Towards Personal Assistants that Can Help Users Plan

Abstract. Intelligent personal assistants, such as Siri and Google Now, are pervasive, and can help users with a variety of requests such as checking the weather, recommending restaurants, getting directions, and more. However, the usefulness of these personal assistants quickly diminish when presented with requests that are no longer limited to a single goal or activity, such as a device command or a straight-forward information lookup. For example, these assistants cannot help users plan a night out after a hard day at work, which may involve a dinner and a movie. They do not understand that these complex requests involve multiple goals and activities with interrelated (and even competing) constraints and preferences that must be met accordingly.
In this paper, we present an intelligent personal assistant, called Uhura, that handles requests involving multiple, interrelated goals and activities by efficiently producing a coherent plan. Uhura achieves this by integrating a collaborative dialog manager, a conflict-directed planner with spatial and temporal reasoning capabilities, and a large-scale knowledge graph. We also present a user study that assesses the usefulness of the plans produced by Uhura in urban travel planning, and show that Uhura performs well on a wide range of scenarios.

1 Introduction

People are now using intelligent personal assistants for many day-to-day tasks. For example, we use Siri, Google Now or Cortana to send messages, check weather and find restaurants. Many of these assistants support both verbal and visual communications to make the interaction simpler. However, their functions are limited to simple commands and information retrieval tasks. None of them can understand user requests involving multiple goals and activities, which require planning and scheduling capabilities to properly fulfill.

There is also much research on advanced end-to-end personal assistants, but most of these assistants also do not support planning. For example, [18] reports an end-to-end personal assistant framework, but their work is focused on proactivity and task management. [17] and [7] report end-to-end personal assistants for TV program discovery, but these assistants are unable to plan (and hence cannot support complex user requests involving interrelated goals and activities). [3] reports a personal assistant that helps reduce email overload, but again, this assistant cannot support user requests that require planning. Finally, the human-machine collaborative planning prototype system developed by [1], and the task and time management assistant presented in [9] both support planning, but can only account for temporal and spatial constraints.

In contrast, our goal is to develop a personal assistant, called Uhura, that supports complex user requests that require planning (e.g., take kids out for a play date). Uhura can also consider semantic constraints (e.g., an action movie with Tom Cruise, a Greek restaurant with a good wine list), which occur frequently in many day to day scenarios. It achieves these benefits by uniquely utilizing three distinct components: Knowledge Base, Planner, and Collaborative Dialog Manager (CDM). First, Uhura uses a knowledge base that provides the data necessary (e.g., road condition, restaurants, movies, etc.) to formulate candidate plans that meet the users' semantic, temporal and spatial constraints. Second, Uhura integrates a dialog manager and a knowledge base with a planner that supports temporal and spatial reasoning, similar to the ones used in tourist assistant systems [15]. This allows Uhura to support semantic constraints as well as temporal and spatial ones. If there are competing requirements, the Planner will propose relaxations for them, and negotiate with the user until a resolution is reached. Finally, Uhura uses a dialog manager to: 1) serve as a mediator between the planner and knowledge base; 2) maintain the context of the interaction with the users along with their goals; 3) communicate with the users in a natural way. Results from a user study demonstrate the effectiveness of Uhura in helping users plan everyday tasks, especially in over-subscribed scenarios. In this paper, we discuss the roles of the above components and how we integrate them to assist planning.

2 Example

Consider a scenario in which a user works with Uhura to make a trip plan for an evening outing. The user plans to leave the office with a friend at 6pm, have dinner at a nice French restaurant and watch an animation movie. The user needs to be home before 9:30pm, since that is when the babysitter leaves. The trip planning problem is challenging since there is heavy traffic on the way, which may make some constraints in the user's overly packed evening infeasible. Uhura's task is to work out a robust plan with the user, which includes the choices for restaurant and movie showing, the sequence of these activities, and appropriate adjustments to the timing requirements, if necessary. During the first part of the conversation, the user communicate with Uhura about all her goals and requirements for the trip.

User: *I am going on an outing trip with friends this evening.*
Uhura: *OK, when are you planning to leave office.*
User: *6pm. Note that I have to be home before 9:30pm.*
Uhura: *What would you like to do this evening?*
User: *I want to watch an animated movie tonight, and eat at a French restaurant.*
Uhura: *Anything else?*
User: *No, that's it.*

Given this information, CDM, the dialog manager in Uhura, creates a Qualitative State Plan (QSP, [6]) to capture all tasks and requirements. QSP is an abstract plan that encodes all the desired states the user would like to achieve,

and the temporal requirements between them. Graphically, a QSP can be represented by a node-edge graph: each node represents an **event** in the plan, which is a unique point of time; each arc represents an **episode**, which can be either a goal state to achieve, or a temporal constraint that restricts the duration between time points. The semantic meaning of a goal state, such as *animated movie tonight*, is encoded as a set of first-order logic constraints. These FOL constraints capture all the user requirements that cannot be modeled by only using temporal or spatial constraints.

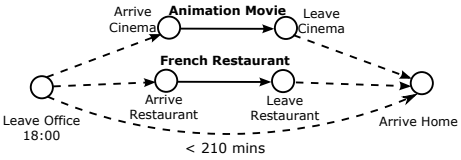

Fig. 1. A QSP for the user's trip

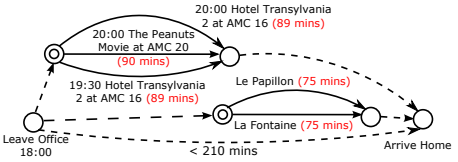

Fig. 2. Expanded QSP with grounded tasks

For example, the QSP for the user's evening trip is shown in Figure 1. There are six events in the graph, and each represents a time point associated with a task or a constraint, such as *Arrive Cinema* and *Leave Office*. The two tasks given by the user, watch an animated movie and dine at a French restaurant, are highlighted by the bold arcs. The tasks are associated with the following sets of semantic constraints that encode the genre and cuisine requirements.

- **Animated Movie**: (film m2) ∧ (surface m3 'animated') ∧ (genre m3) ∧ (cwGenre m2 m3)
- **French Restaurant**: (restaurant r2) ∧ (cuisine r3) ∧ (surface r3 'French') ∧ (servesCuisine r2 r3)

In addition, there are four dotted arcs connecting the tasks to the beginning and end of the trip, which represent temporal constraints that encode the sequencing requirement for the tasks: they must take place after leaving office, and finish before arriving home. Finally, there is one arc connecting the first and last events, which encodes the duration constraint of 210 minutes (from 6pm to 9:30pm) for the entire trip.

Once all the goals and requirements have been collected by CDM, it will pass the semantic constraints for each task to the Knowledge Base, which will search through the data sources and retrieve candidate options. These options will then be encoded as alternative episodes for the tasks and added to the QSP. For example, the expanded QSP for the user's evening trip is shown in Figure 2. The animated movie task is replaced by three grounded movie showings at two theaters, while the French restaurant task is replaced by two restaurants. In the QSP graph, the grounded options for each task share one common start event, which is represented by a double circle and indicates that the subsequent episodes are alternatives. In addition, each grounded task is associated with a duration (highlighted in red). These durations encode the length of the movie or the average time for dinner.

Next, Uhura passes the expanded QSP to the Planner to fill in the details of the plan, select the candidate for each task, and determine their orders while meeting the user's requirements. The result is a complete plan with grounded activities. In addition, the Planner uses an uncertainty-sensitive scheduling algorithm, which builds contingencies into the plan for likely delay during transit. If no feasible plan can be found that meets all requirements, as in this example, the Planner will also try relaxing some constraints or tightening durations in order to make room for completing all tasks.

For example, one plan for the evening trip is shown in Figure 3. It takes the user to dinner before movie, and selects La Fontaine for the French restaurant task and Hotel Transylvania 2 at AMC 16 cinema for the animated movie task. The planner fills in the driving activities between locations, and estimates the lower and upper bounds on their durations. Note that these traversal durations are *uncontrollable* [16], since the user cannot control exactly how much time it will take, as opposed to the other temporal constraints. Activities with uncontrollable durations are represented by double arcs in plan graphs. Finally, the trip duration constraint has been relaxed from 210 minutes to 240 minutes, which indicates that the user will not be home until 10pm.

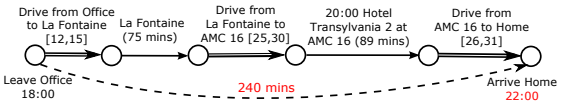

Fig. 3. One plan for the user's evening trip

Uhura: *You may eat at La Fontaine Restaurant then watch the 8pm Hotel Transylvania 2 at AMC 16. However, you won't be back until 10pm. Is that OK?*
User: *No, I have to be home before 9:30pm.*
Uhura: *OK, then can you shorten your dinner to 45 minutes? If so you may catch an earlier showing of the movie at 7:30pm, and arrive home before 9:30pm.*
User: *I prefer to spend at least an hour on dinner.*
Uhura: *How about leaving work at 5:45pm? I can then find a plan that meets all other requirements.*
User: *I am fine with that. Thank you Uhura.*

Once a candidate plan is found, Uhura will engage the user, discuss about the choices made and the relaxations they require. If the users are not satisfied with the proposed plan, they can ask Uhura to search for the next best one, with additional constraints that have not been encoded in the problem. This process continues until Uhura reaches an agreement with the users. In this example, the user is not satisfied with Uhura delaying the arrival home time or shortening the dinner, and rejects Uhura's proposals. Given the additional constraints, Uhura then computes new plans that respect them.

This example demonstrates the desired features of Uhura as a personal assistant that helps users with planning tasks. It works collaboratively with the users, generates plans that achieve the goals while respecting their constraints, and discusses trade-offs with the users in over-subscribed situations. In the fol-

lowing sections, we present the design and integration of Uhura's components that enable these capabilities.

3 System Architecture

Uhura provides the following features to simplify the planning process for the users: 1) natural language communication; 2) mixed initiative goal-directed interaction; 3) supports for multiple tasks and constraints; and 4) being robust to temporal uncertainty and over-subscription. They are supported by a coordinated system of three major components (Figure 4): Collaborative Dialog Manager (CDM), Planner and Knowledge Base. CDM handles the interactions with users, and elicits their goals and requirements as a *Qualitative State Plan* (QSP). It also takes the semantic constraints expressed by the user as well as the temporal and spatial constraints, and formulates queries for the Knowledge Base. The results of these queries ground the tasks in the QSP with additional episodes and constraints. Finally, the expanded QSP is sent to the Planner, which evaluates the alternatives of each task and produces plans that best meet the users' requirements.

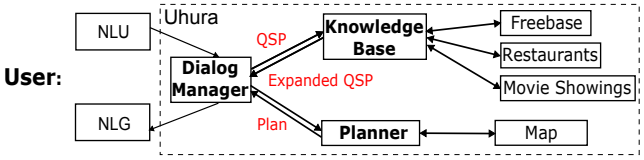

Fig. 4. The architecture graph of Uhura

3.1 Collaborative Dialog Manager

CDM [11] is an extension to Disco [13], an open source dialog development framework based on Collaborative Discourse Theory [5, 4, 8], and Sidner's artificial language for negotiation [14]. It views a personal assistant dialogue as a process of plan augmentation, where the purpose of the dialogue is for the system and the user to collaboratively form a complete sharedPlan in order to meet the user's intention. When the user makes an utterance, the system first attempts to interpret it as one of the built-in utterance types with relation to the current dialogue, using the semantic representation produced by the natural language understanding pipeline as its input. Then it attempts to form a meta plan with the assistance of the recipe library. The recipe library is a collection of Hierarchical Task Networks (HTNs) that capture the high-level task (goal) structures and is written in the ANSI/CEA-2018 standard [12]. Here we call the plan generated by the CDM a meta plan to distinguish it from the user activity plan generated by the Planner. This meta plan is to be executed by CDM, and its actions are often meta actions such as accessing the knowledge base for answering queries, or calling the Uhura planner to find a user activity plan. If a complete meta plan cannot be constructed, then one of the built-in utterance generation rule is fired and a system utterance is generated to acquire necessary information from the user to further the planning process. Some examples of such system utterance include soliciting missing task parameters and

asking to confirm a suggested change of original task parameters. These system utterances are then turned into natural language by the natural language generation pipeline, such as "When are you leaving the office?" and "How about leaving work at 5:45pm?". This cycle of interaction and planning continues until a complete plan can be formed and the user requests are met.

Figure 5 shows two of the key recipes for our activity planning example domain. Here the nodes represent tasks, some of which can be further decomposed, and others are directly executable. The solid arrows represent decomposition step relations, while the dotted arrows represent alternative decompositions. Figure 6 shows the meta plan generated as a result of the example dialog shown in the previous section.

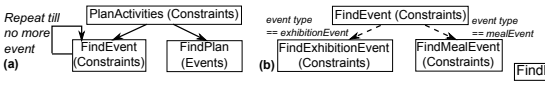

Fig. 5. Recipes for the example domain

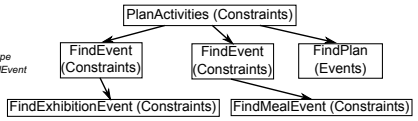

Fig. 6. The CDM meta plan from the example dialog

The plan based approach of CDM puts it as the processing hub of the system, whose assembled meta task plan often includes knowledge base querying (the FindEvent tasks) and Uhura planner invocation (the FindPlan task). On the other hand, it is also the information hub of the system, since CDM is responsible for assembling all user requests to form legitimate knowledge base queries, integrating knowledge base query results into a valid QSP input to the Uhura planner as well as interacting with the user to communicate all the information generated by Uhura. In order to effectively carry out all these responsibilities, CDM uses first order logic (FOL) encoded as semantic graphs to store and process this vast amount of information from various sources. Figure 7 shows an example of semantic graph. The green portion is the graph generated from only the initial user request without further system actions. In the following sections, we will present how the semantic graph is used to communicate with both the knowledge base and the planner, and integrate their results.

3.2 Knowledge Base

Uhura uses a large-scale knowledge base to access the world knowledge, such as restaurants and movie showtimes, to properly construct a plan that satisfies the user's request. This knowledge base is constructed from a combination of open and proprietary sources of content using an ingestion pipeline [10] that 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 (e.g. "michael jordan" is a basketball_player and a person), and a proprietary subsumption hierarchy is used to organize these types. The semantic relations have domain and range constraints,

and also capture inverse relationships. Moreover, this knowledge graph can be efficiently accessed and queried via SparQL, a W3C standard for querying data represented as RDF triples. Our knowledge base has over 1 billion triples, and the processing time for each query is typically less than a few hundred milliseconds.

3.3 Planner

The planner fills in the details of the abstract plan generated by CDM and Knowledge Base, schedules each activity, and adds contingencies for likely delays during transit. Uhura's planner is implemented based on a constraint relaxation algorithm, Conflict-Directed Relaxation with Uncertainty (CDRU, [19]), which was first developed to solve over-constrained conditional temporal problems with uncertain durations. The algorithm uses a conflict-directed search strategy to prune infeasible candidates and find the optimal set of choices. In addition, CDRU is able to detect competing constraints in the QSPs, generate concise explanations for the cause of failure, and suggest trade-offs for the users to resolve the issues.

The Planner takes in a QSP as input, and produces a plan, as well as temporal relaxations for some constraints if necessary. Formally, we define a Qualitative State Plan with grounded task options, such as the one presented in Figure 2, as an 8-tuple $\langle P, Q, V, E, RE, L_e, f_p, f_e \rangle$, where:

- P is a set of finite domain variables defined over the alternative options for each task;
- Q is the collection of domain assignments to P;
- V is a set of events representing designated time points;
- E is a set of episodes between pairs of events $v_i \in V$;
- $RE \subseteq E$ is a set of relaxable episodes whose temporal bounds can be relaxed;
- $L_e : E \rightarrow Q$ is a function that attaches conjunctions of assignments to P, $q_i \in Q$, to some episodes $e_i \in E$, which controls the activation and deactivation of the episodes;
- $f_p : Q \rightarrow \mathcal{R}^+$ is a function that maps each assignment to discrete variable, $q_i \in Q$, to a positive reward;
- $f_e : (e_i, e_i') \rightarrow r \in \mathcal{R}^+$ is a function that maps the relaxation to relaxable episode $e_i \in RE$, from e_i to e_i', to a positive cost.

The planner's output is a 4-tuple, $\langle A, S, R, T \rangle$, where:

- A is a complete set of assignments to variables in P.
- S is a set of additional assignments that defines a total ordering over tasks in the QSP.
- R is a set of relaxations for some episodes in RE.
- T is a set of additional episodes that encodes the traversal activities between tasks, following the order defined by S.

The plan defines a feasible sequence of activities that achieves all tasks specified in the input QSP, as well as the relaxations required to enable these activities. Note that the variables in S and episodes in T are not encoded in the input QSPs. Instead, they are generated during task sequencing, an important feature that is not supported by the original CDRU algorithm. We introduce

a new global constraint, $PATH$, into CDRU to provide this capability. $PATH$ is commonly used in modeling vehicle routing problems: it is one of the global constraints over discrete variables that ensures the vehicle visits all locations following a valid sequence. For Uhura's planning problems, we define the $PATH$ constraint over the task sequence variables, that is, the *what to do next* variable for each task. In order to check candidate plans against $PATH$ constraint, we added an additional PropagatePATH function before the temporal feasibility checking, which is implemented based on the propagation function introduced in [2]. This approach is efficient in identifying invalid task sequences and conflicting assignments, and signaling CDRU to backtrack and try different orders.

4 Approach for Integration

The three major components of Uhura have different responsibilities and applications, and *speak* very different languages. The key to an effective integration is to **disintegrate** the overall problems properly, assign the subproblem to the component that has the right reasoning capability, and supply them with the right set of data. Inside Uhura, CDM is responsible for interacting with the users and capturing the planning problems from them. It creates and assigns subproblems that require temporal and spatial reasoning to the Planner, and subproblems that require semantic reasoning to the Knowledge Base. In this section, we present the interfaces for CDM to create these subproblems, and retrieve results from the other two components.

CDM-Knowledge Base Interface CDM often needs access to background knowledge in order to ground the various events that meet user's requirements so that Uhura can generate a satisfactory activity plan for the user. However, CDM encodes all the event constraints in semantic graph, while the knowledge base uses SparQL as its query language. Additionally, the semantic graph encodes only the constraints that the user has expressed so far, while the SparQL query needed by the knowledge base needs to be very specific and complete with regard to all the information to be returned. For example, in Figure 7, the green part shows the original semantic graph generated by CDM that is equivalent to the user request for "an animated movie". However, it says nothing about the theater where the movie is shown, or the date and time of the showing. The acquisition of this information is essential for the planner to successfully plan for the movie activity. In order for the two components to talk to each other, we need to bridge these two discrepancies.

We designed two auxiliary components to address these issues. Based on the task information from CDM's task library, a simple query reasoner expands the original semantic graph with new query target nodes representing any missing information to be retrieved from the knowledge base in order to completely ground the requested events. It is also responsible for filling in default information such as time and location, if users do not specify them. After the original semantic graph is thus augmented, a translator maps the FOL predicates that encode various constraints into SparQL relations so that a valid knowledge base query can then be generated.

When the knowledge base returns a list of results that meet all the constraints specified in the query, CDM in turn needs to integrate these grounded event instances into its semantic graph. Figure 7 shows an example of the expanded semantic graph that integrated the two ground instances of the showing of an animated movie. This expanded semantic graph is also the base for which a QSP is generated so that the planner can step in and find a valid activity plan. Note that the system does not search through all candidate options in the knowledge base: given the current location of the users and the task descriptions, we can often narrow the domain of possible options down to a few dozen.

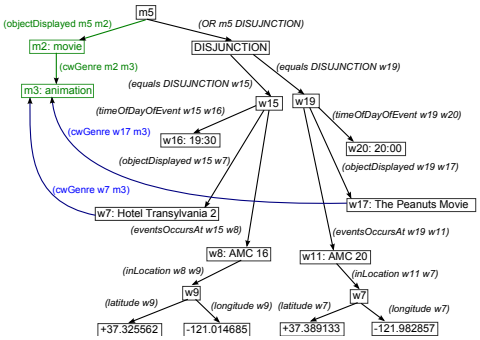

Fig. 7. Semantic graph with grounded tasks

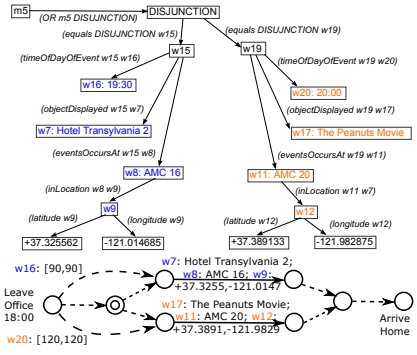

Fig. 8. The QSP generated from a semantic graph branch

CDM-Planner Interface If the users' problems require any temporal and spatial reasoning, such as traveling between places, CDM will pass them to the planner after grounding the tasks with options from the Knowledge Base. The Planner then evaluates different choices and sequences for the tasks, and generates plans that meet the users' requirements. As mentioned in the previous section, the output from CDM is a set of first order logic expressions encoded in a semantic graph, while the input to the planner is a set of goals and requirements encoded as a Qualitative State Plan. A mapping needs to be created between them such that (1) each task's temporal and spatial requirements can be extracted from the semantic graph and encoded in the QSP; and (2) the choices and relaxations in the Planner's output can be mapped back to the nodes in the semantic graph, such that CDM can present them to the user.

We take a breadth-first approach to explore the graph, and extract any temporal and spatial information around a task. The exploration starts from the root of the branch for a task specified by the users, such as $m5$ in Figure 7. We then iterate through all its children, which connect to the root through a *DISJUNCTION* node, to search for grounded candidates from the knowledge base. A task variable is created for the root node, with the grounded options for the task as the variable's domain assignments. Next, we examine the branches under each of the child nodes to extract details about the grounded option. Every time an edge with certain labels is visited, such as *eventOccursAt* and *inLocation*, a corresponding function will be executed to extract the data stored in its end

node. The data is then associated with the episode for the candidate, either as name, location, or temporal constraints. Finally, during the exploration, a map is created from each QSP element to the corresponding semantic graph node. The map is later used by CDM to interpret the Planner's solutions.

For example, Figure 8 shows a branch of the semantic graph for a movie task and the QSP generated from it. The graph encodes two movie showings that meet the description *animation movie*. In the equivalent QSP, each of the movie showings is modeled by an episode, with theaters and their locations. In addition, the temporal constraints connected to the start event of each episode encode their start times. The mapping between the nodes in the branch and the QSP constraint is represented by the node IDs tagged to the episodes' names, locations and durations. Once a plan is generated, we can use the mapping to convert Planner's decision, such as selecting the episode with node tags $w11, w12, w17$, to a natural language output, *You can watch The Peanuts Movie at AMC 20.*

5 User Study

In this section, we present a user study on the usefulness of Uhura in the context of a personal assistant for managing day-to-day tasks. The user study was conducted using a web interface, which provides step-by-step guidance for the users to interact with Uhura. It operates on a set of templates, and will prompt the users to input the requirements and goals for their trips, such as origin, destination, and desired length of trip. Each time a user provides new trip related data, the web interface will send it to CDM, which decodes the data and incorporates it into a QSP for the user's trip. Once the user is done providing trip information, a plan query is sent to Uhura, which starts searching for plans for the QSP. When a plan is found, it is presented to the users, both visually and verbally. Finally, if the user is not satisfied, they can send a NextSolution request to Uhura for a different plan.

There are six sessions in this study, each presenting a different scenario. These scenarios are based on commonly encountered urban travel problems, such as an evening outing or a weekend kids playdate. The users are asked to plan for two or more tasks per session. For example, one scenario is:

Your relatives are visiting today, and you are planning to take them out in the afternoon. This trip may include a lunch, a movie and possibly a dinner. It starts from your home, and ends at the airport they are flying out.

At the end of each session, the participants are asked to evaluate the last plan proposed by Uhura, and grade it on quality and novelty (on a 5 point Likert scale). The quality score captures the user's satisfaction with the plan. The novelty score indicates if the solution is something new to the participant, 5 being very novel and 1 not at all.

Results and Discussion We received results from ten participants. Each participant worked on 6 sessions, for a total of 60 sessions. During each session, we recorded the problems specified by the participants, the number of *NextSolution* requests in each session, the solutions generated, and the quality and novelty scores. In the study, we limited the type of responses from the users to [Yes,No]

for each plan to simplify the results. With the complete system, the user can ask for alternatives of one destination, and fine-tune the relaxations for timing constraints. On average, the urban trips specified by the participants contains 4 tasks, with around 30 options for each. The resulting QSPs have around 500 episodes with a dozen discrete variables, and is usually solved within a few seconds. From the users' perspective, the delay in response is not longer than many popular routing applications. The study result is summarized in Table 1.

Session	Quality	Novelty	# of Solutions Generated
1	3.2 (1.40)	3.8 (1.08)	4.8 (6.14)
2	2.4 (1.43)	2.9 (1.22)	4.8 (4.60)
3	2.9 (1.58)	3.9 (0.83)	4.5 (5.30)
4	3.8 (1.54)	3.8 (1.17)	2.9 (2.91)
5	3.3 (1.35)	3.4 (1.20)	2.9 (1.97)
6	3.3 (1.42)	3.8 (1.08)	3.0 (5.37)

Table 1. Average scores and next solution requests in each session (standard deviation)

In general, participants found Uhura to be useful in planning daily tasks. The plans generated are acceptable in most situations, and the average quality score is above 3. Participants also gave feedback that Uhura simplified the otherwise complicated planning tasks. Without Uhura, planning a day trip may take minutes or even hours. With Uhura, a feasible solution can be found in seconds. Finally, Uhura occasionally produced plans that are novel to the participants, as the average novelty scores are above 3 for most scenarios.

On the other hand, we also discovered a few issues related to Uhura's architecture and implementation in the study. Some participants reported that Uhura is not making good decisions. For example, Uhura may decide to shorten the dinner time in order to get the participant home on time, while the participant may care more about spending enough time on dinner. This is the result of Uhura's lack of personalized preference model. Some participants also reported that Uhura occasionally produced nonsensical plans which can easily be avoided by applying common sense reasoning. For example, one user asked for two meals (lunch and dinner), and Uhura scheduled them back to back. We believe that these are the reasons for the large variance in the quality scores, and are important issues for us to address in the future.

6 Conclusion

In this paper, we presented Uhura, an intelligent personal assistant that helps users plan. Uhura coordinates a spectrum of subsystems, including a collaborative dialog manager, a planner and a knowledge base, to support the communication and solution of complex urban trip planning problems. We also presented a user study that demonstrates Uhura's effectiveness. Based on qualitative user feedback, the users found that Uhura can provide a coherent solution to complex planning requests. As part of future work, we are working to address limitations surfaced in the user study. These include better preference models and common sense reasoning in Uhura's planning process, which will provide higher quality plans for the users.

References

1. Allen, J., Ferguson, G.: Human-machine collaborative planning. In: Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space. pp. 27–29 (2002)
2. Francis, K.G., Stuckey, P.J.: Explaining circuit propagation. Constraints 19(1), 1–29 (2014), http://dx.doi.org/10.1007/s10601-013-9148-0
3. Freed, M., Carbonell, J., Gordon, G., Hayes, J., Myers, B., Siewiorek, D., Smith, S., Steinfeld, A., Tomasic, A.: Radar: A personal assistant that learns to reduce email overload. In: AAAI (2008)
4. Grosz, B.J., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence 86(2), 269–357 (1996)
5. Grosz, B.J., Sidner, C.L.: Plans for discourse. In: Cohen, P.R., Morgan, J., Pollack, M.E. (eds.) Intentions in Communication, pp. 417–444. MIT Press, Cambridge, MA (1990)
6. Leaute, T., Williams, B.C.: Coordinating agile systems through the model-based execution of temporal plans. In: ICAPS. pp. 22–28 (2005)
7. Liu, J., Cyphers, S., Pasupat, P., McGraw, I., Glass, J.: A conversational movie search system based on conditional random fields. In: INTERSPEECH (2012)
8. Lochbaum, K.E.: A collaborative planning model of intentional structure. Computational Linguistics 24, 525–572 (1998)
9. Myers, K., Berry, P., Blythe, J., Conley, K., Gervasio, M., McGuinness, D.L., Morley, D., Pfeffer, A., Pollack, M., Tambe, M.: An intelligent personal assistant for task and time management. AI Magazine 28(2), 47 (2007)
10. Noessner, J., Martin, D., Yeh, P., Patel-Schneider, P.: Cogmap: A cognitive support approach to property and instance alignment. In: ISWC (2015)
11. Ortiz, C., Shen, J.: Dynamic intention structures for dialogue processing. In: Proceedings of the 18th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2014) (2014)
12. Rich, C.: Building task-based user interfaces with ansi/cea-2018. Computer 42(8), 20–27 (Aug 2009)
13. Rich, C., Sidner, C.L.: Using collaborative discourse theory to partially automate dialogue tree authoring. In: Intelligent Virtual Agents. pp. 327–340. Springer (2012)
14. Sidner, C.: An artificial discourse language for collaborative negotiation. In: In Proceedings of the Twelfth National Conference on Artificial Intelligence. pp. 814–819. MIT Press (1994)
15. Vansteenwegen, P., Souffriau, W., Berghe, G.V., Oudheusden, D.V.: The city trip planner: An expert system for tourists. Expert Systems with Applications 38, 6540–6546 (2011)
16. Vidal, T., Fargier, H.: Handling contingency in temporal constraint networks: from consistency to controllabilities. Journal of Experimental and Theoretical Artificial Intelligence 11, 23–45 (1999)
17. Yeh, P., Ramachandran, D., Douglas, B., Ratnaparkhi, A., Jarrold, W., Provine, R., Patel-Schneider, P., Laverty, S., Tikku, N., Brown, S., Mendel, J., Emfield, A.: An end-to-end conversational second screen application for tv program discovery. AI Magazine 36(3) (2015)
18. Yorke-Smith, N., Saadati, S., Myers, K., Morley, D.: The design of a proactive personal agent for task management. IJAIT 21(1) (2012)
19. Yu, P., Fang, C., Williams, B.: Resolving uncontrollable conditional temporal problems using continuous relaxations. In: Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14) (2014)