AI-MIX: Using Automated Planning to Steer Human Workers Towards Better Crowdsourced Plans

Lydia Manikonda Tathagata Chakraborti Sushovan De
Kartik Talamadupula Subbarao Kambhampati
Department of Computer Science and Engineering
Arizona State University, Tempe AZ 85287 USA
{lmanikon, tchakra2, sushovan, krt, rao} @ asu.edu

Abstract

One subclass of human computation applications are those directed at tasks that involve planning (e.g. tour planning) and scheduling (e.g. conference scheduling). Interestingly, work on these systems shows that even primitive forms of automated oversight on the human contributors helps in significantly improving the effectiveness of the humans/crowd. In this paper, we argue that the automated oversight used in these systems can be viewed as a primitive automated planner, and that there are several opportunities for more sophisticated automated planning in effectively steering the crowd. Straightforward adaptation of current planning technology is however hampered by the mismatch between the capabilities of human workers and automated planners. We identify and partially address two important challenges that need to be overcome before such adaptation of planning technology can occur: (i) *interpreting* inputs of the human workers (and the requester) and (ii) *steering* or critiquing plans produced by the human workers, armed only with *incomplete* domain and preference models. To these ends, we describe the implementation of **AI-MIX**, a tour plan generation system that uses automated checks and alerts to improve the quality of plans created by human workers; and present a preliminary evaluation of the effectiveness of steering provided by automated planning.

1 Introduction

In solving computationally hard problems – especially those that require input from humans, or for which the complete model is not known – human computation has emerged as a powerful and inexpensive approach. One such core class of problems is *planning*. Several recent efforts have started looking at crowd-sourced planning tasks (Law and Zhang 2011; Zhang et al. 2012; 2013; Lasecki et al. 2012; Lotosh, Milo, and Novgorodov 2013). Just like in a formal organization, the quality of the resulting plan depends on effective leadership. We observe that in most of these existing systems, the workers are steered by primitive automated components that merely enforce checks and ensure satisfaction of simple constraints. Encouragingly, experiments show that even such primitive automation improves plan quality, for little to no investment in terms of cost and time.

This begs the obvious question: *is it possible to improve the effectiveness of crowdsourced planning even further by using more sophisticated automated planning technologies?* It is reasonable to expect that a more sophisticated automated planner can do a much better job of steering the crowd (much as human managers "steer" their employees). Indeed, work such as (Law and Zhang 2011) and (Zhang et al. 2012) is replete with hopeful references to the automated planning literature. There exists a vibrant body of literature on automated plan generation, and automated planners have long tolerated humans in their decision cycle – be it mixed initiative planning (Ferguson, Allen, and Miller 1996) or planning for teaming (Talamadupula et al. 2010). The context of crowdsourced planning scenarios, however, introduces a *reversed mixed initiative planning* problem – the planner must act as a guide to the humans, who are doing the actual planning. The humans in question can be either experts who have a stake in the plan that is eventually created, or crowd workers demonstrating collective intelligence.

In this paper, we present **AI-MIX** (Automated Improvement of Mixed Initiative eXperiences), a new system that implements a general architecture for human computation systems aimed at planning and scheduling tasks. **AI-MIX** foregrounds the types of roles an automated planner can play in such systems, and the challenges involved in facilitating those roles. The most critical challenges include:

Interpretation: Understanding the requester's goals as well as the crowd's plans from semi-structured or unstructured natural language input.

Steering with Incompleteness: Guiding the collaborative plan generation process with the use of incomplete models of the scenario dynamics and preferences.

The *interpretation* challenge arises because human workers find it most convenient to exchange / refine plans expressed in a representation as close to natural language as possible, while automated planners typically operate on more structured plans and actions. The challenges in *steering* are motivated by the fact that an automated planner operating in a crowdsourced planning scenario cannot be expected to have a complete model of the domain and the preferences; if it does, then there is little need or justification for using human workers! Both these challenges are further complicated by the fact that the (implicit) models used by the human workers and the automated planner are very likely to differ in many ways, making it challenging for the planner to critique the plans being developed by the human workers.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

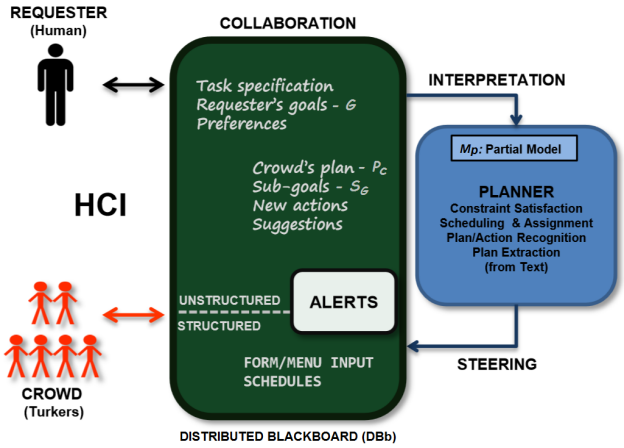

Figure 1: A generalized architecture for crowdsourced planning systems.

In the rest of the paper we describe how these challenges are handled in AI-MIX, and present an evaluation of their effectiveness in steering the crowd. The paper is organized as follows. We first look at the problem of planning *for* crowdsourced planning in more detail, and present a generalized architecture for this task. Next, we consider the roles that an automated planner can play within such an architecture, and discuss the challenges that need to be tackled in order to facilitate those roles. We then describe **`AI-MIX`** and present a preliminary evaluation of the effectiveness of the steering provided by automated planning on Amazon's MTurk platform[1]. We hope that this work will spur more directed research on the challenges that we have identified.

2 Planning for Crowdsourced Planning

The crowdsourced planning problem involves constructing a plan from a set of activities suggested by the crowd (henceforth referred to as the turkers) as a solution to a task, usually specified by a user called the *requester*. The requester provides a high-level description of the task – most often in natural language – which is then forwarded to the *turkers*. The turkers can perform various roles, including breaking down the high-level task description into more formal and achievable sub-goals (Law and Zhang 2011), or adding actions into the plan that support those sub-goals (Zhang et al. 2012). The term *planner* is used to refer to the automated component of the system, and it performs various tasks ranging from constraint checking, to optimization and scheduling, and plan recognition. The entire planning process must itself be iterative, proceeding in several rounds which serve to refine the goals, preferences and constraints further until a satisfactory plan is found. A general architecture for solving this crowdsourced planning problem is depicted in Figure 1.

2.1 Roles of the planner

The planning module, or the automated component of the system, can provide varying levels of support. It accepts both the sub-goals S_G, and crowd's plan P_C, as input from the turkers. This module analyzes the current plan generated by the crowd, as well as the sub-goals, and determines constraint and precondition violations according to the model M_P of the task that it has. The planner's job is to steer the crowd towards more effective plan generation.

However, the three main actors – turkers, requester, and planner – need a common space in which to interact and exchange information. This is achieved through a common interactive space – the *Distributed Blackboard* (DBb) – as shown in Figure 1. The DBb acts as a collaborative space where information related to the task as well as the plan that is currently being generated is stored, and exchanged between the various system components.

In contrast to the turkers, the planner cannot hope for very complex, task-specific models, mostly due to the difficulty of creating such models. Instead, a planner's strong-suit is to automate and speed-up the checking of plans against whatever knowledge it *does* have. With regard to this, the planner's model M_P can be considered shallow with respect to preferences, but may range the spectrum from shallow to deep where domain physics and constraints are concerned (Zhuo, Kambhampati, and Nguyen 2012). The planning process itself continues until one of the following conditions (or a combination thereof) is satisfied:

- The crowd plan P_C reaches some satisfactory threshold and the requester's original goal G is fulfilled by it; this is a subjective measure and is usually determined with the intervention of the requester.
- There are no more outstanding alerts, and all the sub-goals in S_G are supported by one (or more) actions in P_C.

3 Planning Challenges

From the architecture described in Figure 1, it is fairly obvious that a planner (automated system) would interact with the rest of the system to perform one of two tasks: (1) **interpretation** and (2) **steering**.

Interpretation is required for the planner to inform itself about what the crowd *is* doing; steering is required for the planner to tell the crowd what they *should* be doing.

3.1 Interpretation of the Crowd's Evolving Plan

The planner must interpret the information that comes from the requester, and from the crowd, in order to act on that information. There are two ways in which the planner can ensure that it is able to understand that information:

Force Structure The system can enforce a pre-determined structure on the input from both the requester, and the crowd. This can by itself be seen as part of the model M_P, since the planner has a clear idea about what kind of information can be expected through what channels. The obvious disadvantage is that this reduces flexibility for the turkers. In the tour planning scenario (our main application domain that we explain in Section. 4), for example, we might force the requester to number his/her goals, and force the turkers to explicitly state which goals their proposed plan aims to handle (c.f. (Zhang et al. 2012)). The turkers could also be required to add other structured attributes to their plans such as the duration and cost of various activities (actions).

[1] Amazon Mechanical Turk, http://www.mturk.com

Extract Structure The planner can also *extract* structure from the turker inputs to look for specific action descriptions that are part of the planner's model M_P, in order to understand what aims a specific plan is looking to achieve. Although this problem has connections to plan recognition (Ramírez and Geffner 2010), it is significantly harder as it needs to recognize plans not from actions, but rather textual descriptions. Thus it can involve first recognizing actions and their ordering from text, and then recognizing plans in terms of those actions. Unlike traditional plan recognition that starts from observed plan traces in terms of actions or actions and states, the interpretation involves first extracting the plan traces. Such recognition is further complicated by the impedance mismatch between the (implicit) planning models used by the human workers, and the model available to the planner.

Our system uses both the techniques described above to gather relevant information from the requester and the turkers. The requester provides structured input that lists their constraints as well as goals (and optionally cost and duration constraints), and can also provide a free unstructured text description for the task. The turkers in turn also provide semi-structured data - they are given fields for activity title, description, cost and duration. The turkers can also enter free text descriptions of their suggestions; the system can then automatically extract relevant actions by using Natural Language Processing (NLP) methods to match the input against the planner's model M_P.

3.2 Steering the Crowd's Plan

There are two main kinds of feedback an automated planner can provide to the human workers:

Constraint Checking One of the simplest ways of generating helpful suggestions for the crowd is to check for quantitative constraints imposed by the requester that are violated in the suggested activities. In terms of the tour planning scenario, this includes: (i) cost of a particular activity; and (ii) the approximate duration of an activity. If the requester provides any such preferences, our system is able to check if they are satisfied by the crowd's inputs.

Constructive Critiques Once the planner has some knowledge about the plan that the turkers are trying to propose (using the extraction and recognition methods described above), it can also try to actively help the creation and refinement of that plan by offering suggestions as part of the alerts. These suggestions can vary depending on the depth of the planner's model. Some examples include: (i) simple notifications of constraint violations, as outlined previously; (ii) plan critiques (such as suggestions on the order of actions in the plan and even what actions must be present); (iii) new plans or plan fragments because they satisfy the requester's stated preferences or constraints better; (iv) new ways of decomposing the current plan (Nau et al. 2003); and (v) new ways of decomposing the set of goals S_G.

4 System Description

The following section describes in detail the **AI-MIX** system that was deployed on Amazon's MTurk platform to engage the turkers in a tour planning task. The system is similar to Mobi (Zhang et al. 2012) in terms of the types of inputs it can handle and the constraint and quantity checks that it can provide (we discuss this further in Section 5.1). However, instead of using structured input, which severely restricts the turkers and limits the scope of their contributions, our system is able to parse natural language from user inputs and reference it against relevant actions in a domain model. This enables more meaningful feedback and helps provide a more comprehensive tour description.

4.1 Requester Input

The task description, as shown in Figure 2, is provided by the requester in the form of a brief description of their preferences, followed by a list of activities they want to execute as part of the tour, each accompanied by a suitable hashtag. For example, the requester might include one dinner activity and associate it with the tag `#dinner`. These tags are used internally by the system to map turker suggestions to specific tasks. The upper half of Figure 2 shows an example of a requester task, which includes a block of text for the turkers to extract context from, and structured task requests associated with hashtags.

4.2 Interface for Turkers

In addition to the task description, the **AI-MIX** interface also contains a section that lists instructions for successfully submitting a Human Intelligence Task (HIT) on Amazon MTurk. HIT is the individual task that the turkers work on, in this context consisting of either adding an action or a critique, as discussed in more detail later. The remaining components, arranged by their labels in the figure, are:

1. **Requester Specification**: This is the list of requests and to-do items that are yet to be satisfied. All the unsatisfied constituents of this box are initially colored red. When a tag receives the required number of supporting activities, it turns from red to green. Tags that originated from the requester are classified as top-level tags, and are always visible. Tags that are added by the automated planner or by turkers are classified as lower priority, and disappear once they are satisfied by a supporting activity.
2. **Turker Inputs**: Turkers can choose to input one of two kinds of suggestions: (i) a new action to satisfy an existing to-do item; or (ii) a critique of an existing plan activity.
3. **Turker Responses**: The "Existing Activities" box displays a full list of the current activities that are part of the plan. New turkers may look at the contents of this box in order to establish the current state of the plan. This component corresponds to the *Distributed Blackboard* mentioned in Section 2.1.
4. **Planner Critiques**: The to-do items include automated critiques of the current plan that are produced by the planner. In the example shown, "broadwayshow_showing" is a planner generated to-do item that is added in order to improve the quality of the turkers' plan.

Finally, the right hand portion of the interface consists of a map, which can be used by turkers to find nearby points of interest, infer routes of travel or the feasibility of existing

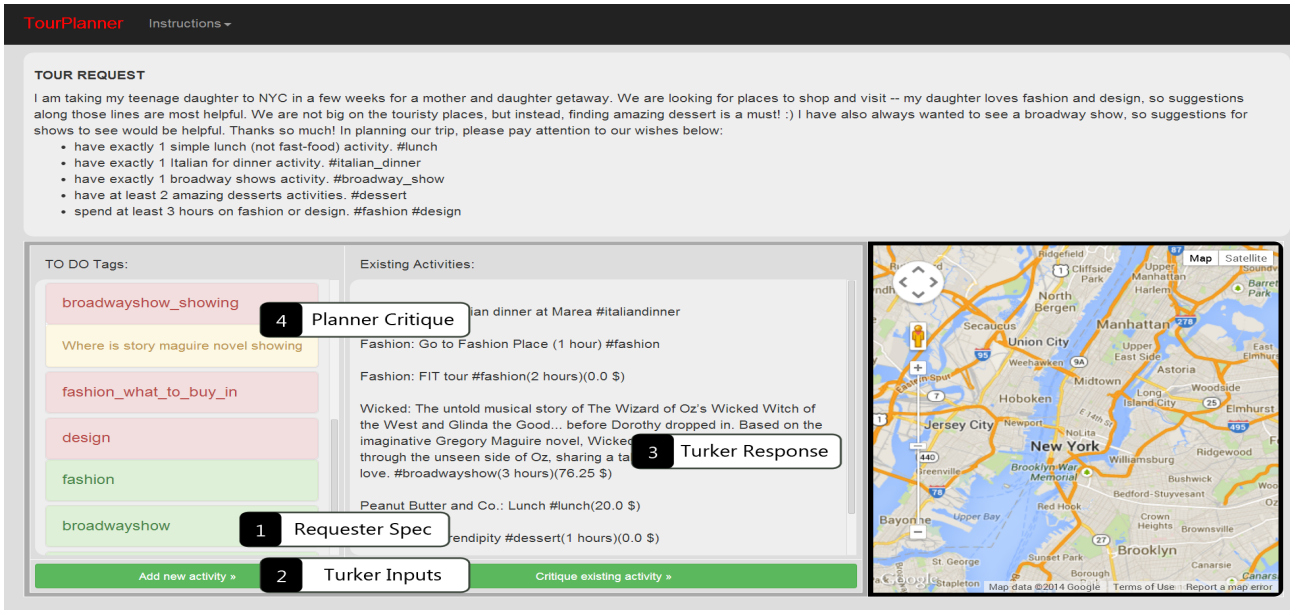

Figure 2: The **AI-MIX** interface showing the distributed blackboard through which the crowd interacts with the system.

suggestions, or even discover new activities that may satisfy some outstanding tags.

Activity Adddition The "Add Activity" form is shown in Figure 3. Turkers may choose to add as many new activities as they like. Each new activity is associated with one of the to-do tags. After each activity is submitted, a quantitative analysis is performed where the activity is (i) checked for possible constraint (duration or cost) violations; or (ii) critiqued by the planner.

Action Extraction In order to extract meaning from the new activities described by the turkers, the system performs parts of speech (PoS) tagging on the input text using the *Stanford Log-Linear Part-of-Speech* tagger (Toutanova et al. 2003). It identifies the name of the suggested activity and places that the turkers are referring to using the verb and noun parts of the tagger's output respectively.

Sub-Goal Generation **AI-MIX** uses the same tags used by turkers while inputting activities in order to determine whether the planner has additional subgoal annotations on that activity. To facilitate this, the planner uses a primitive PDDL (McDermott et al. 1998) domain description of general activities that may be used in a tour-planning applications – this description corresponds to the *planner model* M_P introduced earlier. Examples of actions in M_P include high level activities such as `visit`, `lunch`, `shop` etc. Each activity is associated with a list of synonyms, which helps the planner in identifying similar activities. Currently, we generate these synonyms manually, but it is possible to automate this via the use of resources such as WordNet. Each action also comes with some generic preconditions. When the planner determines that a turker generated activity matches one of the actions from its model, it generates subgoals to be added as to-do items back in the interface based on the preconditions of that action. An example of an action description (for the "visit" action) is given below:

```
(:action visit ;; synonyms: goto, explore
 :parameters (?p - place)
 :precondition (at ?p) ;; Getting to ?p,
 ;; Entrance fee ?p, ;; Visiting hours ?p
 :effect (visited ?p))
```

In the example given above, the planner would pop up the three preconditions – `Getting to`, `Entrance fee`, and `Visiting hours` – as to-do sub-goals for any `visit` actions suggested by turkers. The system also provides some helpful text on what is expected as a resolution to that to-do item – this is indicated by the yellow "planner critique" box in Figure 2.

Constraint Checking In addition to generating sub-goals for existing activities, our system also automatically checks if constraints on duration and cost that are given by the requester are being met by the crowd's plan. If these constraints are violated, then the violation is automatically added to the to-do stream of the interface, along with a description of the constraint that was violated. Turkers can then choose to add an action that resolves this to-do item using the normal procedure.

Adding Turker Critiques The turkers can also add *critiques* of the actions in the existing plan. To do this, they use the form shown in the lower half of Figure 3. The turkers click on an existing activity, and enter the note or critique in a text box provided. Additionally, they are also asked to enter a child tag, which will be used to keep track of whether an action has been added to the plan that resolves this issue. Turkers can add as many critiques as they want.

Though the current system uses only a preliminary form of automated reasoning, this effort can be seen as the first

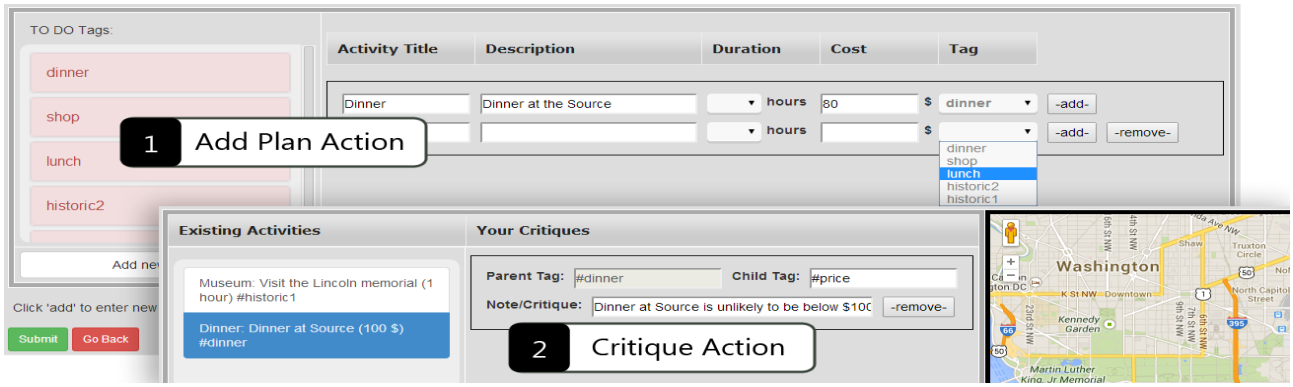

Figure 3: Adding and critiquing activities (plan actions) in the **AI-MIX** system.

step towards incorporating more sophisticated methods for plan recognition and generation (Talamadupula and Kambhampati 2013). A video run-through of our system can be found at the following URL: http://youtu.be/73g3yHClx90.

5 Experiments

5.1 Experimental Setup

For our study, HITs were made available only to the turkers within US (since the requests involved locations inside the US) with a HIT approval rate greater than 50%. Turkers were paid 20 cents for each HIT, and each turker could submit 10 HITs per task. We used tour planning scenarios for six major US cities, reused from the Mobi system's evaluation (Zhang et al. 2012). To measure the impact of automated critiquing on the generated plans, we compared the results from three experimental conditions:

C1: Turkers could give suggestions in free text after reading the task description - there were no automated critiques.

C2: Turkers quantified their suggestions in terms of cost and duration, and the system checked these constraints for violations with respect to the requester demands.

C3: In addition to C2, the system processed free-form text from turker input, and extracted actions to match with our planning model in order to generate alerts for sub-goals and missing preconditions.

C1 and C2 were compared to the proposed approach, C3, separately. Each set was uploaded at the same time, with the same task description and HIT parameters. In the first run, C3 and C2 were compared on 6 scenarios (New York, Chicago, San Francisco, Las Vegas, Washington and Los Angeles) and were given 2 days before the HITs were expired. The interfaces for both C3 and C2 were made identical to eliminate any bias. In the second run, the conditions C1 and C3 were run over a period of one day, for the two scenarios which were most popular in the first run (New York and Chicago). For each of these tasks, the requester prepopulated the existing activities with one or two dummy inputs that reflect the kinds of suggestions she was looking for. In sum, we had more than 150 turkers who responded to our HITs. The analysis that follows is from the 35 turkers who contributed to the final comparisons among C1, C2, and C3.

5.2 Task Completion Latency

When C3 was compared to C1 over a period of one day, we found that C3 received four responses from 3 distinct turkers, whereas C1 failed to attract any responses. This might indicate that the presence of the "TO DO" tags generated by the automated critiquing component was helpful in engaging the turkers and guiding them towards achieving specific goals. However, there may also be alternate explanations for the fact that C1 did not receive any inputs, such as turker fatigue, or familiarity with the C3 interface from previous runs. There is need for further experimentation before these results can be conclusively proved.

We also looked at the number of HITs taken to complete the tasks for each of the scenarios. After the HITs were expired, none of the tasks were entirely complete (a task is "completed" when there are no more outstanding to-do items), but C2 had 3.83 unfulfilled tags per HIT as compared to 10.5 for C3. As expected, the task completion latency seems to have increased for C3, since alerts from the system drive up the number of responses required before all the constraints are satisfied. As shown in the following paragraph, however, the increased quality of generated plans may justify this additional latency.

5.3 Generated Tour Plan Quality

We see that the quality of the plans, in terms of detail and description, seems to increase in C3, since we now have users responding to planner critiques to further qualify suggested activities. For example, a turker suggested "not really fun, long lines and can not even go in and browse around" in response to a planner generated tag (related to a "fun club" activity suggested previously), while another suggested a "steamer" in response to a planner alert about "what to eat for lunch". A comparison between the plans generated for C2 and C3 (for New York City) is given in Table 1. This seems to indicate that including a domain description in addition to the simplistic quantity and constraint checks increases the plan quality.

5.4 Role Played by the Planner Module

We now look at some statistics that indicate the role played by the automated module in the tasks. We received a to-

Show: Go to TKTS half ticket discount booth. You have to stand in line early but it's an authentic nyc experience #show(3 hours)(200.0 $)
Show: Go to show #show(3 hours)(200.0 $)
Show: ABSOLUTELY CANNOT go wrong with Phantom of the Opera #show(3 hours)(200.0 $)
Lunch: Alice's Tea Cup #lunch(20.0 $)
Design: Walk around the Garment District (go into shops) just south of Times Square. They often print their own fabrics. #design(2 hours)(0.0 $)
Dessert: Serendipity #dessert(1 hours)(10.0 $)
piccolo angolo: Italian in the Village - real deal #italiandinner(2 hours)(60.0 $)
Lombardi's Pizza: #italian_dinner #italiandinner_todo1
Ice Cream: http://www.chinatownicecreamfactory.com/ #italiandinner_todo0
#lunch: Mangia Organics #lunch_todo0
watch Wicked (musical): Do watch Wicked the musical. It's a fantastic show and one of the most popular on Broadway right now! #broadwayshow(3 hours)(150.0 $)
watch How to Succeed in Business: Also a great show, a little less grand than Wicked. #broadwayshow(3 hours)(150.0 $)
Activity Steamer: #lunch #lunch_todo1
Paradis To-Go: Turkey & Gruyere is pretty delicious. The menu is simple, affordable, but certainly worth the time #lunch(1 hours)(10.0 $)
cupcakes!: Magnolia Bakery on Bleecker in the Village #dessert(1 hours)(10.0 $)

Table 1: Sample activity suggestions from turkers for the two conditions: C2 (top) and C3 (bottom).

tal of 31 new activity suggestions from turkers, of which 5 violated quantity constraints. The C3 interface attracted 39 responses, compared to 28 for C2, which may indicate that the planner tags encouraged turker participation.

Note that in the **AI-MIX** interface, there is no perceptual difference between the critiques generated by the planner and the critiques suggested by humans. With this in mind, there were 8 flaws pointed out by humans, but none were acted upon by other turkers; the planner on the other hand generated 45 critiques, and 7 were acted upon and fixed by turkers. This seems to indicate that turkers consider the planner's critiques more instrumental to the generation of a high quality plan than those suggested by other turkers. Though these results are not entirely conclusive, and might also be attributed to possibilities like the critiques of the planner being more popular because they might have been easier to solve; there is enough evidence to suggest that the presence of an automated system does help to engage and guide the focus of the crowd.

6 Conclusion

In this paper, we presented a system, **AI-MIX**, that is a first step towards using an automated planner in a crowdsourced planning application. We identified two major challenges in achieving this goal: *interpretation* and *steering*. We then described the framework of **AI-MIX**, and showed how these challenges were handled by our system – using forced structure and structure extraction for interpreting actions; and using constraint checking and automated planner critiques for steering. We also presented preliminary empirical results over the tour planning domain, and showed that using an automated planner results in the generation of better quality plans. Interestingly, it is possible to improve the completeness of the domain model of a planner over time (Yang, Wu, and Jiang 2007). We are continuing to run experiments using more scenarios and larger time scales to provide further validation for our hypotheses. We are also looking at the problem of eliciting information (Kaplan et al. 2013) from the crowd in order to go from the current list of activities suggested by the crowd, to a more structured *plan* in the traditional sense of the word.

Acknowledgments. This research is supported in part by the ARO grant W911NF-13-1-0023, the ONR grants N00014-13-1-0176 and N0014-13-1-0519, and a Google Research Grant.

References

Ferguson, G.; Allen, J.; and Miller, B. 1996. Trains-95: Towards a mixed-initiative planning assistant. In *Proc. of AIPS-96*, 70–77.

Kaplan, H.; Lotosh, I.; Milo, T.; and Novgorodov, S. 2013. Answering planning queries with the crowd. *In Proc. of VLDB Endowment* 6(9):697–708.

Lasecki, W. S.; Bigham, J. P.; Allen, J. F.; and Ferguson, G. 2012. Real-time collaborative planning with the crowd. In *Proc. of AAAI*.

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

Lotosh, I.; Milo, T.; and Novgorodov, S. 2013. CrowdPlanr: Planning Made Easy with Crowd. In *Proc. of ICDE*. IEEE.

McDermott, D.; Knoblock, C.; Veloso, M.; Weld, S.; and Wilkins, D. 1998. PDDL–the Planning Domain Definition Language: Version 1.2. *Yale Center for Computational Vision and Control, Tech. Rep. CVC TR-98-003/DCS TR-1165*.

Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. Shop2: An HTN planning system. *JAIR* 20:379–404.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proc. of AAAI*.

Talamadupula, K., and Kambhampati, S. 2013. Herding the crowd: Automated planning for crowdsourced planning. *CoRR* abs/1307.7720.

Talamadupula, K.; Benton, J.; Kambhampati, S.; Schermerhorn, P.; and Scheutz, M. 2010. Planning for human-robot teaming in open worlds. *TIST* 1(2):14.

Toutanova, K.; Klein, D.; Manning, C. D.; and Singer, Y. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic dependency network. In *Proc. of HLT-NAACL*.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence Journal*.

Zhang, H.; Law, E.; Miller, R.; Gajos, K.; Parkes, D.; and Horvitz, E. 2012. Human Computation Tasks with Global Constraints. In *Proc. of CHI*, 217–226.

Zhang, H.; Andre, P.; Chilton, L.; Kim, J.; Dow, S. P.; Miller, R. C.; MacKay, W.; and Beaudouin-Lafon, M. 2013. Cobi: Communitysourcing Large-Scale Conference Scheduling. In *CHI Interactivity 2013*.

Zhuo, H. H.; Kambhampati, S.; and Nguyen, T. A. 2012. Modellite case-based planning. *CoRR* abs/1207.6713.