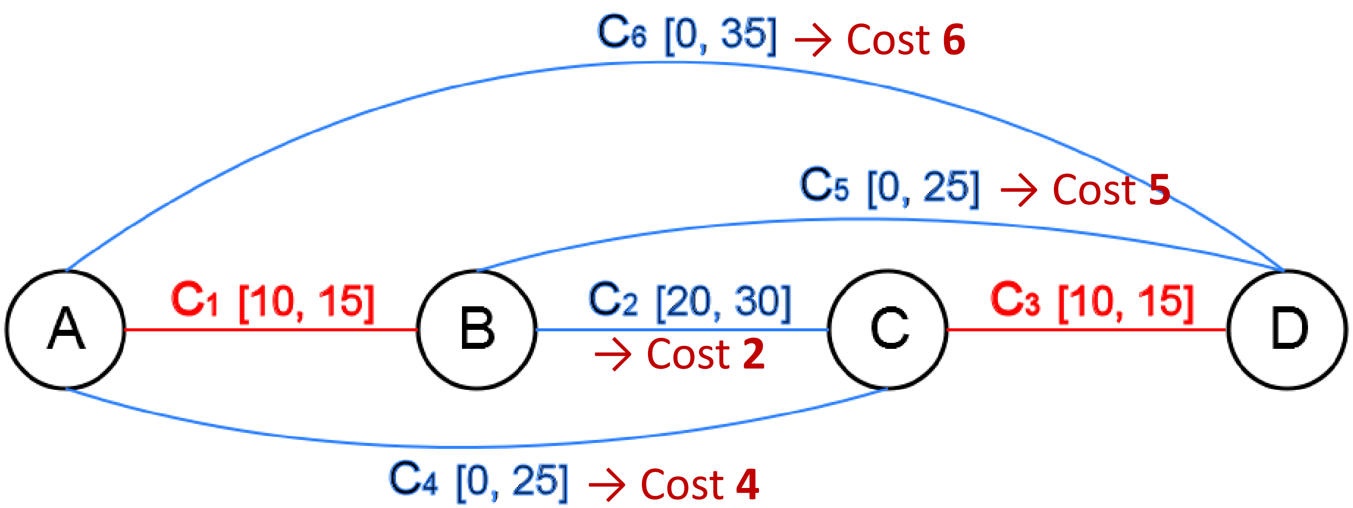


# Problem Statement: Preferences

- Quantitative Constraint Relaxation Cost:

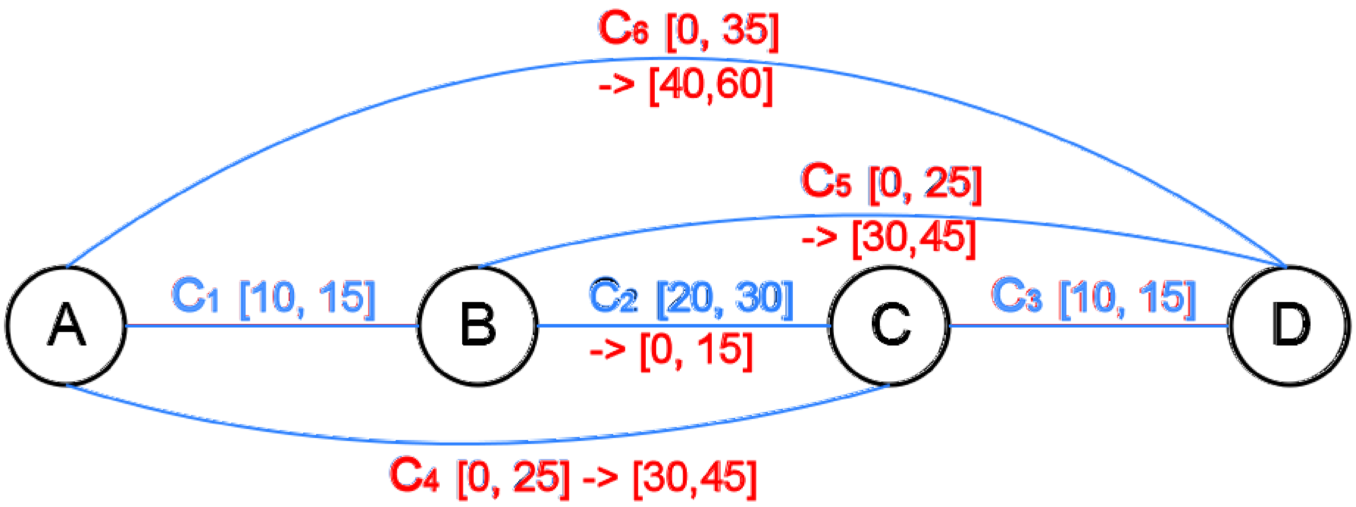
$$f: C_i \rightarrow \mathbb{R}$$

  - Activated when constraints are relaxed.
    - Relax C2: Cost 2.
    - Relax C4, C6: Cost 10.
  - Assume Mutual Preferential Independence.

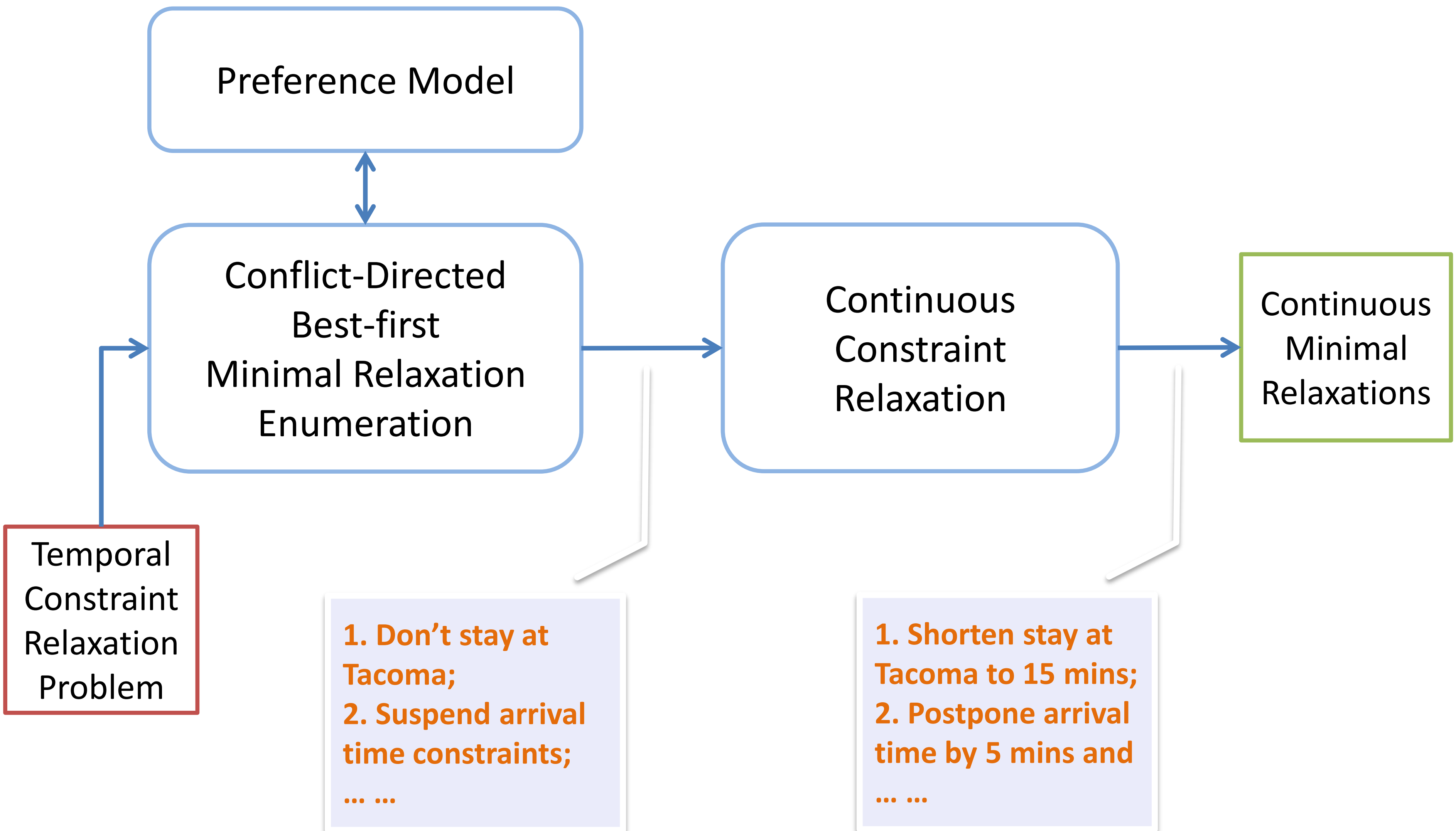


# Problem Statement: Output

- Preferred Continuous Minimal Relaxations.
  - Restore consistency by updating temporal bounds.
    - Relax C2 to [0,15].
    - Relax C4,C5,C6 to [30,45],[30,45],[40,60].
  - No proper subsets of which are valid relaxations.
  - In best-first order according to user preference.



# Conflict-Directed Temporal Relaxation



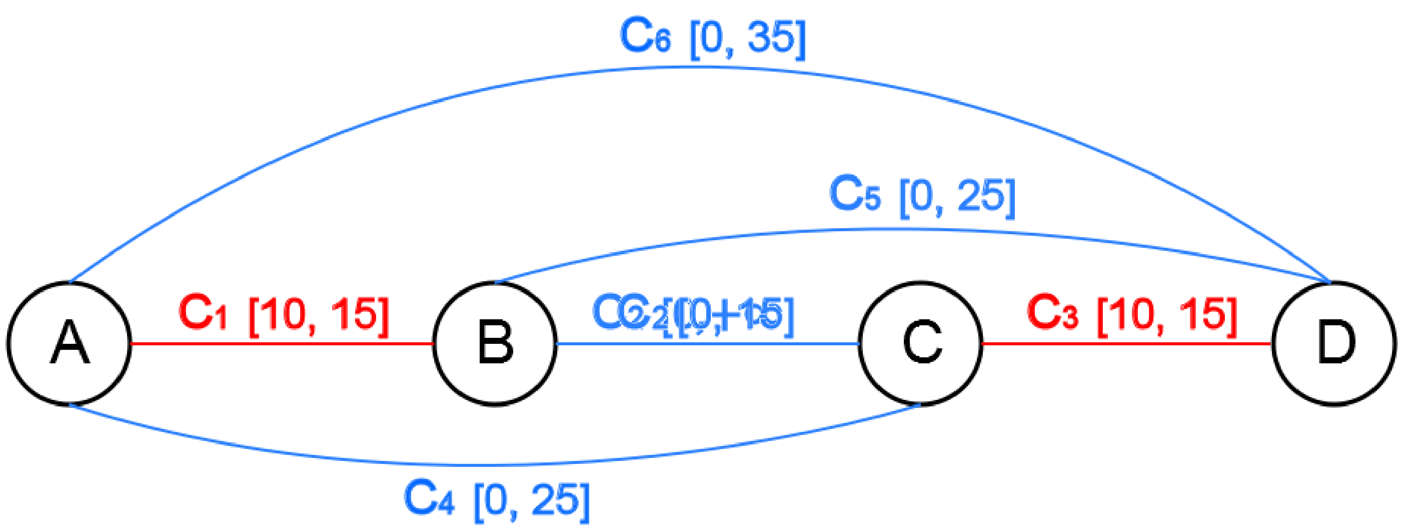
# Continuous Constraint Relaxation

- Previous approaches restore consistency by suspending constraints.
  - All-or-nothing approach.
  - Break the user’s plan.

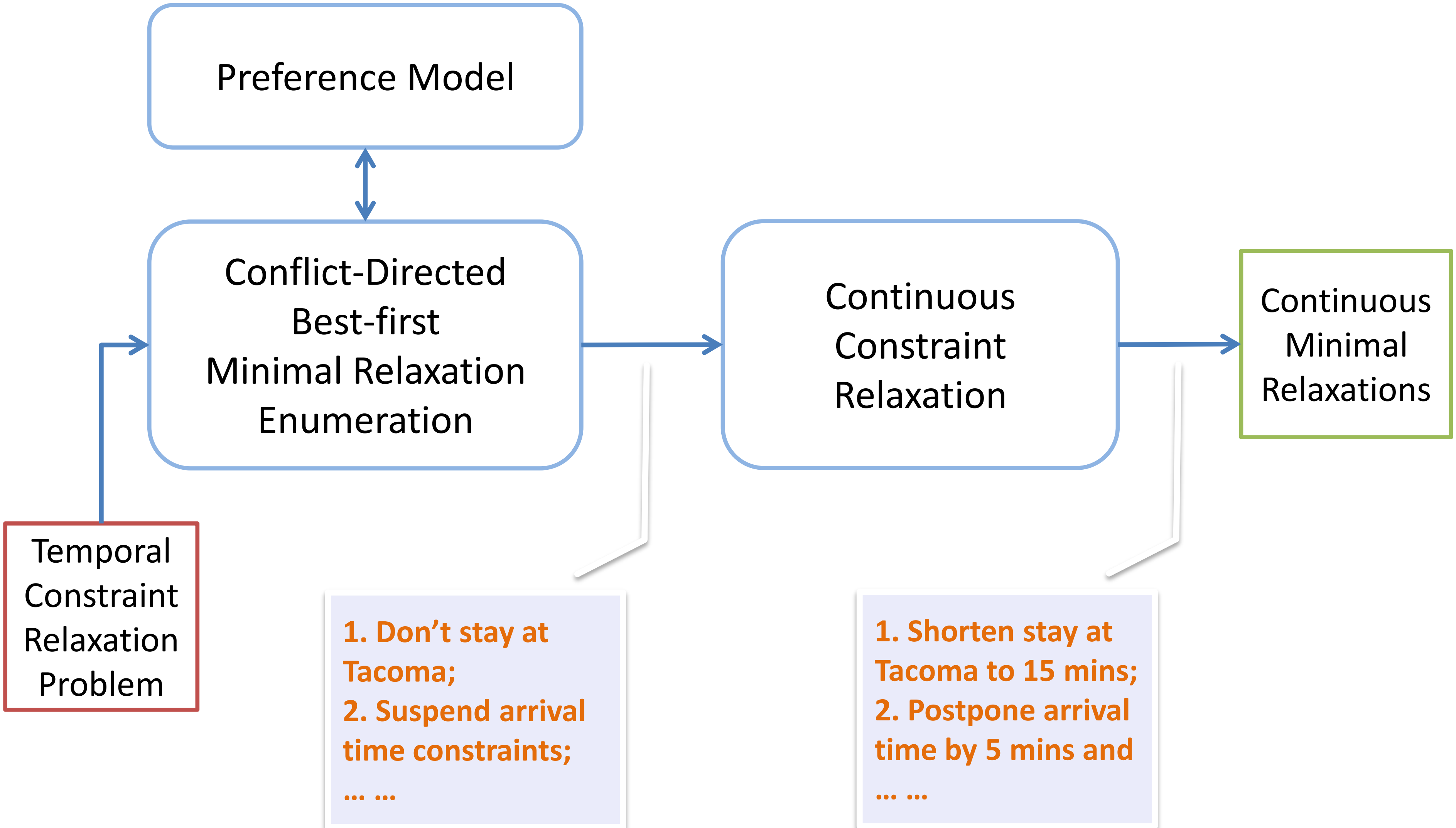
- Continuous Constraint Relaxation preserves all constraints.
  - Restore consistency by updating the temporal bounds .
  - Use Floyd-Warshall algorithm to compute the consistent temporal bounds of suspended constraints.

# Continuous Constraint Relaxation

- Step 1 generates a minimal relaxation: {Suspending C2}.
- CCR relaxes the temporal bound of C2 to restore consistency.
  - Reset C2 to [0, +∞].
  - Run Floyd-Warshall to update C2 with the tightest bound.
  - Extract the new temporal bound, [0, 15].

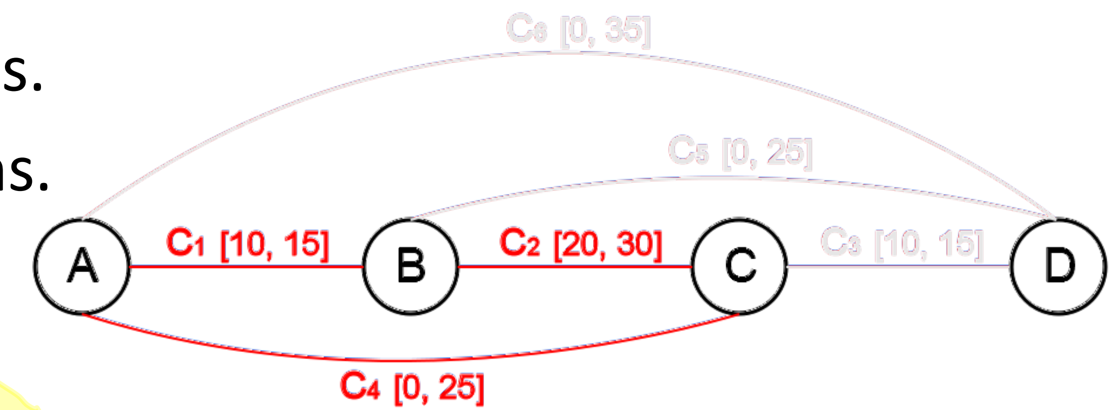


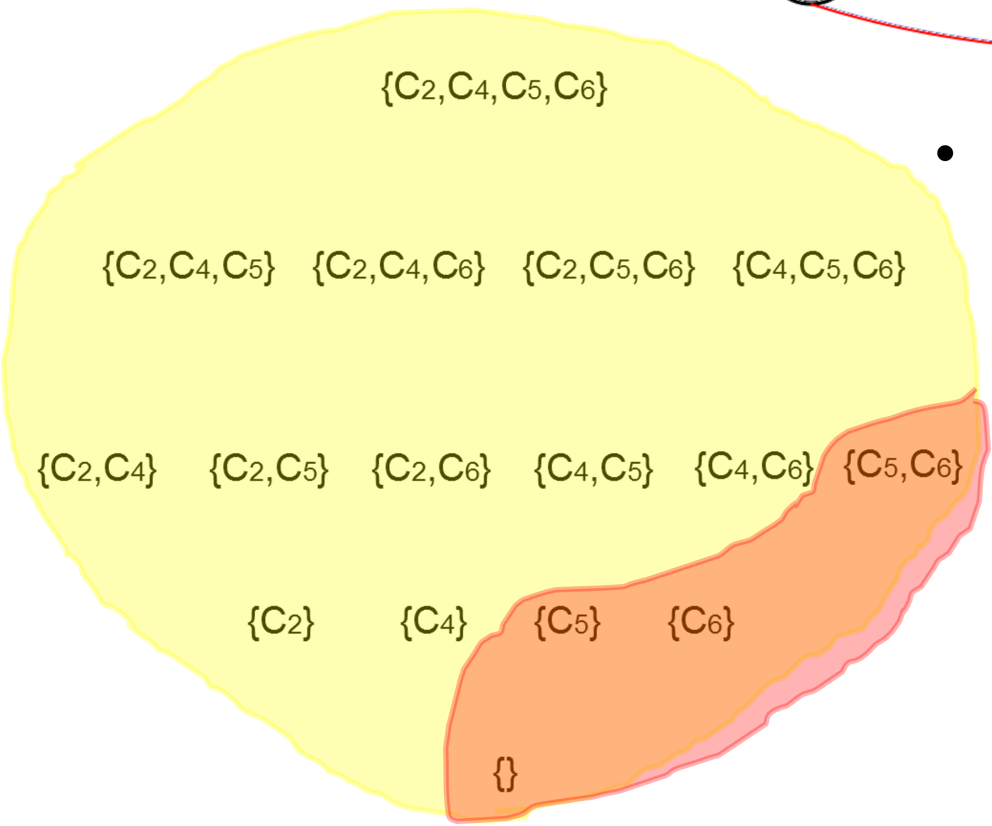
# Conflict-Directed Temporal Relaxation



# 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}.

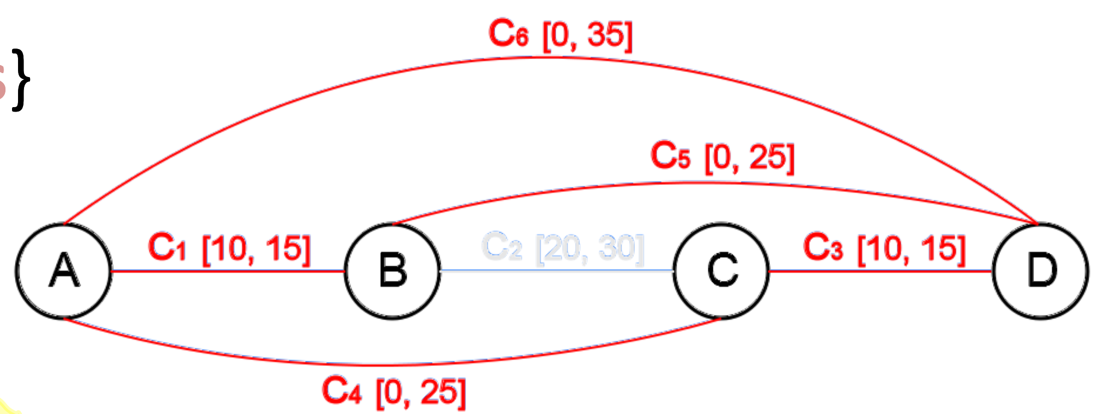
- Inconsistent!
- Extract **least cost** minimal conflict

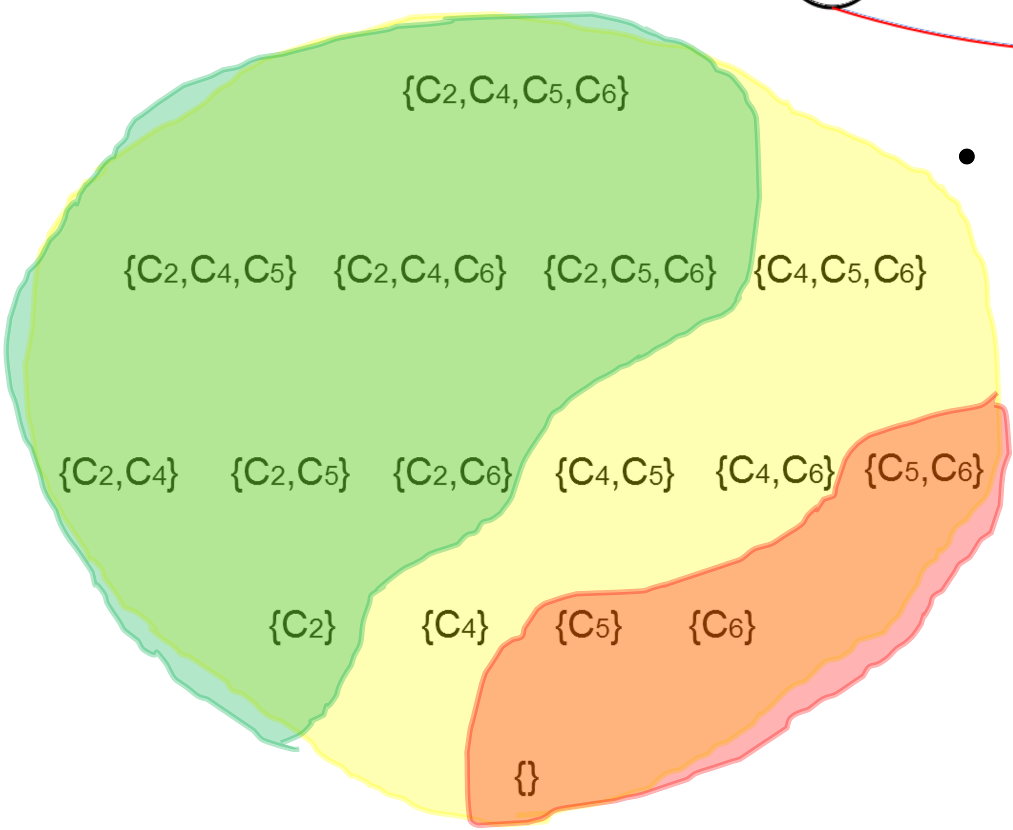
{C1=act,C2=act,C4=act}.

- Candidate **minimal** relaxations: {C2=sus},{C4=sus}.

# Enumerate Minimal Relaxations

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



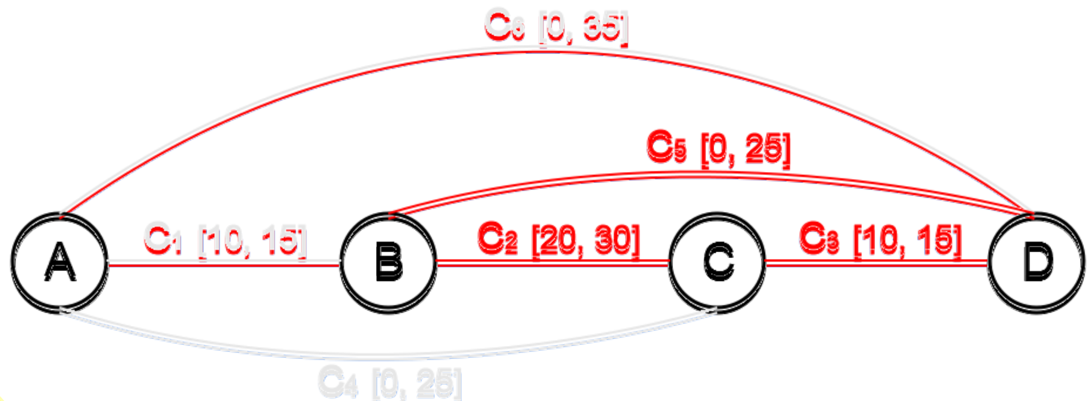



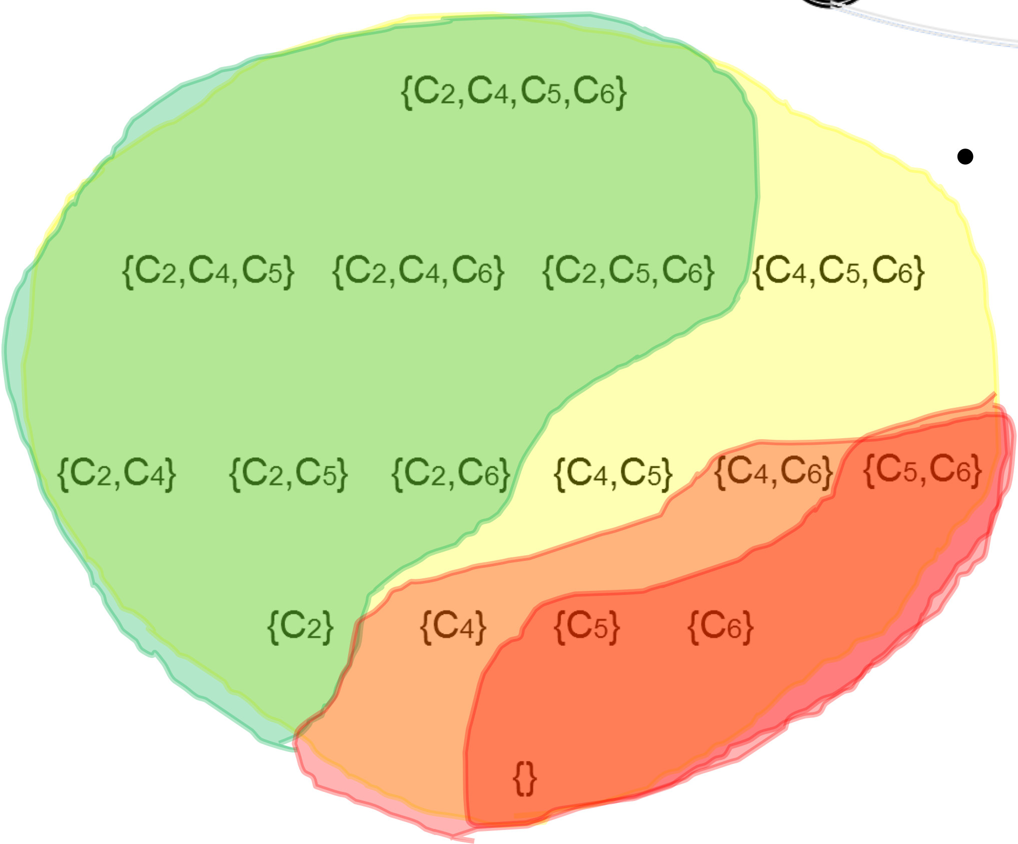

- Check {C2=sus}
  - Test {C2=sus,C4=act,C5=act,C6=act}.
  - Consistent!
  - All supersets of {C2=sus} are valid relaxations.

# Enumerate Minimal Relaxations

- Q = {C4=sus}

  {C4=sus,C5=sus}



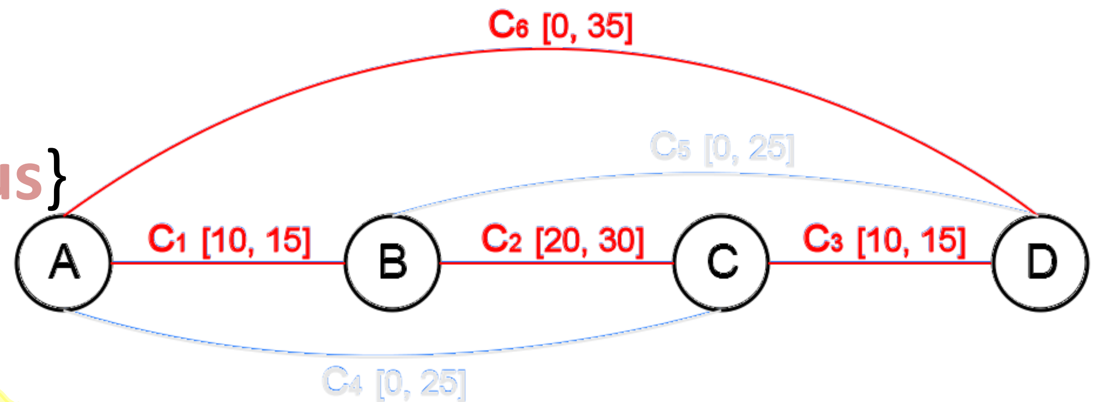



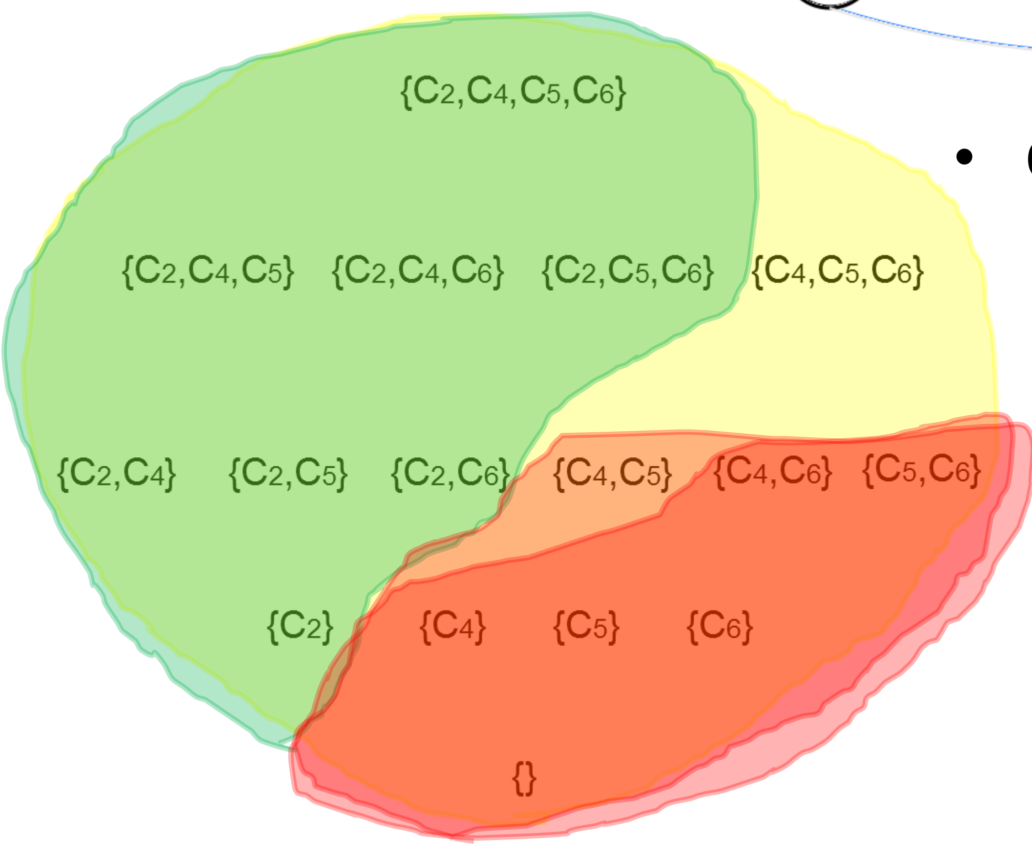

- Check {C4=sus}
  - Test
  
    {C2=act,C4=sus,C5=act,C6=act}.
  - Inconsistent!
  - Extract least cost minimal conflicts {C2=act,C3=act,C5=act}, {C1=act,C2=act,C4=act}.
  - Candidate minimal relaxations: {C4=sus, C5=sus}.

# Enumerate Minimal Relaxations

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

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

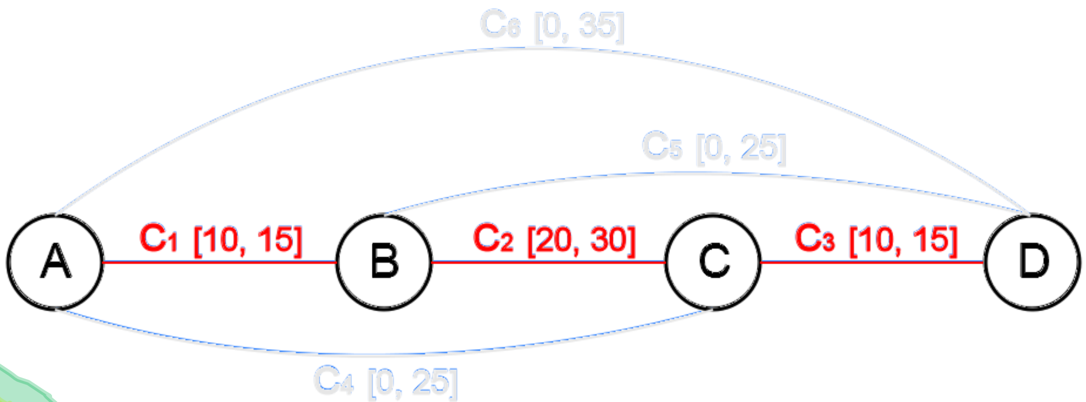


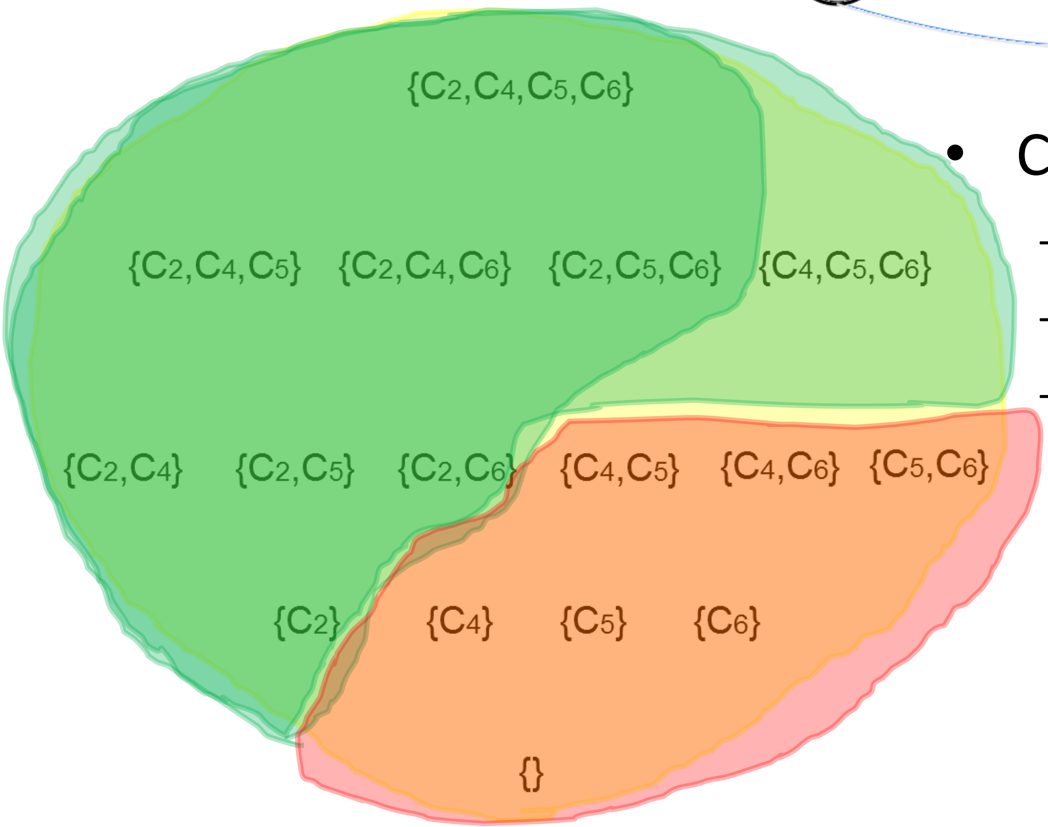




- Check {C4=sus,C5=sus}
  - Test {C2=act,C4=sus,C5=sus,C6=act}.
  - Inconsistent!
  - Extract conflicts {C1=act, C2=act, C3=act, C6=act}.
  - Candidate minimal relaxations: {C4=sus, C5=sus, C6=sus}.

# Enumerate Minimal Relaxations

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

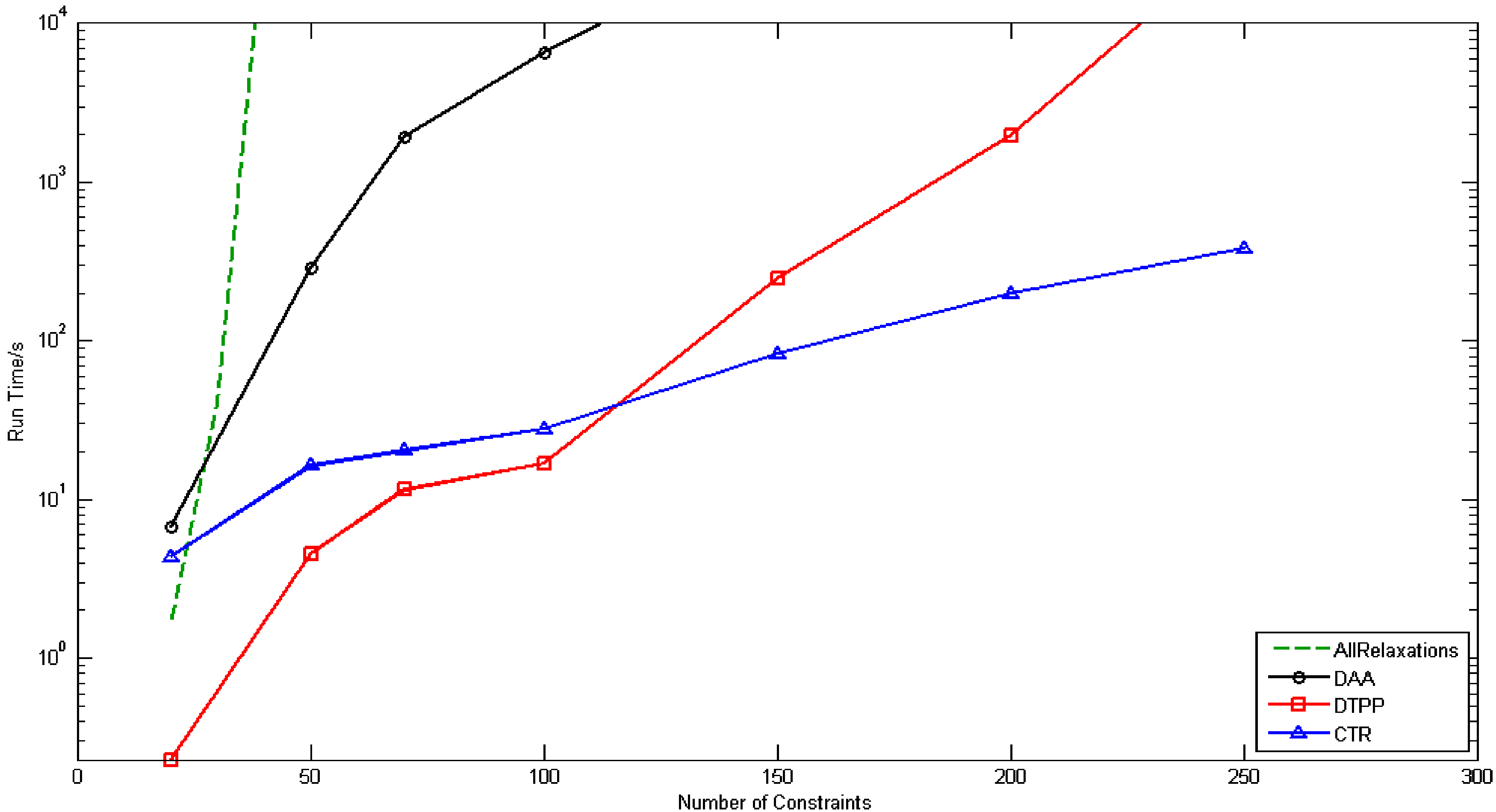



- Check {C4=sus,C5=sus,C6=sus}
  - Test {C2=act,C4=sus,C5=sus,C6=sus}.
  - Consistent!
  - No more candidate relaxations left, search complete!

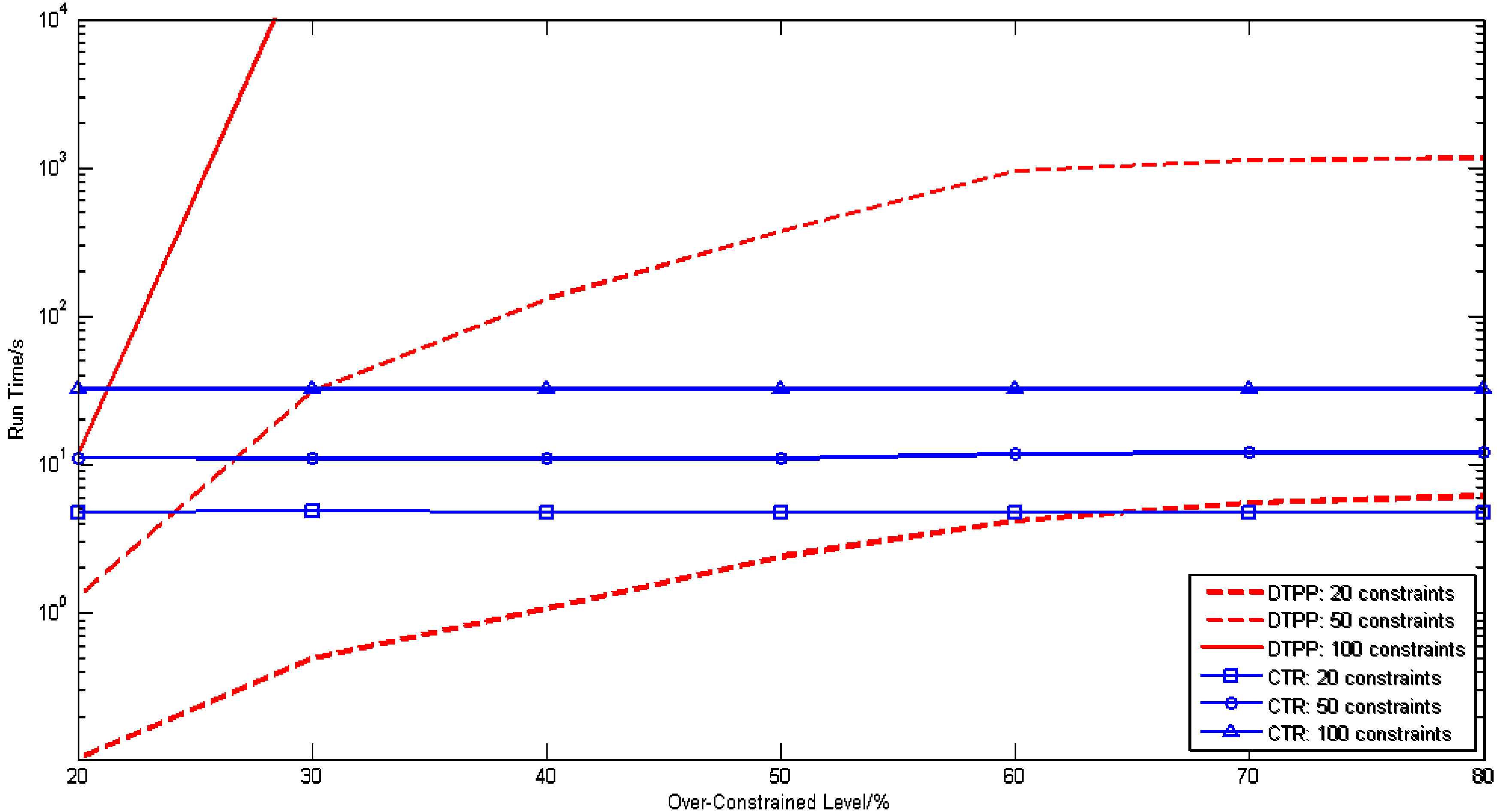
# Performance: Run-time

- All Relaxation, Dualize & Advance, DTPP, and CTR.



# Performance: Run-time

- Better performance on hard problems:
  - DTPP and CTR.



# Future Work

- A work together model:
  - For a larger set of applications, such as manufacturing, where human and robots are working together.
- Generative explanation:
  - Propose alternative plans with new constraints and actions in addition to suspending/relaxing existing constraints.
- A more natural user interface:
  - Shared knowledge model.
  - Efficient description of spatial and temporal commands.

# Achievements

- An algorithm to efficiently resolve over-constrained temporal problems.

- New features provided:
  - Best-first minimal relaxation enumeration.
  - Continuous constraint relaxation.
  - Relaxation enumeration with qualitative preferences.

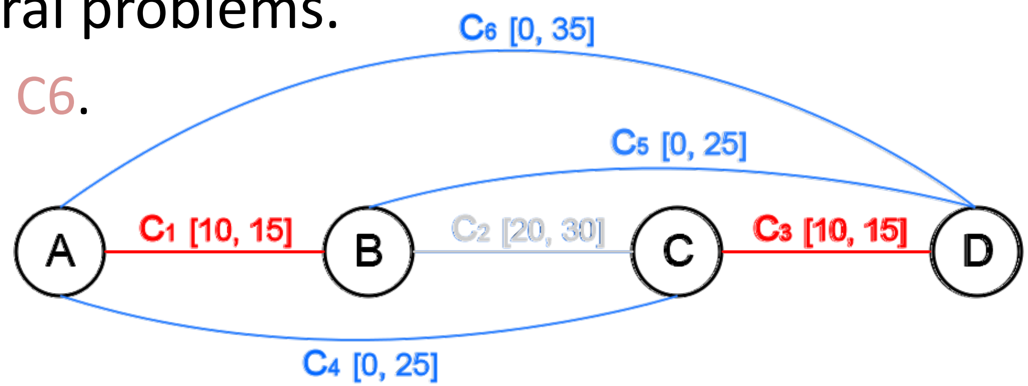
- Improvements:
  - Faster relaxations generation compare to state-of-the-art methods.

# Backup Slides

# 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:

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

  - Assigned by the user: C1 → 1, C2 → 2, C3 → 3 … …
  - {C2=**sus**, C5=**sus**}: Cost 7; {C2=**sus**, C4=**sus**}: Cost 6.

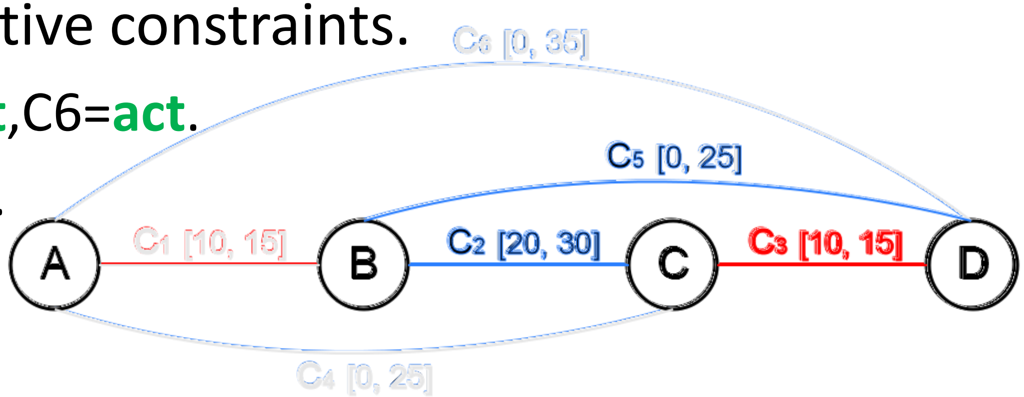
- Conflicts with lower cost will be extracted first.
  - {C1=**act**,C2=**act**,C4=**act**} vs. {C2=**act**,C3=**act**,C5=**act**}.

- Candidate relaxations with lower cost will be tested first.
  - {C2=**sus**} vs. {C4=**sus**, C5=**sus**, C6=**sus**}.

# Conflict-Directed

- What is a conflict?
  - A set of inconsistent active constraints.
    - C1=**act**,C2=**act**,C3=**act**,C6=**act**.
    - C1, C2, C3, C4, C5, C6.
    - C2, C3, C5.

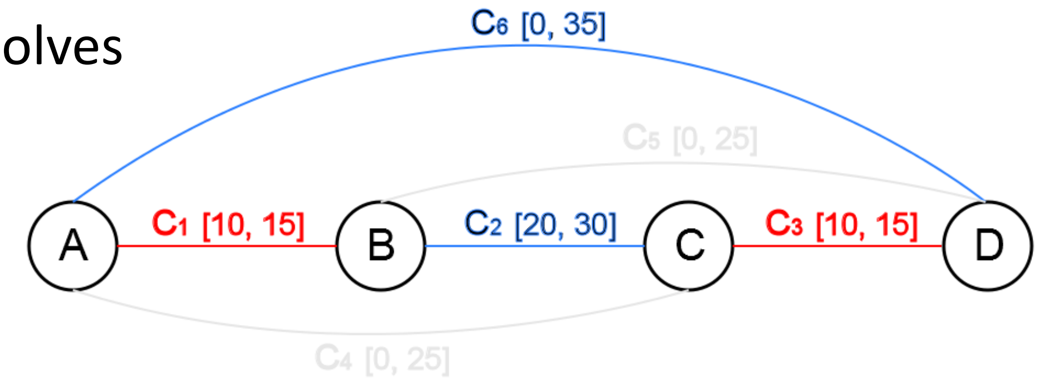
    … …




- 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**.

# Conflict-Directed, cont.

- Conflicts vs. 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).
  - Minimal relaxations are minimal covering sets of minimal conflicts.

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

# Generate Least Cost Conflicts

- Least cost conflicts:
  - Minimal conflicts with the least cost constraints.
- Procedure:
  - Prioritize the set of constraints in conflict.
  - Iterate through the set, starting with high cost constraints.
    - Remove the current constraint and check the consistency of the set.
    - If inconsistent, keep the constraint removed.
    - Otherwise, put it back to the set.

{C1,C2,C3,C4,C5,C6} → {C1,C2,C4}

# Generate Random Temporal Problems

- Given number of constraints, N:
  - Randomly generate:
    - Number of decisions: D;
    - Number of constraints per branch: Ci;

  such that

$$N = \sum_{i=1}^{D} C_i.$$

- Given the over-constrained level, O:
  - Manually add constraints to create conflicts.
  - Use the ratio, #conflicts/#constraints, to estimate O.

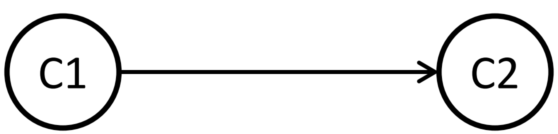
# Qualitative Preference Model: CP-nets

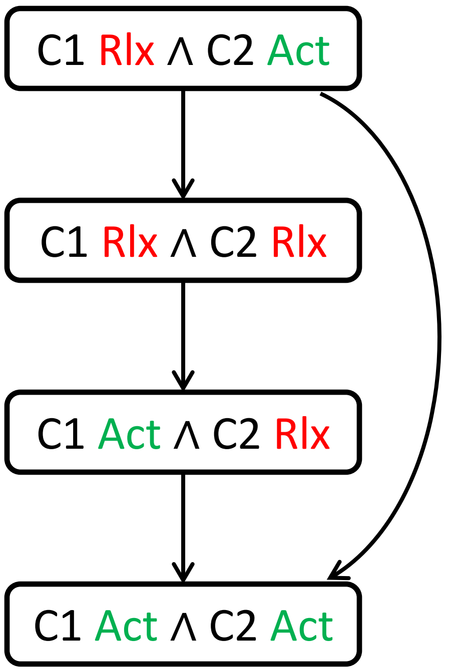
- Conditional Preference Networks:
  - Binary comparison over the decision outcomes with conditional independence assumption.
  - Assume relative importance between variables.
  - Compact and intuitive graphical representation.

- CP-Nets and relaxation problems:
  - Variables: relaxable constraints.
  - Outcomes: {Relax, Active}.

C1: Active $\succ$ Relax

C1 Active: Active $\succ$ Relax
C1 Relax: Relax $\succ$ Active

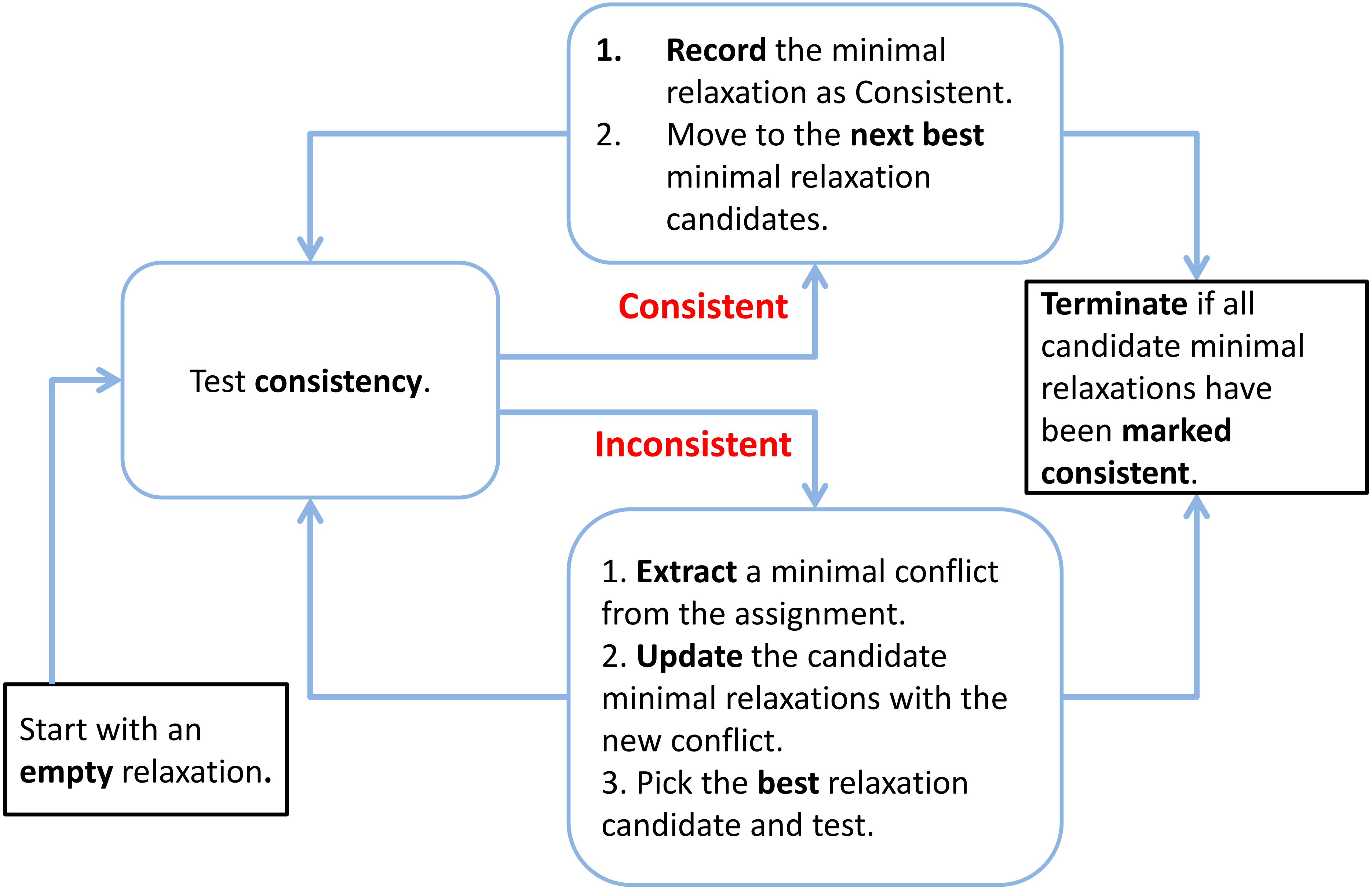


# Main Components

- Conflict-Directed Best-First Minimal Relaxation enumeration.
  - Leveraged from Conflict-Directed A* and Dualize & Advance.
- Continuous Constraint Relaxation.
  - Update the temporal bounds of suspended constraints to restore consistency.
- Conditional Preference Networks as Qualitative Preference Model.

# Procedure: Enumerate Minimal Relaxation



# Solving Relaxation Problem with MILP

- Relaxation cost: C1, C2,…,Cn.
- To Be Relaxed?: R1, R2,…,Rn ∈ {0,1}.
- Events: E1, E2,…,En.
- Constraints: a1 < Ei – Ej < b1; a2 < Em – En < b2;…

- Objective: Minimize $\sum_{i=1}^{n} C_i * R_i$;

  s.t. a1 – a1*R1 < Ei – Ej < b1 + M*R1;

  a2 – a2*R2 < Em – Em < b2 + M*R2;

  … …

- Generates the preferred full relaxations, but not minimal relaxations.

# All vs. Minimal relaxations: More compact results

| Number of Constraints | 20 | 50 | 70 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|---|---|
| All Relaxations | 3984 | 1.02e10 | 1.62e17 | 2.43e23 | 1.89e34 | 4.54e47 | 4.01e58 |
| Minimal Relaxations | 90 | 1680 | 9520 | 24200 | 2.97e6 | 3.84e8 | 5.14e10 |

# References


- de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. Artificial Intelligence 32:97–130.
- Freuder, E. C., and Wallace, R. J. 1992. Partial constraint satisfaction. Artificial Intelligence 58(1-3):21–70.
- Beaumont, M.; Sattar, A.; Maher, M.; and Thornton, J. 2001. Solving overconstrained temporal reasoning problems. In Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AI-2001), 37–49.
- Bailey, J., and Stuckey, P. 2005. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In Hermenegildo, M., and Cabeza, D., eds., Practical Aspects of Declarative Languages, volume 3350 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg. 174–186.
- Williams, B. C., and Ragno, R. J. 2003. Conflict-directed a* and its role in model-based embedded systems. Journal of Discrete Applied Mathematics.
- Moffitt, M. D., and Pollack, M. E. 2005. Partial constraint satisfaction of disjunctive temporal problems. In Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS-2005).
- Peintner, B.; Moffitt, M. D.; and Pollack, M. E. 2005. Solving over-constrained disjunctive temporal problems with preferences. In Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005).