Continuously Relaxing Over-constrained Conditional Temporal Problems

Peng Yu and Brian Williams
Massachusetts Institute of Technology
IJCAI-13 August 6, 2013
Robotic Personal Transportation System

- Video that highlights the interaction during trip planning.
• Desired capabilities in the video:

  – Minimize the perturbations to users’ requirements;
    “You have to shorten the dinner from 15 to 10 minutes”.

  – Explain both cause of failures and resolutions;
    “Because of the extended driving time”.

  – Prioritize different alternative solutions;
    “then how about arriving home 5 minutes late”.

  – Adapt to modified requirements during interactions.
    “if you want to spend at least 12 minutes on dining, then you
    will arrive at home 2 minutes late”.

Key features
Motivation:
- Provide **helpful** and **parsimonious** alternative solutions with **insights** into cause of failure.
- Like an experienced assistant.

Objectives:
- Enumerate only **continuous relaxations**;
- Record the **conflicts** detected in the requirements;
- Explore the candidate space in **best-first** order;
- Be **reactive** to newly added requirements.
Continuous Relaxations: Definition

• Previous approaches resolve over-constrained temporal problems by **suspending** constraints.
  – “Remove your trip duration requirement” or “eliminate your stay at the restaurant”.
  – Unnecessary in most scenarios.

• We only relax constraints continuously to reduce the perturbation:
  – Definition: a continuous relaxation, $CR$, is a set of relaxed constraints, $rc_1, rc_2, \ldots, rc_n$, such that the temporal problem is consistent.
Model: (Over-constrained) Controllable Conditional Temporal Problems.

- All variable are controllable.
- Allowing temporal constraints to be relaxed to restore consistency;
  
  • A subset of the temporal constraints, \( RE \subseteq E \), are relaxable.
  
  • A relaxable temporal constraint, \( re_i: [t_L, t_U] \), can be relaxed to \( re_i': [t_L - \Delta_L, t_U + \Delta_U] \) if necessary.
From conflicts to relaxations

- A conflict composes of an inconsistent set of temporal constraints and their required assignments.
- We learn conflicts from the negative cycles detected by temporal consistency algorithms.

Conflict:
- \(GS = B, LU = Y\);
- Home \(\rightarrow B \geq 35\); Shop at \(B \geq 35\);
- Drive \(B \rightarrow Y \geq 25\); Lunch at \(Y \geq 75\);
- \(Y \rightarrow Home \geq 40\); Arrive Home \(\leq 180\).
Conflict Resolution

• Compute both *discrete* and *continuous* constituent resolutions to a conflict.

<table>
<thead>
<tr>
<th>Discrete Resolutions</th>
<th>Continuous Resolutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS = A; LU = X; LU = Z;</td>
<td></td>
</tr>
<tr>
<td>Shop at B ≥ 5; Lunch at Y ≥ 45; Arrive Home ≤ 210;</td>
<td></td>
</tr>
<tr>
<td>... ...</td>
<td></td>
</tr>
</tbody>
</table>

• Note that only relaxable temporal constraints can be relaxed to resolve conflicts.
Preferences over relaxations

- Defining preference functions over variable assignments and constraint relaxations.
  - Each variable assignment is mapped to a positive reward value by function $f_p$.
  - Each constraint relaxation is mapped to a positive cost value by function $f_e$.

<table>
<thead>
<tr>
<th>Grocery</th>
<th>A</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>100</td>
</tr>
<tr>
<td>Lunch</td>
<td>X</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>30</td>
</tr>
</tbody>
</table>

Assignment reward:
\[
\alpha = \{\text{Grocery} = B, \text{Lunch} = Y\}
\]
\[
f_p(\alpha) = 100 + 80 = 180
\]

Relaxation cost:
\[
r = \text{ReservationTime}[0,180] \rightarrow [0,200]
\]
\[
f_p(r) = f_p(200 - 180) = 40
\]
Compute Preferred Continuous Relaxations

- Using the cost function over relaxable constraints, we compute only the most preferred continuous relaxation.

- This is framed as a linear optimization problem.

**Objective Fn:**
\[
\min \sum_{e_i \in \text{Conflict}} (f_e (UB'_e - UB_e) + f_e (LB_e - LB'_e))
\]

**Constraints:**
\[
\sum_{e_i \in \text{Conflict}} [(UB'_e - UB_e) + (LB_e - LB'_e)] \geq -1 \times V_{n_{cycle}}
\]

- For example:

  Home → B ≥ 35; Shop at B ≥ 35; Drive B → Y ≥ 25; Lunch at Y ≥ 75; Y → Home ≥ 40; Arrive Home ≤ 180.

\[
\min (f(\Delta_{\text{Shop at } B}) + f(\Delta_{\text{Lunch at } Y}) + f(\Delta_{\text{Arrive Home}}))
\]

\[
\text{s.t. } \Delta_{\text{Shop at } B} + \Delta_{\text{Lunch at } Y} + \Delta_{\text{Arrive Home}} \geq 30
\]

- We restrict the cost functions to be convex so that the relaxation process is always tractable.
• BCDR enumerates relaxations in best-first order:
  – It searches over subsets of constraints by making different variable assignments.
  – It resolves a conflict by relaxing a constraint, partially and completely.
Expand on Unresolved Conflicts

- If a node has an unresolved conflict, we expand it using both continuous and discrete constituent relaxations.

- The utility of the continuous relaxation is computed using the grounded solution of the lowest cost.
Continuous Relaxations for Multiple Conflicts

• For two or more continuous relaxations on the same branch, the utility is determined by the ground solution that respects both inequalities.

\[
\min \left( f(\Delta_{Shop at B}) + f(\Delta_{Lunch at Y}) + f(\Delta_{Arrive Home}) + f(\Delta_{Drive to B}) + f(\Delta_{Drive B to X}) + f(\Delta_{Travel Time}) \right)
\]

s.t.
\[
\Delta_{Shop at B} + \Delta_{Lunch at X} + \Delta_{Arrive Home} \geq 30
\]

and
\[
\Delta_{Drive to B} + \Delta_{Drive B to X} + \Delta_{Travel} \geq 30
\]
Incorporating User Responses

• BCDR is reactive to newly added user requirements.
• We encode these requirements as new conflicts, and follow them as constraints in the continuous relaxation.

No, I do not want to extend my reservation time.

No, I want to spend at least 25 minutes on shopping.

\[
\text{Arrive Home} \leq 180; \\
\text{Shop at B} \geq 25;
\]

\[
\begin{align*}
\min(f(\Delta_{\text{Shop at B}}) + f(\Delta_{\text{Lunch at Y}}) + f(\Delta_{\text{Arrive Home}})) \\
\text{s.t. } & \Delta_{\text{Shop at B}} + \Delta_{\text{Lunch at X}} + \Delta_{\text{Arrive Home}} \geq 30; \\
& \Delta_{\text{Arrive Home}} \leq 0; \\
& \Delta_{\text{Shop at B}} \leq 10.
\end{align*}
\]
Applications

• Personal Transportation System.

• Mission advisory system for autonomous underwater vehicles.

• Scheduling assistant for factory floor operations.

• Trip advisor for car-sharing network users.
Empirical Validation - Setup

• We simulated a car-sharing network in Boston using randomly generated car locations and destinations.

• Test cases are characterized by:
  – Number of reservations per car.
  – Number of cars in the network.
  – Number of activities per reservation.
  – Number of alternative options per activity.

• Time change may affect neighboring reservations.
We compare BCDR-GC, BCDR-DC and DFS-GC in finding the first relaxation to a CCTP.
Empirical Validation - Limitations

- The advantage of generalized conflict resolution is limited when the relaxation cost is significant lower than rewards.
  - We reduced the range of cost values in the previous test by 100 times.
• In this paper, we presented the Best-first Conflict-Directed Relaxation approach to:
  – Resolve over-constrained temporal problems using continuous relaxations.
  – Minimally relax constraints for minimal perturbation.
  – Efficiently enumerate relaxations using generalized conflict learning and resolutions.
  – Be reactive to newly added user requirements.

• Executable, test cases and slides are available at:
  http://people.csail.mit.edu/yupeng/software.html

Contributions
Backup slides
Objectives

• To resolve an over-constrained CTPs:
  – Preferably: using the most preferred relaxations;
  – Minimally: minimizing the perturbations to the problem;
  – Efficiently: speeding up the search for feasible relaxations.

• BCDR extends the Conflict-Directed A\(^*\) algorithm to continuous variables, constraints and preference functions for best-first relaxation enumeration.

• Implemented as a reservation advisory system for car-sharing networks, BCDR demonstrates significant improvements in performance and solution quality.
Motivating Example

- Planning a weekend trip using a car-sharing network.
Trade-off Between Trip Parameters

- Which store/restaurant to visit?
- How much time should the shopping/dining time be?
- How long should I reserve the car?
- How much am I willing to spend for the trip?

### Driving Time in minutes

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Time Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td>[30,40]</td>
</tr>
<tr>
<td>B</td>
<td>X</td>
<td>[35,40]</td>
</tr>
<tr>
<td>A</td>
<td>Y</td>
<td>[25,30]</td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
<td>[25,40]</td>
</tr>
<tr>
<td>A</td>
<td>Z</td>
<td>[20,25]</td>
</tr>
<tr>
<td>B</td>
<td>Z</td>
<td>[30,35]</td>
</tr>
<tr>
<td>Home</td>
<td>A</td>
<td>[35,50]</td>
</tr>
<tr>
<td>Home</td>
<td>B</td>
<td>[35,40]</td>
</tr>
<tr>
<td>X</td>
<td>Home</td>
<td>[45,50]</td>
</tr>
<tr>
<td>Y</td>
<td>Home</td>
<td>[40,50]</td>
</tr>
<tr>
<td>Z</td>
<td>Home</td>
<td>[50,60]</td>
</tr>
</tbody>
</table>

### Shopping/Dining Time in minutes

<table>
<thead>
<tr>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>40</td>
</tr>
<tr>
<td>B</td>
<td>35</td>
</tr>
<tr>
<td>X</td>
<td>50</td>
</tr>
<tr>
<td>Y</td>
<td>75</td>
</tr>
<tr>
<td>Z</td>
<td>100</td>
</tr>
</tbody>
</table>

![Graph showing the trade-off between driving time and shopping/dining time](image-url)
Intelligent Reservation System

- Automatically find and negotiate a solution with the user that satisfies the user’s requirements to the maximal extent.

I want to shop for grocery and have lunch afterwards. Complete the trip in 3 hours, if possible.

I see. You may shop at B and have lunch in restaurant Y. The lunch time will be 50 minutes and you only need to extend your reservation by 5 minutes. Is it ok?

No, I do not want to extent my reservation time.

No, I want to spend at least 25 minutes on shopping.

OK. How about eating at X for 48 minutes and reduce shopping time to 17 minutes?

OK, that’s fine.

OK. How about eating at Y for 55 minutes? You can then shop at B for 25 minutes.

OK, that’s fine.
Approach

• **Model:** Controllable Conditional Temporal Problem.
  
  - Captures the users’ requirements, preferences, alternatives and environment constraints.

• **Algorithm:** Best-first Conflict-Directed Relaxation.
  
  - Enumerate solutions/relaxations to a CCTP in best-first order.
    
    • Assignments: where to go and what to do.
    
    • Relaxations to temporal constraints: alternative goals/requirements to ensure consistency.
Minimal Continuous Relaxations

• Minimal **discrete** relaxations:
  – A valid relaxation, whose proper subsets are not valid relaxations.
  – Generated by computing the minimal covering sets of minimal conflicts.

• Minimal continuous relaxations:
  – A set of continuous relaxations whose
  – Generated by solving a linear optimization problem against the conflicts.
Solution

• A solution to a CCTP $T$ is a pair $\langle A, R \rangle$:
  – $A$: a set of assignments to discrete variables.
  – $R$: a set of relaxations for some temporal constraints.

such that no variable in $T$ is left unassigned and $T$ is temporally consistent.

• The most preferred solution has the highest utility, evaluated by:

$$\sum_{a_i \in A} f_{p_i}(a_i: p_i \leftarrow v_i) - \sum_{r_j \in R} f_{e_i}(r_j: e_j \rightarrow e'_j)$$