Resolving Over-constrained Conditional Temporal Problems Using Semantically Similar Alternatives

Peng Yu
MIT
yupeng@mit.edu

Jiaying Shen and **Peter Z. Yeh**
Nuance Communications, Inc.
{Jiaying.Shen,Peter.Yeh}@nuance.com

Brian Williams
MIT
williams@mit.edu

Abstract

In recent literature, several approaches have been developed to solve over-constrained travel planning problems, which are often framed as conditional temporal problems with discrete choices. These approaches are able to explain the causes of failure and recommend alternative solutions by suspending or weakening temporal constraints. While helpful, they may not be practical in many situations, as we often cannot compromise on time.

In this paper, we present an approach for solving such over-constrained problems, by also relaxing non-temporal variable domains through the consideration of additional options that are semantically similar. Our solution, called Conflict-Directed Semantic Relaxation (CDSR), integrates a knowledge base and a semantic similarity calculator, and is able to simultaneously enumerate both temporal and domain relaxations in best-first order. When evaluated empirically on a range of urban trip planning scenarios, CDSR demonstrates a substantial improvement in flexibility compared to temporal relaxation only approaches.

1 Introduction

From an evening outing to a summer vacation, we frequently plan for travels of different length and complexities. Unfortunately, we are not good at estimating times and compensating for uncertainty, while often trying to do too much. These situations can lead to anywhere from being late for a dinner, to missing a flight. It would be of great help if our intelligent personal assistants, such as Siri and Google Now, can keep us informed about such issues, and provide advice on which activities and requirements should be modified, such that a robust travel plan can be found.

Such over-subscribed situations have often been modeled by over-constrained temporal problems. A temporal problem is over-constrained if no execution strategy [Dechter *et al.*, 1991; Vidal and Fargier, 1999] can be found that meets all constraints. To solve an over-constrained temporal problem, one has to identify its conflicting constraints and weaken some of them, such that all conflicts are resolved and a feasible execution strategy, either a static schedule or a dynamic policy, can be generated. In literature, several methods have been developed to solve such problems. [Beaumont *et al.*, 2001; 2004] took a partial constraint satisfaction approach to find subsets of satisfiable constraints for over-constrained Simple Temporal Problems (STPs). [Moffitt and Pollack, 2005a; 2005b; Peintner *et al.*, 2005] extended the approach to disjunctive temporal problems and with preferences (DTPs). In [Yu and Williams, 2013; Yu *et al.*, 2014], a conflict-directed approach was used to diagnose conflicting constraints in temporal problems with alternatives and uncertain durations, and compute continuous relaxations instead of suspension for constraints. However, prior works only resolve the problems through modifying the temporal constraints. While helpful, these suggestions may not be useful in many scenarios, as we are often not flexible with time. This motivates us to develop a new system that behave more like an experienced travel assistant, who can not only suggest relaxing timing requirements, but also recommend good alternative destinations in over-constrained situations.

In this paper, we present a new approach that enables such a capability: *Semantic Relaxation*. It gives the users more options and flexibility when many of their temporal constraints cannot be weakened. First, we model the travel planning scenario as Controllable Conditional Temporal Problems (CCTP, [Yu and Williams, 2013]), and augment the formulation with semantic constraints for decision variables. These semantic constraints, represented by SparQL queries, a W3C standard for querying data represented as RDF triples, encode the semantic meaning of the users' destinations and the domain values of the variables. Second, we developed a new algorithm, Conflict-Directed Semantic Relaxation (CDSR), to enumerate temporal and domain relaxations in best-first order. CDSR generalizes the conflict-resolution techniques in [Yu and Williams, 2013] and resolves conflicts by relaxing temporal constraints, as well as adding additional values to the variable domains through weakening their semantic constraints. The relaxation of temporal constraints are guided by a user-defined cost function, while the weakening of semantic constraints are guided by a semantic similarity model generated by the Word2Vec package [Mikolov *et al.*, 2013], which uses high-dimension vector representations of concepts trained on a large corpus of Natural Language data. The word vector approach has been shown to carry semantic

meanings when comparing concepts, and this is the reason we name our approach *Semantic Relaxation*.

Motivating Example

Consider a scenario in which a user, Simon, is planning for an evening outing trip with his intelligent travel assistant. Simon is leaving the office at 6pm, and would like to have dinner with a friend at a Chinese restaurant then watch a comedy movie. He needs to be home before 9:30pm, the time when his babysitter leaves. The CCTP for his travel problem is shown in (Figure 1), which encodes the two activities for dinner and movie, as well as the temporal constraints over the trip departure and completion times. The two tasks requested by Simon, *watch a comedy movie* and *dine at a Chinese restaurant*, are highlighted in bold. In addition, these tasks are associated with the following sets of SparQL queries that encode the genre and cuisine requirements (m.05p553 and m.01xw9 are Freebase Machine IDs for entity *Comedy film* and *Chinese food*).

- **Comedy Movie**: SELECT ?m WHERE{
  ?m ns:type.object.type ns:film.film.
  ?g ns:type.object.type ns:film.film_genre.
  ?m ns:film.film.genre ?g.
  FILTER (?g = <http://rdf.freebase.com/ns/m.05p553>).}
- **Chinese Restaurant**: SELECT ?r WHERE{
  ?r ns:type.object.type ns:dining.restaurant.
  ?c ns:type.object.type ns:dining.cuisine.
  ?r ns:dining.restaurant.cuisine ?c.
  FILTER (?c = <http://rdf.freebase.com/ns/m.01xw9>).}

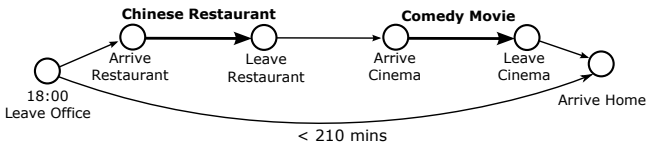

Figure 1: A CCTP for Simon's trip

Given the semantic constraints, we pass their SparQL queries to a knowledge base, which can search through multiple data sources and retrieve candidate options for the tasks [Yeh *et al.*, 2015]. These options will then be added as domain values for the activities with travel times encoded as conditional constraints. For example, the expanded problem for Simon's outing trip is shown in Figure 2. The comedy movie task is replaced by two movie showings at different theaters, while the Chinese restaurant task is replaced by two restaurant options. Each grounded activity is also associated with a duration (highlighted in red in the figure). The constraints for traversals between locations (Table 1) are omitted from the graph to save space.

The solution to the CCTP is a set of choices for the dinner and movie tasks that satisfies all temporal constraints. Unfortunately, due to the long travel times to and from the candidate Chinese restaurants, no solution can be found that meets all temporal requirements. The travel assistant (TA) engages Simon (SI) and initiates a discussion about possible resolutions for his problem.

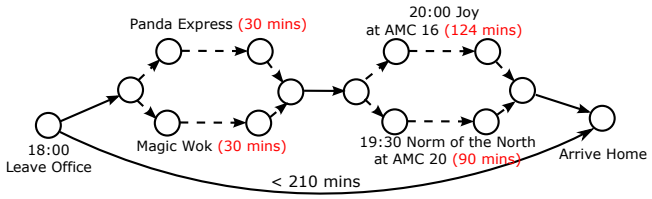

Figure 2: An expanded CCTP with options for each activity

Traversal Durations	Guard Assignments
Office → PE: [40,65]	$Dinner \leftarrow PE$
Office → MW: [30,35]	$Dinner \leftarrow MW$
PE → AMC 16: [25,45]	$Dinner \leftarrow PE$ $Movie \leftarrow JY$
PE → AMC 20: [35,55]	$Dinner \leftarrow PE$ $Movie \leftarrow NN$
MW → AMC 16: [35,45]	$Dinner \leftarrow MW$ $Movie \leftarrow JY$
MW → AMC 20: [40,55]	$Dinner \leftarrow MW$ $Movie \leftarrow NN$
AMC 16 → Home: [25,30]	$Movie \leftarrow JY$
AMC 20 → Home: [20,25]	$Movie \leftarrow NN$

PE for Panda Express, MW for Magic Wok, JY for movie Joy, NN for movie Norm of the North.

Table 1: Travel times between locations (in minutes)

TA: *You may have dinner at Magic Wok then watch the 8pm Joy at AMC 16. However, due to the length of the movie you won't be back home until 10:34pm. Is that OK?*

SI: *No, I cannot ask the babysitter to stay any longer.*

TA: *OK, then can you leave office 30 minutes earlier? If so you may watch Norm of the North at 7:30pm, and arrive home at 9:30pm.*

SI: *No I cannot leave office before 6pm.*

TA: ***How about eating at Sunny Bowl, a Korean restaurant?*** *It is closer and you can make the 7:30pm movie without leaving any earlier.*

SI: *Sounds good. Thank you.*

In this example, Simon cannot change the departure and arrival times, hence he rejected the first two proposals. Previous approaches would have failed at the step, since no more temporal relaxation can be found that resolves the conflicts between long travel times to the restaurants and movie start times. However, CDSR is able to explore relaxations along a different dimension and find Simon a satisfactory solution: it weakened the semantic constraints for the restaurant task, such that three new options became available for his trip (Figure 3). In this case, CDSR discovered a close alternative, Korean, for the cuisine requirement of restaurant. It then queried the knowledge base to retrieve additional restaurant candidates, and found one that is closer to Simon's route and satisfies all temporal constraints (Figure 4).

This example demonstrates the desired feature of CDSR: resolving over-constrained conditional temporal problems through relaxing the domain requirements, and actively searching for candidate solutions that are not encoded in the

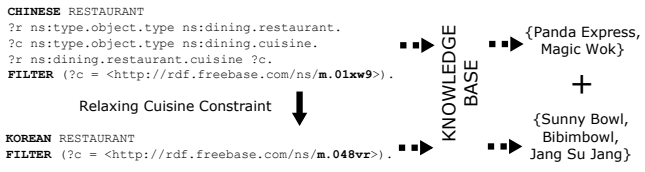

Figure 3: Domain relaxation for the restaurant cuisine

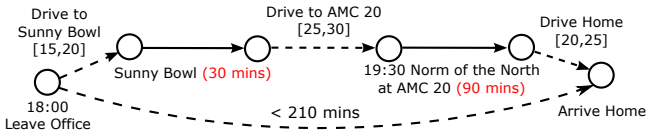

Figure 4: A solution enabled by relaxed cuisine constraint

original problem. It gives the users higher chance of and more flexibility in resolving conflicts in their problems. CDSR has been incorporated as part of a planning assistant, and demonstrated its effectiveness in helping users solve travel planning problems in urban scenarios. In the rest of the paper, we discuss the design and implementation of CDSR in detail.

2 Problem Statement

The input to CDSR is a temporal problem encoded using an augmented Controllable Conditional Temporal Problem formulation. CCTP was originally presented in [Yu and Williams, 2013] as an extension to the Simple Temporal Problem formulation (STP [Dechter *et al.*, 1991]). It adds discrete choices and conditional constraints labeled with the outcomes of choices, and supports relaxable temporal constraints. CCTP shares most of the notations with other conditional temporal problem formulations, such as Conditional Temporal Problem [Tsamardinos *et al.*, 2003], Temporal Network with Alternatives [Barták and Cepek, 2007], and Conditional Temporal Problems with Preferences [Falda *et al.*, 2010], with two distinct features. First, all variables are controllable. To determine the consistency of a CCTP, it is sufficient to find one consistent set of discrete choices. Second, CCTP extends the domains of discrete variables from binary to any finite domains, and allows the discrete variables to be conditioned on assignments to other variables. We first repeat the definition of CCTP, then present the extension required by CDSR for encoding semantic constraints.

Definition 1 *A CCTP is a 9-tuple* $\langle P, Q, V, E, RE, L_e, L_p, f_p, f_e \rangle$*, where:*

- P *is a set of finite domain variables;*
- Q *is the collection of domain assignments to* P*;*
- V *is a set of events representing designated time points;*
- E *is a set of temporal constraints,* $v_i - v_j \in [l, u]$*, defined over events* $v_i, v_j \in V$*;*
- $RE \subseteq E$ *is a set of relaxable constraints whose temporal bounds can be relaxed;*
- $L_e : E \rightarrow Q$ *is a function that attaches conjunctions of assignments* $q_i \in Q$*, to some temporal constraints* $e_i \in E$*;*
- $L_p : P \rightarrow Q$ *is a function that attaches conjunctions of assignments* $q_i \in Q$*, to some variables* $p_i \in P$
- $f_p : Q \rightarrow \mathcal{R}^+$ *is a function that maps each assignment to discrete variable,* $q_i \in Q$*, to a positive reward value;*
- $f_e : (e_i, e_i') \rightarrow r \in \mathcal{R}^+$ *is a function that maps the relaxation to constraints* $e_i \in RE$*, from* e_i *to* e_i'*, to a positive cost value;*

We extended CCTP with semantic constraints S and function L_s, such as the example presented in Figure 2, where:

- S is a set of semantic constraints, each $s_i \in S$ is a SparQL query;
- $L_s : P \rightarrow S$ is a function that attaches semantic constraints, $s_i \in S$, to some variables $p_i \in P$, which defines the domain of the variable.

To avoid increasing the length of the acronym, we chose to continue using the term CCTP for the extended formulation in this paper. As presented in the previous section, some of the discrete variables are associated with semantic constraints to encode the meanings of their domains. These semantic constraints, represented by SparQL queries, are used to retrieve domain values, encoded as object bindings, from knowledge bases. In the case of over-constrained problems, some of the triples in the queries can also be relaxed, which effectively weakens the semantic constraint and provides additional options to the domain for conflict resolution. Formally, we encode the SparQL query as a 4-tuple $\langle N, H, W, RW \rangle$, where:

- N is the namespace of the query;
- H is the SELECT clause, which identifies the variables to appear in the query results;
- W is a collection of SparQL query triples, each contains a subject, predicate and object field. They are encoded as part of the WHERE clause;
- $RW \subseteq W$ is a set of relaxable triples, whose object field can be modified to other values.

For example, Figure 3 demonstrates the relaxation to a semantic constraint for the $Restaurant$ variable. The original set of triples in the SparQL query retrieves only two Chinese restaurants, and neither of which can meet Simon's tight temporal requirements. A domain relaxation for this variable weakens the FILTER triple, allowing its object to be Korean cuisine in addition to Chinese cuisine, which adds three more restaurants to be considered. Internally, all objects are encoded using their unique Freebase Machine IDs (MIDs), such as m.01xw9 for Chinese cuisine and m.048vr for Korean cuisine, to avoid ambiguity.

Note that it is also possible to include uncertain temporal durations in the formulation, as the CCTPU formulation presented in [Yu *et al.*, 2014]. They are necessary for generating robust solutions for many real-world problems. For simplicity, we omitted the uncertain durations from our extended formulation. The CDSR algorithm is capable of checking controllability and compute relaxations robust to the uncertain durations, when configured with the corresponding controllability checking and conflict extraction functions.

The output of CDSR is a consistent set of choices, given the temporal relaxations for some constraints, and domain relaxations for some variables. Formally, we define the solution to a CCTP as the following:

Definition 2 *The solution is a 4-tuple* $\langle A, R_e, R_d, E' \rangle$*, where:*

- A *is a complete set of assignments to variables in* P*.*
- R_e *is a set of temporal relaxations for some constraints in* RE*.*
- R_d *is a set of domain relaxations for some variables in* P*.*
- E' *is a set of additional constraints that encodes the duration and traversal times associated with the additional options added by* R_d

In addition to new domain values, domain relaxation also introduces new constraints into the problem, such as the travel times to and from the new restaurant. In this paper, we discuss the application of CDSR to travel problems with location-tagged activities, since the additional temporal constraints in these scenarios are straightforward to compute. Generating constraints for domain relaxations in some other domains can be very difficult, and it is not the focus of this paper.

3 Approach

In this section, we present the design and implementation of CDSR, which resolves over-constrained CCTPs using both temporal and domain relaxations. CDSR leverages ideas from conflict-directed diagnosis and relaxation algorithms in the literature: it uses the conflict-directed framework from [Williams and Ragno, 2002] for efficient conflict detection and resolution, and generalizes methods from the Best-first Conflict-Directed Relaxation algorithm (BCDR, [Yu and Williams, 2013]) for enumerating discrete and continuous relaxations. Algorithm 1 presents the pseudo code of CDSR.

Compared to the BCDR algorithm, CDSR introduces two key modifications to support the enumeration of domain relaxations (highlighted in bold):

- First, in addition to temporal constraint relaxations, the conflict resolution step (Line 24) was extended to also compute domain relaxations.
- Second, an additional step of relaxation expansion is added for candidates with partially initiated domain relaxations (Line 7). During conflict resolution, CDSR only computes partial domain relaxation candidates, which are not initiated until being dequeued. It allows CDSR to delay the knowledge base queries and semantic similarity comparison, which are computationally expensive operations.

In addition, the CDSR algorithm integrates with a large-scale knowledge base to access the world knowledge, such as restaurants, movies and showtimes, for computing domain relaxations. This knowledge base is constructed from a combination of open and proprietary sources of content using an ingestion pipeline [Noessner *et al.*, 2015]. It transforms the raw content into RDF triples and performs entity resolution, that is, merging duplicate entities across different content sources. The resulting knowledge base can be viewed as a very large knowledge graph where the nodes represent entities and the edges represent semantic relations between these entities. The entities are typed, and a proprietary subsumption hierarchy is used to organize these types. The semantic relations have domain and range constraints, and also capture inverse relationships. This knowledge graph can be efficiently accessed and queried via SparQL.

Input: An extended CCTP with semantic constraints.
Output: A consistent solution that maximizes $f_p - f_e$.
Initialization:
1 $Cand \leftarrow \langle A, R_e, R_d, E', C_r, C_{cont} \rangle$; the first candidate;
2 $Q \leftarrow \{Cand\}$; a priority queue of candidates;
3 $C \leftarrow \{\}$; the set of all known conflicts;
4 $U \leftarrow P$; the list of unassigned variables;
Algorithm:
5 **while** $Q \neq \emptyset$ **do**
6 $Cand \leftarrow$DEQUEUE(Q);
7 **if** **EXPANDDOMAINRELAXATION**$(Cand, Q)$ **then**
8 CONTINUE;
9 **endif**
10 $currCFT \leftarrow$UNRESOLVEDCONFLICTS$(Cand, C)$;
11 **if** $currCFT == null$ **then**
12 **if** *isComplete?*$(Cand, U)$ **then**
13 $newCFT \leftarrow$FEASIBILITYCHECK$(Cand)$;
14 **if** $newCFT == null$ **then**
15 `return` $Cand$;
16 **else**
17 $C \leftarrow C \cup \{newCFT\}$;
18 $Q \leftarrow Q \cup \{Cand\}$;
19 **endif**
20 **else**
21 $Q \leftarrow Q \cup$EXPANDONVARIABLE$\{Cand, U\}$;
22 **endif**
23 **else**
24 $Q \leftarrow Q \cup$**EXPANDONCONFLICT**$\{Cand, currCFT\}$;
25 **endif**
26 **end**
27 `return` $null$;

Algorithm 1: An overview of the CDSR algorithm

Note that CDSR does not expand the new candidate with semantic relaxation during conflict resolution. Due to the size of the knowledge base, it is deployed on a dedicated server and CDSR communicates with it through network connections. On average, one query takes 500 ms to complete, which is very significant compared to temporal relaxations. While implementing CDSR, it is important to delay the expansion of semantic relaxation candidate as late as possible. Therefore, we only do the expansion (Function EXPANDDOMAINRELAXATION) after a candidate is dequeued.

Resolving Conflicts using Domain Relaxations

Function EXPANDONCONFLICT (Algorithm 2) expands the search tree using new candidates computed from the resolutions to known conflicts. Two options have been used by prior approaches in this procedure: (1) changing the assign-

Input: A candidate $cand$: $\langle A, R_e, R_d, E', C_r, C_{cont}\rangle$ and an unresolved conflict $currCFT$.
Output: A set of candidates $Cands$ that resolves $currCFT$.
Initialization:
1 $Cands \leftarrow \{\}$;
Algorithm:
2 **for** $a \in A$ **do**
3 $\quad A_{alter} = A_{alter} \cup$ GETALTERNATIVES(a);
4 $\quad A_{alter} = A_{alter} \cup$ GETALTERNATIVES$(guard(a))$;
5 $\quad$ **if** ISRELAXABLE$(variable(a))$ **then**
6 $\quad\quad A_{alter} = A_{alter} \cup \{variable(a) = SthElse\}$;
7 $\quad$ **end**
8 **end**
9 **for** $a_{alt} \in A_{alter}$ **do**
10 $\quad$ **if** NOTCOMPETING(A, a_{alt}) **then**
11 $\quad\quad cand_{new} \leftarrow \langle A \cup \{a_{alt}\}, R_e, R_d, E', C_r, C_{cont}\rangle$;
12 $\quad\quad Cands \leftarrow Cands \cup \{cand_{new}\}$;
13 $\quad$ **end**
14 **end**
15 $Cands \leftarrow Cands \cup$ $\{$CONTINUOUSLYRELAX$(cand, C_{cont} \cup \{currCFT\})\}$;
16 `return` $Cands$;

Algorithm 2: Function EXPANDONCONFLICT

Input: A candidate $cand$: $\langle A, R_e, R_d, E', C_r, C_{cont}\rangle$. And the search queue Q.
Output: A boolean value indicating if special assignment $SthElse$ was detected.
Algorithm:
1 **for** $a \in A$ **do**
2 $\quad$ **if** ISSTHELSE(a) **then**
3 $\quad\quad$ **for** $q \in$ RELAXABLETRIPLES$(var(a))$ **do**
4 $\quad\quad\quad Q_{alt} \leftarrow$ GETOPTIONS$(q, var(a))$;
5 $\quad\quad\quad q_{sim} \leftarrow$ GETSIMILAR$(Q_{alt}, var(a), R_d)$;
6 $\quad\quad\quad$ **for** $\langle a_r, E_r\rangle \in$ QUERYKB$(var(a), q_{sim})$ **do**
7 $\quad\quad\quad\quad cand_{new} \leftarrow \langle A \setminus \{a\} \cup \{a_r\}, R_e, R_d \cup \{q_{sim}\}, E' \cup E_r, C_r, C_{cont}\rangle$;
8 $\quad\quad\quad\quad Q \leftarrow Q \cup \{cand_{new}\}$;
9 $\quad\quad\quad$ **end**
10 $\quad\quad$ **end**
11 $\quad\quad$ `return` $True$;
12 $\quad$ **end**
13 **end**
14 `return` $False$;

Algorithm 3: Function EXPANDDOMAINRELAXATION

ments to discrete variables, and (2) relaxing temporal bounds of constraints. For (1), Function GETALTERNATIVES collects all alternative domain values for assignments in the conflict, and uses the ones that are not competing with any existing assignments (Function NOTCOMPETING) to generate new candidates. For (2), Function CONTINUOUSLYRELAX evaluates the temporal constraints in the conflict, and weakens the bounds of some relaxable ones to resolve it. CDSR introduces a third option: domain relaxation using semantically similar options. This requires an additional step in the function (Line 5-7): if the discrete variable involved in the conflict has a relaxable domain, a special assignment for the variable, called $SthElse$ (something else), will be added in addition to alternatives already encoded in its domain. This assignment will be used to generate a new candidate that resolves the conflicts.

When CDSR dequeues a candidate, Function EXPANDDOMAINRELAXATION (Algorithm 3) will first check if it has any such special assignment (Line 2). If so, it will iterate through all relaxable triples in the variable's semantic constraint (Line 3), and extract alternative objects from the knowledge base for them (Line 4). Next, given all alternative options (Q_{alt}) for a triple, Function GETSIMILAR selects the one that has the highest semantic similarity score to the original one, but have not been used before (not in R_d). CDSR then queries the knowledge base with the new triple q_{sim}, and generates new domain assignment a_r and constraints E_r for each of the result. Finally, a new candidate is created from each of them and added to the queue (Line 8).

We demonstrate this procedure using the travel example from Section 1. One conflict in Simon's plan is that he cannot go to restaurant Magic Wok and watch Joy at AMC 16 while arriving home before 9:30pm (Figure 5). Function EXPANDONCONFLICT generated four new candidates from resolutions to this conflict: two from alternative domain assignments in variable $Dinner$ and $Movie$, one from continuous relaxation for the trip duration constraint, and one from semantic relaxation for variable $Dinner$. After CDSR has evaluated all other candidates and found them to be infeasible or rejected by the users, it dequeues the domain relaxation candidate $Dinner \leftarrow SomethingElse$ and compute grounded options for it. Function EXPANDDOMAINRELAXATION takes in this special assignment and extracts the relaxable triple of cuisine in the variable's semantic constraint. It then identifies the most similar alternative object, Korean, for the original triple with Chinese. Finally, using the new cuisine triple, three new options for $Dinner$, Sunny Bowl (SB), Bimbibowl (Bb) and Jang Su Jang (JSJ) were found by querying the knowledge base, and used to resolve the conflict and expand the search tree.

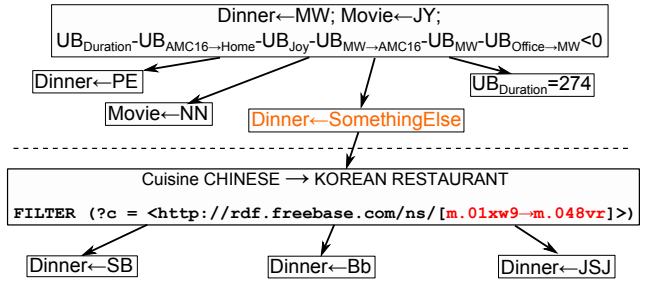

Figure 5: Expansion with temporal and domain relaxations

Computing Good Domain Relaxations
Given a set of alternative triples, Function GETSIMILAR calculates the similarity scores between the object in the original triple to them, and returns the most similar one. For example, given that no Chinese restaurant fulfills Simon's requirement, we would like to try Korean first instead of Mexican or

American, since they are less likely to be preferred by him. CDSR measures the similarity using a vector representation of words, first proposed in [Mikolov *et al.*, 2013]: each object (represented by its Freebase MID) is associated with a 1000-dimension vector of numbers. The vector representations are learned by neural network model based on a huge corpus of natural language data, which has been shown to capture semantic properties very well. The similarity score between two objects is computed using the cosine distance between their vectors: higher scores mean more similar. The vector model of CDSR is trained by the continuous skip-gram algorithm in the *Word2Vec* package with a Google News dataset [Word2Vec, 2013]. For example, Figure 6 presents the semantic similarity scores between a few alternative cuisines and movie genres in Simon's problem. Out of four alternative cuisines to Chinese, Korean and Thai are very close, with similarity scores of 0.7140 and 0.6945, while Mexican and American cuisine only have scores of 0.5169 and 0.3183, respectively. With the semantic similarity measurement, CDSR is able to explore the domain relaxations in best-first order, like prior approaches did for temporal relaxations.

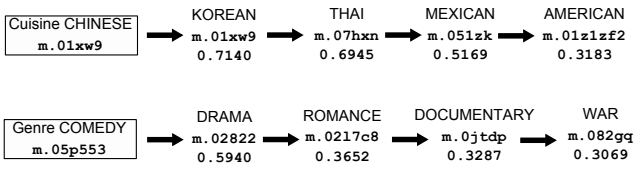

Figure 6: Semantic distances between cuisines and genres

Note that the model CDSR is using only covers about 1.4 million out of 50 million topics in Freebase due to the corpus used in training. When asked to compare with an object with an undefined vector, the similarity measurement will return zero, which greatly reduces the chance that this object will be considered in relaxation. This limitation was observed during our experiments, and we are working on a backup measurement with better coverage to address this issue.

4 Empirical Evaluation

The objective of CDSR is to provide more options for users while resolving over-constrained travel planning problems. To evaluate the usefulness of CDSR in such scenarios, we conducted a user study using the personal assistant built with the algorithm. It is designed for users to manage their day-to-day tasks, and the study examines CDSR in two aspects: (1) can it help users find solutions in scenarios that would be impossible to solve by only temporal relaxations, and (2) is the quality of CDSR's domain relaxations acceptable in different scenarios. In this section, we present the design of the user study, and discuss the results and lessons learned.

4.1 Setup

The travel assistant behaves much like the example presented in the introduction section. In the user study we use a web-based GUI to interact with the participants, which provides step-by-step guidance for them to interact with the assistant. It operates on a set of template scenarios, and promotes the users to input their requirements and tasks for their trips, such as origin, destination, time of departure, and desired trip length. Once a solution is found by CDSR, it will be presented to the users both verbally using a story line, and visually using a polyline overlay on Google Map. For example, Figure 7 shows an example trip plan with a dinner and movie, with relaxations to the departure time and dinner duration.

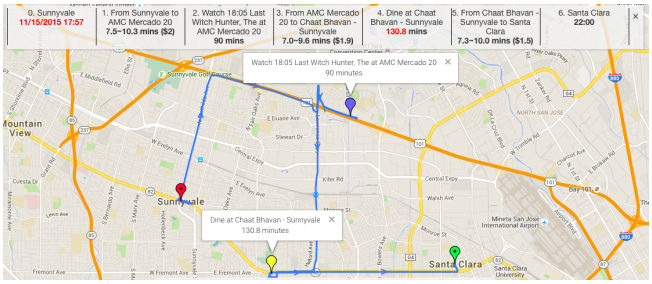

Figure 7: A trip plan presented in the web interface

There are six scenarios in this study, which are constructed based on commonly encountered travel planning problems, such as an evening outing, a date or a weekend get together for kids. The users were asked to plan for two to four tasks in a session, which can be either lunch, dinner or movie, subject to different departure and arrival time constraints. For example, one scenario is defined as the following:

You are planning for a monthly weekend get together with a small group of close friends. This trip includes a 2-hour lunch, a movie and possibly dinner. It starts from a meeting point you selected, and ends at your home.

Several of the scenarios are designed to be over-subscribed: the trip duration constraints are too tight for completing all tasks. Some trade-offs must be made to resolve the competing requirements in these scenarios. Finally, at the end of each session, we asked the participants to evaluate the last solution proposed by CDSR, and submit a quality score in the scale of 1 to 5. The score indicates if the user is satisfied with the solution, with 5 being very satisfied and 1 being not satisfied.

4.2 Results and Discussion

We received study results from nine different participants, for a total of 54 sessions. During the study, we recorded the problems specified by the participants, the number of *NextSolution* requests, the solutions generated by CDSR, and the quality scores. Using the problems recorded, we also evaluated how many of them can be solved with a temporal-only configuration after the study. CDSR found solutions and reached an agreement with the participants in 52 out of 54 sessions. In the solutions for five out of six scenarios, domain relaxation was used for resolving conflicts in the problems specified by the participants (Table 2). Compared to temporal-relaxation only approaches, which gave up on 11 sessions, the introduction of domain relaxation indeed provides the users more flexibility and higher chance of finding solutions for their over-constrained trips.

In addition to finding feasible resolutions to the conflicts, we are also interested in the quality of CDSR's solutions. The

#	Quality	Reject& NextSol	Temporal Relaxation	Domain Relaxation
1	3.3 (1.4)	2.9 (2.3)	2.0 (2.6)	2.1 (2.7)
2	2.4 (1.5)	3.5 (2.8)	1.3 (2.9)	3.0 (3.3)
3	2.7 (1.5)	4.7 (5.5)	2.9 (3.0)	3.1 (2.8)
4	3.7 (1.6)	3.1 (3.0)	0.3 (0.7)	1.7 (3.4)
5	3.2 (1.4)	2.9 (2.1)	1.9 (2.6)	1.7 (3.0)
6	3.3 (1.5)	1.2 (0.6)	0.6 (1.1)	0.0 (0.0)

Table 2: Average quality scores, *NextSolution* requests, temporal and domain relaxations (with standard deviation)

quality scores indicate that CDSR's solutions are acceptable, but not much preferred, as the average ratings are in the range of 2s and 3s. The scores are lower in scenario 2 and 3, for which more domain relaxations were used in the solutions (average 3.0 and 3.1 per session). This is likely caused by the issues in CDSR's preference model. First, as the cost functions defined over temporal relaxations and domain relaxations are not compatible (weighted linear cost vs. cosine distance), we normalized the distance by computing its inverse (1/distance) and used it as the cost for semantic relaxations. The results showed that this procedure does not penalize domain relaxations enough sometimes, which makes CDSR too aggressive in relaxing domains, even in some scenarios where slightly weakening temporal constraints is sufficient. In addition, some participants reported that CDSR lacks of a personalized preference model: it uses the same static cost functions over temporal relaxations, and vector distance models over domain relaxations for all users. While some users find it good in capturing their preferences, others may think CDSR's trade-offs do not make sense at all. This is the cause of the large variance in the quality scores, and is an important problem for future research. One alternative approach is to use a multi-objective preference model, which is likely to perform better in the integration of these different objective functions. Note that it will require a different configuration of CDSR's search queue for enumerating candidates along the pareto-front.

Finally, as presented in previous sections, the cost of computing domain relaxations is significantly higher than that of temporal relaxation. On average, each knowledge base query takes around 500 ms, and each semantic similarity calculation takes about 200 ms. Due to the size of the knowledge base and Word2Vec model for Freebase entities, they were deployed on separate servers from the one for CDSR. The delay in network connection is a big factor that affects CDSR's performance. Therefore, when using CDSR for domain specific applications, one may reduce the coverage of the knowledge base and similarity model for better run-time performance.

5 Contributions

In this paper, we present a new approach, Conflict-Directed Semantic Relaxation, for solving over-constrained conditional temporal problems. In addition to continuously relaxing temporal constraints, it also computes relaxations for variable domains. These domain relaxations allow more options to be added for resolving conflicts, and the additional options are semantically similar to existing ones. CDSR is able to simultaneously enumerate both temporal and domain relaxations in best-first order, and has been integrated with a knowledge base and a semantic distance calculator for finding good relaxation candidates. When evaluated empirically on a range of urban trip planning scenarios, CDSR demonstrates a substantial improvement in flexibility compared to temporal relaxation only approaches. As part of future work, we are actively working on improving the user preference function and a more robust semantic similarity model, which should provide solutions of higher quality for different users.

6 Acknowledgments

The authors would like to thank Deepak Ramachandran, Daniel Walker, Jonathan Raiman and Szymon Sidor for discussing and reviewing the work.

References

[Barták and Cepek, 2007] Roman Barták and Ondrej Cepek. Temporal networks with alternatives: Complexity and model. In *Proceedings of the Twentieth International Florida AI Research Society Conference (FLAIRS)*, pages 641–646, 2007.

[Beaumont *et al.*, 2001] Matthew Beaumont, Abdul Sattar, Michael Maher, and John Thornton. Solving overconstrained temporal reasoning problems. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AI-2001)*, pages 37–49, 2001.

[Beaumont *et al.*, 2004] Matthew Beaumont, John Thornton, Abdul Sattar, and Michael Maher. Solving overconstrained temporal reasoning problems using local search. In *Proceedings of the 8th Pacific Rim Conference on Artificial Intelligence (PRICAI-2004)*, 2004.

[Dechter *et al.*, 1991] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[Falda *et al.*, 2010] Marco Falda, Francesca Rossi, and K. Brent Venable. Dynamic consistency of fuzzy conditional temporal problems. *Journal of Intelligent Manufacturing*, 21:75–88, 2010.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Moffitt and Pollack, 2005a] Michael D. Moffitt and Martha E. Pollack. Applying local search to disjunctive temporal problems. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, pages 242–247, 2005.

[Moffitt and Pollack, 2005b] Michael D. Moffitt and Martha E. Pollack. Partial constraint satisfaction of disjunctive temporal problems. In *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS-2005)*, 2005.

[Noessner *et al.*, 2015] J. Noessner, D. Martin, P. Yeh, and P. Patel-Schneider. Cogmap: A cognitive support approach to property and instance alignment. In *ISWC*, 2015.

[Peintner *et al.*, 2005] Bart Peintner, Michael D. Moffitt, and Martha E. Pollack. Solving over-constrained disjunctive temporal problems with preferences. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, 2005.

[Tsamardinos *et al.*, 2003] Ioannis Tsamardinos, Thierry Vidal, and Martha Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8:365–388, 2003.

[Vidal and Fargier, 1999] Thierry Vidal and Helene Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, 11:23–45, 1999.

[Williams and Ragno, 2002] Brian C. Williams and Robert J. Ragno. Conflict-directed A* and its role in model-based embedded systems. *Journal of Discrete Applied Mathematics*, 155(12):1562–1595, 2002.

[Word2Vec, 2013] Word2Vec. Tool for computing continuous distributed representations of words. `https://code.google.com/archive/p/word2vec/`, 2013. Accessed: 2016-01-29.

[Yeh *et al.*, 2015] P.Z. Yeh, D. Ramachandran, B. Douglas, A. Ratnaparkhi, W. Jarrold, R. Provine, P.F. Patel-Schneider, S. Laverty, N. Tikku, S. Brown, J. Mendel, and A. Emfield. An end-to-end conversational second screen application for tv program discovery. *AI Magazine*, 36(3), 2015.

[Yu and Williams, 2013] Peng Yu and Brian Williams. Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 2429–2436, 2013.

[Yu *et al.*, 2014] Peng Yu, Cheng Fang, and Brian Williams. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*, 2014.