Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Real-Time Collaborative Planning with the Crowd

Walter S. Lasecki, Jeffrey P. Bigham, James F. Allen, and **George Ferguson**
University of Rochester Department of Computer Science
160 Trustee Rd, Rochester, NY. 14627
{wlasecki, jpbigham, james, ferguson}@cs.rochester.edu

Abstract

Planning is vital to a wide range of domains, including robotics, military strategy, logistics, itinerary generation and more, that both humans and computers find difficult. Collaborative planning holds the promise of greatly improving performance on these tasks by leveraging the strengths of both humans and automated planners. However, this requires formalizing the problem domain and input, which must be done by hand, *a priori*, restricting its use in general real-world domains. We propose using a real-time crowd of workers to simultaneously solve the planning problem, formalize the domain, and train an automated system. As plans are developed, the system is able to learn the domain, and contribute larger segments of work.

Introduction

Planning is a critical part of many problems, often involving highly complex tasks that even expert users are not able to handle, domain expertise that arbitrary workers do not posses, and problems too large for automated planners. Mixed-initiative collaborative planning leverages the strengths of both humans and machines. Humans are able to use their domain expertise to perform high level planning and propose partial solutions, while automated planners can fill in details, compute consequences, and answer questions, such as if and how the high-level plan is feasible.

Previous collaborative planners relied on users to specify the problem. However, asking the user to do this task detracts from the benefit of the system itself. Instead, we propose using *the crowd*, a dynamic group of workers of varying quality and skill sets, to assist the user in formalizing and solving problems. Crowd workers can be recruited from micro-task marketplaces such as Mechanical Turk, other groups of expert users, or a combination of the two. Using the crowd as both a collaborator and interface to the automated system leverages the inherent parallelism of the crowd to make collaboration between the user and computer more fluid and enable larger problems to be solved. Additionally, using real-time crowds in concert with automated planners enables solutions to be generated quicker than a single user could, not just with less effort.

We ground or discussion of this system using a trip planning domain due to its use in previous work on crowd planning. Our goal is for a user to plan a two-week vacation for a family of four from Rochester to Hawaii. The trip will take place in July, and has an initially unspecified cost.

Previous work has focused on the crowd's ability to decompose tasks (Zhang, Horvitz, and Miller 2011; Kulkarni, Can, and Hartmann 2011), generate plans (Law and Zhang 2011), and satisfy constraints (Zhang et al. 2012). Our method uses the crowd to enable automated systems to go beyond this and also contribute work to the task.

Mixed-initiative planning has explored users working together with an automated planner. Users propose partial plans, then a planner solves smaller problems to find a complete solution (Ferguson and Allen 2007). In order to work *with* a user, our system must be able to respond in real-time to user actions and changes in the current state of the problem. Legion (Lasecki et al. 2011) enabled continuous real-time control of existing interfaces by combining input from workers into a single stream. Further work has demonstrated *organizational learning* within the crowds used by Legion (Lasecki et al. 2012). This means that workers can implicitly pass knowledge from one generation to the next via *demonstration points*, events where knowledgeable workers teach new workers by example. We make this idea explicit by letting workers decompose tasks and leave messages.

Collaborating with the Crowd

To begin, users provide a high level description of the task they want to complete, which is then forwarded to workers. Workers can either select a portion of the task to complete, decompose an existing task into subtasks, define constraints on the problem, or add a known fact or goal. To define a constraint, the user enters a name, an existing or new type and a limiting value, then proposes it to other workers, who can approve or reject the change collectively. The same general approach is used for all proposal types. Requiring agreement prevents multiple conflicting approaches to the solution from being interleaved. In our example, workers first decompose the task of planning the trip into four decomposable subtasks: 'find transportation', 'find lodging', 'schedule activities', and 'find restaurants'. Workers also define actionable tasks such as 'find a dog sitter' and 'find a house sitter'. When subtasks are defined, they are posted to the *pending tasks* list. The pending tasks list shows workers a tree of

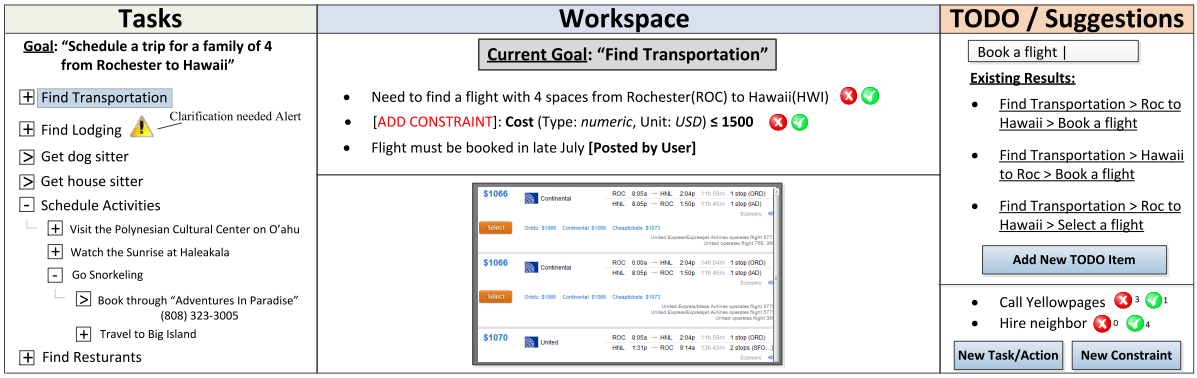

Figure 1: A prototype of the planning interface used by both workers and users. A document showing flight schedules is being shared amongst workers planning the 'find transportation' subtask in the main workspace.

all tasks and subtasks currently defined. Workers can move freely between tasks posted to this list, and the list can be reorganized by agreeing to move a portion of the tree to another branch. Allowing workers to participate in any part of this tree gives them the ability to work from both the top up and bottom down simultaneously, and lets workers self-select groups for tasks based on their own skill sets.

Both qualitative and quantitative constraints can be defined for a task, then propagated to others by being proposed by the system. These constraints can be merged, removed, or linked to other constraints via an equation. When planning our trip, the user may define a maximum cost of the trip, then workers infer compatible costs for individual parts. The overall constraint can then be defined as the sum of the costs of the subtasks, allowing the system to check satisfiability. The system also helps find matching tasks based on semantic similarity in either name or description to help prevent repeated elements from being added proposed. Figure 1 shows a worker adding 'Book a flight', a task the system recognizes as being redundant. If the worker incorrectly chooses to create a new task, the crowd can merge the two later.

The system will use a collaborative planner that is able to accept input from both users and workers during the planning process. Users are able to perform the same actions as workers, but do not need consensus to post or remove tasks, and cannot have their input voted down by workers (although they can tag elements for possible revision). The crowd can post queries that must be answered to proceed. Many of these may be answerable by other workers, but for others, the user will be required to provide an answer. These posts allow workers to obtain information that may have been neglected in the initial high-level problem description.

Workers submit responses using a structured language format to enable them to formalize both the domain and proposed solutions. While defining this language is ongoing work, in this paper, we assume it is a structured form of natural language based on worker-generated keywords. As natural language understanding systems improve, this structure can be relaxed in order to be more usable.

The system can assist workers with formalization by extracting additional information from their input. For example, `(IN family Rochester)` being `true`, implies `family` and `Rochester` exist, and that the predicate `IN`, takes arguments of types `family.type` and `Rochester.type`. The system then proposes these facts, and workers can update or merge them with existing knowledge bases. By formalizing the domain and partial solutions, the system is also able to propose tasks and decompositions that it has seen previously. This can aid users in identifying aspects of the problem, helping solutions emerge quicker.

Conclusions and Future Work

We are currently implementing the system described here as a Java server application, with a JavaScript based web front-end that both users and workers can use to collaborate.

We have presented the framework for a collaborative planning system that leverages a crowd of workers with various skill sets and levels of expertise to both solve problems and formalize the domain so that automated systems can take a more active role. We believe this synthesis of human and machine computation is the key to deploying collaborative planning systems for real-world use in the near future.

References

Ferguson, G., and Allen, J. 2007. Mixed-initiative dialogue systems for collaborative problem-solving. In *AAAI Magazine 28(2)*, 23–32.

Kulkarni, A. P.; Can, M.; and Hartmann, B. 2011. Turkomatic: automatic recursive task and workflow design for mechanical turk. In *Proc. of CHI 2011*, 2053–2058.

Lasecki, W. S.; Murray, K. I.; White, S.; Miller, R. C.; and Bigham, J. P. 2011. Real-time crowd control of existing interfaces. In *Proc. of UIST 2011*, 23–32.

Lasecki, W. S.; White, S.; Murray, K. I.; and Bigham, J. P. 2012. Crowd memory: Learning in the collective. In *Proc. of Collective Intelligence 2012*, To Appear.

Law, E., and Zhang, H. 2011. Towards large-scale collaborative planning: Answering high-level search queries using human computation. In *Proc. of AAAI 2011*.

Zhang, H.; Law, E.; Miller, R. C.; Gajos, K. Z.; Parkes, D. C.; and Horvitz, E. 2012. Human computation tasks with global constraints. In *Proc. of CHI 2012*. To appear.

Zhang, H.; Horvitz, E.; and Miller, R. C. 2011. Crowdsourcing general computation. In *Proc. of CHI 2011*.