Human Computation (2017) 4:1:25-46
© 2017, Manikonda, Chakraborti, Talamadupula & Kambhampati. CC-BY-3.0
ISSN: 2330-8001, DOI: 10.15346 /hc.v4i1.2

# Herding the Crowd: Using Automated Planning for Better Crowdsourced Planning

LYDIA MANIKONDA, ARIZONA STATE UNIVERSITY

TATHAGATA CHAKRABORTI, ARIZONA STATE UNIVERSITY

KARTIK TALAMADUPULA, ARIZONA STATE UNIVERSITY

SUBBARAO KAMBHAMPATI, ARIZONA STATE UNIVERSITY

## 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 crowdsourced planning. Straightforward adaptation of current planning technology is however hampered by the mismatch between the capabilities of human workers and automated planners. We identify and 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 plan generation system that uses automated checks and alerts to improve the quality of plans created by human workers.

## 1. INTRODUCTION

Human computation (Law and von Ahn, 2011) is emerging as one of the fastest growing fields to solve computationally hard problems because it offers a powerful and inexpensive alternative that helps combine the relative strengths of humans and computers. This field is attracting more attention from different research communities as interesting connections between them and human computation are being revealed (Dai et al., 2010, 2011; Kamar et al., 2012; Lotosh et al., 2013). In solving computationally hard problems from different fields – 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) which attempt to optimize workflows used in crowdsourcing (Kamar et al., 2013; Dai et al., 2013), learning and planning to guide the best use of people and machines in hierarchical tasks (Kamar and Horvitz, 2015), etc. Just as in a formal organization, the quality of the resulting plan depends on effective leadership. We observe that in most of these existing systems, workers are steered by primitive automated components that merely enforce checks and ensure satisfaction of simple constraints. Encouragingly, experiments show that even these primitive automated components improve plan quality, for little to no investment in terms of cost and time (Zhang et al., 2012).

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 good human managers "steer" their employees more effectively). 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 et al., 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 aim of our research is to implement and evaluate a spectrum of solutions for the most critical challenges in the **AI-MIX** system which 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 or 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 requester preferences; if it does, then there is little need or justification for using human workers as it can automatically generate the best plan that meets requester preferences and also knows which type of plan to generate. Both of 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.

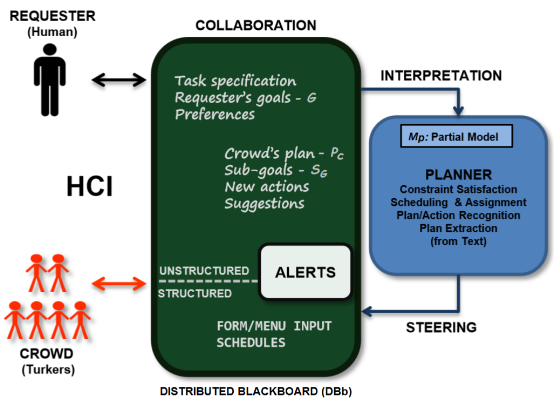


**Figure 1.** A generalized architecture for crowdsourced planning systems that includes Requester, Crowd, Distributed Blackboard (DBb) and Planner.

In this paper, we first provide a brief survey on the existing crowdsourced planning systems and how they are related to the current system proposed in this paper. We then look at 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 the **AI-MIX** system, and the types of planning support it provides to the crowd workers. We then present an empirical study aimed at understanding the impact of automated planning support on the effectiveness of crowd-sourced planning. Accordingly, our general strategy involves comparing the relative performance of different versions of **AI-MIX** with increasing levels of planning support. We present and discuss results of two sets of experiments done both on Amazon Mechanical Turk platform, and on a separate standalone platform. Our results show that increased planning support that goes beyond simple constraint checking (as used in earlier systems) does indeed result in improved performance of crowd planning. In our previous work only the suggestions provided by the crowd are analyzed. Plans generated from these suggestions in different experimental settings are included in this paper. Thus the summary of our contributions are as follows:

i. We address the interpretation challenge as follows:
   – Our system leverages established natural language processing techniques to automatically detect the type and context of actions described in the crowd suggestions.

ii. We address the steering challenge as follows:
   – The system provides planning support by automatically extracting subgoals.
   – It automatically performs constraint checking on duration and cost.
   – It lets crowd workers to critique the existing plan actions.
   – When no other sub goals can be extracted and all the constraints have been satisfied, the system generates an automated plan or schedule to reflect the crowd suggestions.

iii. Finally we deploy this system in different experimental scenarios to evaluate the strengths of this approach as compared to the state of the art approaches in crowd planning.

## 2. RELATED WORK

Research in planning and crowd-sourcing intersect paths in three different ways: (i) to use the crowd to do planning tasks (ii) to use the crowd to provide planning knowledge, to be used by an automated planner, and (iii) to use the planning technology to manage the crowd and their activities (e.g. monitor the crowd quality, allocate Human Intelligent Tasks (HITs)[1], optimize the workflow etc) . Of these, the first direction is the one most relevant to this paper–as our focus is on using automated planning technology as a "force multiplier" to help the crowd workers engaged in planning tasks. Accordingly, we start this section with a comprehensive account of related work in that direction.

A number of other implemented human computation systems that use automated technology to assist with and improve the quality of tasks other than planning are listed in a wide-ranging survey (Quinn and Bederson, 2011) of the field. Our paper focuses solely on the crowd-planning aspect, rather than the gamut of general human-computation tasks. Both the challenges of interpreting the crowd's plan and the challenge of steering it can have primitive solutions (e.g. force structure and critique the plan in terms of lower level consistency checks), and more ambitious solutions (e.g. interpret structure by extracting actions and plans from text, and evaluate the extracted plan in terms of the planning model to provide constructive extensions or alternatives for the crowd's consideration). Most existing work uses the primitive solutions for interpretation and steering. Their success argues for the exploration of more ambitious solutions to these problems. A few systems have attempted to solve some version of the crowdsourced planning problem. All of these systems can be seen as special cases of the general architecture described in the latter section of this paper. We describe these approaches that rely on automated systems to improve the synthesis of crowd-plans and the quality of those plans.

Mobi (Zhang et al., 2012) takes a planning mission that consists of both preferences and constraints as input from a requester, and generates a plan or itinerary by allowing workers in the crowd to plan in a shared manner. Constraints are limited to two types: qualitative; and quantitative that may be specified either over the amount of time to be spent on activities in each category, or on the number of such activities. They show that: (i) for the same amount of money spent on human workers, a system with automated alerts tends to come up with higher quality plans; and (ii) the automated alerts tend to spur the plan towards breaching a set plan quality threshold in far fewer steps than a system without them. Another system proposed in (Law and Zhang, 2011) – which introduces CrowdPlan, a collaborative planning algorithm, takes as input a high-level mission from the user and provides web-based resources for accomplishing that mission. To facilitate this, CrowdPlan uses human workers to decompose the high-level mission into goals (like "stop smoking", "eat healthier food"). Although the decomposition process has similarities to HTN planning (Nau et al., 2003), CrowdPlan itself doesn't have any automated planning component overseeing the human workers.

On the other hand, there are systems like CrowdPlanr (Lotosh et al., 2013) that focus more on sequencing the steps in a plan once the actions themselves have been selected. CrowdPlanr takes a given set of actions, determines the least number of questions (and what those questions are) to ask the crowd of workers to achieve a plan of acceptable quality. The *model* consists of both the constraints specified by the requester as well as the crowd's knowledge. It is quite likely that the quality of plans produced by such a system would be sensitive to the familiarity of the workers with the task at hand. The Cobi (Zhang et al., 2013) system employs the same basic idea – that

[1]Human Intelligence Task (HIT) is the individual task that the turkers work on.

the *crowd* assisting with the planning already has a built-in model of preferences and constraints. Cobi sought to “communitysource” the scheduling of a large-scale conference (CHI 2013) by taking input from organizers, as well as authors and attendees in order to come up with a schedule (plan) of good quality, that violates a fewer number of constraints while being feasible. The collection of constraints, and the grouping of papers into areas of expertise for the clustering, may be seen as Cobi’s model. The automated system thus uses this model in order to resolve as many of the constraints as possible, and come up with a conference schedule which is both satisfactory, and more importantly transparently collaborative.

The system proposed (but not yet implemented) by (Lasecki et al., 2012) is the closest in spirit and idea to applying automated planning methods on a distributed interaction platform to aid crowdsourced planning. That system allows workers to decompose tasks and interact by posting constraints and further sub-tasks to the problem of planning a trip. The automated system enables collaboration amongst the workers and the requester, and also extracts additional information from their input (restricted to a structured form) in order to update the model.

The role played by automated planning in crowdsourced planning problems has interesting connections and contrasts to the role of planners in mixed-initiative planning (Ferguson et al., 1996) and human-in-the-loop planning (Kambhampati and Talamadupula, 2015). Broadly speaking, mixed-initiative planning work involved humans entering the land of planners; while crowdsourced planning requires the planner to enter the land of humans. For example, in mixed-initiative planning, the "interpretation" problem is punted away by expecting the human in the loop to interact with the plan on the planner’s terms; this will certainly not work in crowdsourced planning. Further, in mixed-initiative and human-robot teaming scenarios, the planner is expected to have a complete model of the planning problem – which is rarely the case in crowdsourced planning. Instead, the planner must deal with a model-lite (Kambhampati, 2007) spectrum, where models may range from simple feasibility constraints, through incomplete theories of the task domain and, very rarely, preferences specified in a standardized format. Planning techniques that have so far expected input in standard forms (like PDDL) must also change to take this model-lite spectrum into account.

We will end this section with a brief discussion of the other two intersections between planning and crowdsourcing. The approach described in (Zhuo, 2015) is an example of using the crowd to acquire planning knowledge (domain operators, initial state information etc.), which can then be used by an automated planner. In (Zhuo, 2015), they acquire formal domain action descriptions through a combination of crowd-sourced knowledge, and prior planning traces. Our work, in contrast, focuses on using planning technology to complement the crowd workers engaged in planning tasks.

There are several research efforts that use planning and scheduling techniques to manage the crowd, regardless of the specific task being supported by the crowd-sourced system. The work proposed in (Dai et al., 2011) is an example of the strand of research where planning techniques are used to manage the crowd resources. The TurKontrol project, which is an end-to-end system that dynamically optimizes live crowdsourcing tasks, and deals with the problem of assigning HITs to both improving the quality of a solution, as well as checking the current quality. This work also concentrates on optimizing iterative, crowdsourced workflows by learning the model parameters (Weld et al., 2011) from real Mechanical Turk data, and modeling worker and his accuracy (for quality improvement) (Kamar and Horvitz, 2015) and voting patterns and incentives (Mao et al., 2013) (to check the quality of work done). Such systems are independent of the actual task at hand

– whether that be text improvement or human intelligence to produce plans – and focus more on worker-independent parameters to assign improvement and voting jobs instead.

## 3. PLANNING FOR CROWDSOURCED PLANNING

The crowdsourced planning problem involves constructing a plan from a set of activities suggested by the crowd 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 *crowd workers* (*or turkers* in this paper we use these terms interchangeably). 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 - it performs various tasks ranging from constraint checking, to optimization and scheduling, and action 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 a crowdsourced planning problem is shown in Figure 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. The planner's model $M_P$ can thus be considered shallow with respect to preferences, but may range the spectrum from shallow to deep where domain physics and constraints are concerned (Zhuo et al., 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; this is a subjective measure and is usually determined with intervention from the requester.
- There are no more outstanding alerts, and all the sub-goals in $S_G$ are supported by one (or more) actions in $P_C$.

## 4. PLANNING CHALLENGES

In the current system's architecture that is depicted in the Figure 1, 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. In the next section, we will describe specific components of the **AI-MIX** system which solve some of the planning challenges listed here in more detail.

## 4.1. 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 approach this problem.

### *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 a given planning scenario (we consider *tour planning* as our main application domain that we explain in Section. 5), 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) that are part of the plan.

### *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 $M_P$.

## 4.2. Steering the Crowd's Plan

The planner can steer the turkers by offering helpful suggestions, alerts, and perhaps even its own plan. There are two main kinds of feedback an automated planner can provide to the human workers:

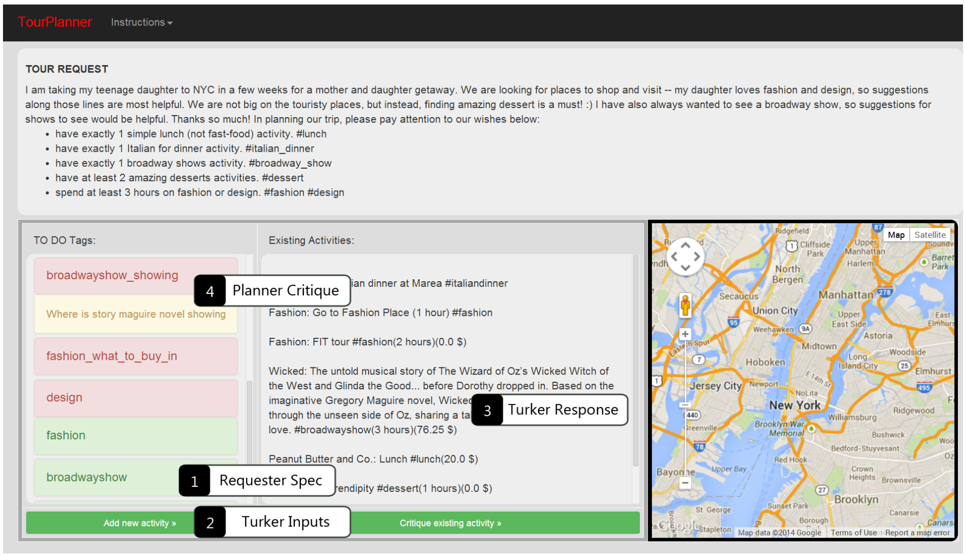
### *Constraint Checking*

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 $S_G$.

## 5. AI-MIX ARCHITECTURE


TourPlanner Instructions

TOUR REQUEST

I am taking my teenage daughter to NYC in a few weeks for a mother and daughter getaway. We are looking for places to shop and visit -- my daughter loves fashion and design, so suggestions along those lines are most helpful. We are not big on the touristy places, but instead, finding amazing dessert is a must! :) I have also always wanted to see a broadway show, so suggestions for shows to see would be helpful. Thanks so much! In planning our trip, please pay attention to our wishes below:

- have exactly 1 simple lunch (not fast-food) activity. #lunch
- have exactly 1 Italian for dinner activity. #italian_dinner
- have exactly 1 broadway shows activity. #broadway_show
- have at least 2 amazing desserts activities. #dessert
- spend at least 3 hours on fashion or design. #fashion #design

TO DO Tags:

broadwayshow_showing

Where is story maguire novel showing

fashion_what_to_buy_in

design

fashion

broadwayshow

Existing Activities:

ian dinner at Marea #italiandinner

Fashion: Go to Fashion Place (1 hour) #fashion

Fashion: FIT tour #fashion(2 hours)(0.0 $)

Wicked: The untold musical story of The Wizard of Oz's Wicked Witch of the West and Glinda the Good... before Dorothy dropped in. Based on the imaginative Gregory Maguire novel, Wicke through the unseen side of Oz, sharing a ta love. #broadwayshow(3 hours)(76.25 $)

Peanut Butter and Co.: Lunch #lunch(20.0 $)

rendipity #dessert(1 hours)(0.0 $)

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

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 6.1). 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.

## 5.1. 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.

## 5.2. 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:

i. **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.

ii. **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 (action).

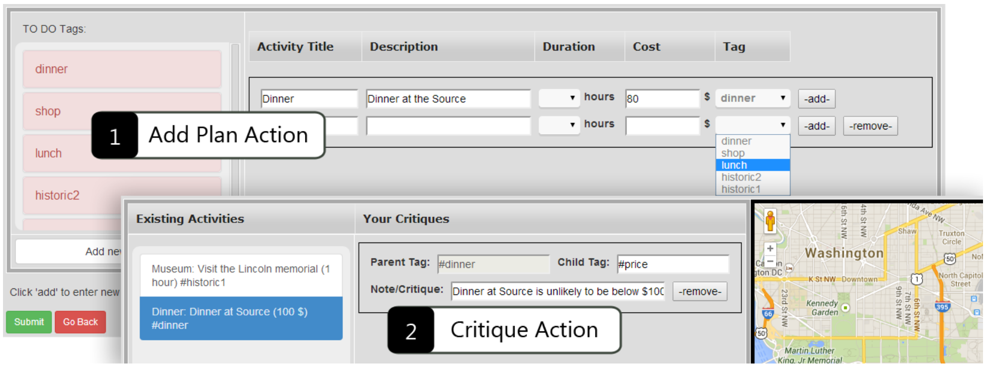
iii. **Turker Responses**: The "Existing Activities" box displays a full list of the current activities that are part of the plan. New turkers may look at the contents of this box in order to establish the current state of the plan. This component corresponds to the *Distributed Blackboard* mentioned in Section 3.

iv. **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 suggestions, or even discover new activities that may satisfy some outstanding tags. Turkers have two choices in terms of the kinds of responses (HITs) that they can provide to the system: (i) they may add a new activity in response to one of the to-do tags; or (ii) they may enter a critique or a note to point out flaws in one of the existing activities.

### *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. Turkers also have the option to propose new tags (new missing goals) for their novel suggestions. After each activity is submitted, a quantitative analysis is performed where the activity is (i) checked for possible constraint (duration or cost) violations; and (ii) critiqued the planner.


TO DO Tags: dinner, shop, lunch, historic2

| Activity Title | Description | Duration | Cost | Tag |
|---|---|---|---|---|
| Dinner | Dinner at the Source | hours | 80 $ | dinner |
| | | hours | $ | lunch |

Click 'add' to enter new

Existing Activities:
- Museum: Visit the Lincoln memorial (1 hour) #historic1
- Dinner: Dinner at Source (100 $) #dinner

Your Critiques:
Parent Tag: #dinner Child Tag: #price
Note/Critique: Dinner at Source is unlikely to be below $10C

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

## 5.3. Interpretation: Action Extraction

To facilitate the extraction of meaning from the turker generated activities, the system performs parts of speech (PoS) tagging on the activities to identify the name of the activity as well as the places that turkers are referring to; currently, we assign the *verb* and *noun* parts of the tagger’s output to these respectively. Due to its wide popularity in efficient tagging, we utilized the *Stanford Log-Linear Part-of-Speech* tagger (Toutanova et al., 2003).

## 5.4. Steering: Planning Support

In **AI-MIX**, we experimented with different levels of automated planning support for crowd workers. In addition to simple constraint checking (of the type done in systems like Mobi (Zhang et al., 2012), we use 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 previously. Examples of actions in $M_P$ include high level activities such as `visit`, `lunch`, `shop` etc. As discussed in the next two sections, the PDDL model is used to provide helpful subgoal generation suggestions, as well as to do more sophisticated aggregation of crowd suggestions into a globablly consistent plan/schedule.

### *Subgoal 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. Each activity is associated with a list of synonyms, which help the planner in identifying similar activities. Currently, we generate these synonyms manually, but it is trivial to automate this via the use of resources such as WordNet (Miller, 1995). 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 sub-goals 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 are free to add as many critiques as they want.

### *Automated Plan/Schedule Generation from Crowd Suggestions*

Finally, we compute a consolidated plan for the requester by translating the plan suggestions provided by the crowd into the semantics of logic programming. We compile the suggestions into an Answer Set Program (ASP), which acts as a declarative planner that identifies the answer sets or stable models (or plans) that satisfy the preferences and constraints imposed by both the crowd and the knowledge base. The workflow of our system shown in Figure 4 shows how the requester provides a set of incomplete preferences, the crowd responds with activities or additional constraints, all of which are compiled into an existing knowledge base which is then used to produce a logic program to compute satisfying plans as answer sets when all the constraints has been satisfied.

The Tour Planning Domain is initialized with some basic knowledge base to start with and the assumptions on some generic actions in the domain. For example, each problem is instantiated with primitive actions like breakfast, lunch and dinner. These are then associated with time ranges when these activities usually take place as given by rules (2) and (3). More actions involved specifically with tourism (like a visit museum action and museum timings) can be added similarly.

```
action(lunch;dinner;breakfast).                                    (1)
← activity(dinner,T,X) & T<18.                                      (2)
← activity(dinner,T,X) & T>22.                                      (3)
activity(dinner,T1,1):time(T1).                                    (4)
```

These actions are first initialized with choice rules such that any action can happen in any time. Then each action is bound with the following (fairly intuitive) domain independent rules:

– *no concurrence*: no two actions can occur at the same time, given by rule (5).
– *existence*: it is maintained that at least one instance of each action requested for is there in the final schedule, given by rule (6).
– *uniqueness*: one action occurs at one unique place over all the time points it is true - rule (7).
– *contiguity*: the same action must be true for contiguous periods of time, given by rule (8).

```
?[T,X]: activity(breakfast,T,X).                                   (5)
← activity(A,T,X) & activity(AA,T,XX) & A!=AA.                      (6)
← activity(A,T,X) & activity(A,TT,XX) & X!=XX.                      (7)
activity(museum,TTT,X) ←activity(museum,T,X) &
        activity(museum,TT,X) & TTT>T & TTT<TT.                     (8)
```

These rules are added to the program every time the crowd enters a new suggestion. The crowd can impose further rules for their suggestions, like time (9) or durations (10) or even orderings (11) which are all compiled into logic rules as shown. Specifically (9) says that the coffee activity, if it exists, should occur at 10 O'clock, while (10) specifies that the museum visit should last one hour, and (11) stipulates an ordering that the coffee activity should happen before 10 O'clock.

```
activity(coffee,10,1) ← ?[T]: activity(coffee,T,1).                 (9)
1{activity(museum,T1,1):time(T1)}1 ←
          ?[T]: activity(museum,T,1).                               (10)
← activity(coffee,T,1) & T<10.                                      (11)
```

In our system, the types of actions that already exist in the knowledge base, or are subsequently added by the crowd, are identified by the unique hashtags used in the plan recommendations. The forms through which the crowd interacts has both structured and unstructured parts to harness the best abilities of both the crowd and the automated component. The unstructured free form text input gives free description that is available back to the crowd to reason with, while the structured fields are used to build the constraints in clingo. Specifically the activity tag is used as the domain predicate (the crowd may suggest entirely new activities out-side the specifications, which go on to enrich the domain model), and the location is added to the points of interest associated with this action. The rest of the fields (time, duration, cost) are optional, which when entered adds the appropriate rules in the asp compilation. Notice how most of these constraints are not binding (in the sense one rule can overwrite another). This is necessary due to the iterative nature of the crowd-planning process, and thus most of the fields are optional, or these constraints themselves have precedences. This is the reason for the somewhat unconventional styling of some of the rules, as opposed to the case if we were solving a one-time specified problem. Cost does not contribute to the schedule and is hence ignored in setting up the constraints.

We used *f2lp* [2] as a grounder and solver for this logic program to compute stable models or answer sets when the program is satisfiable. These answer sets are the final plans that are obtained given all the preferences and constraints. Currently we accept any answer set as a plan but frameworks like *asprin* (Brewka et al., 2015) may be utilized for a quantitative and qualitative optimization for setting preferences over stable models generated by the ASP solver.

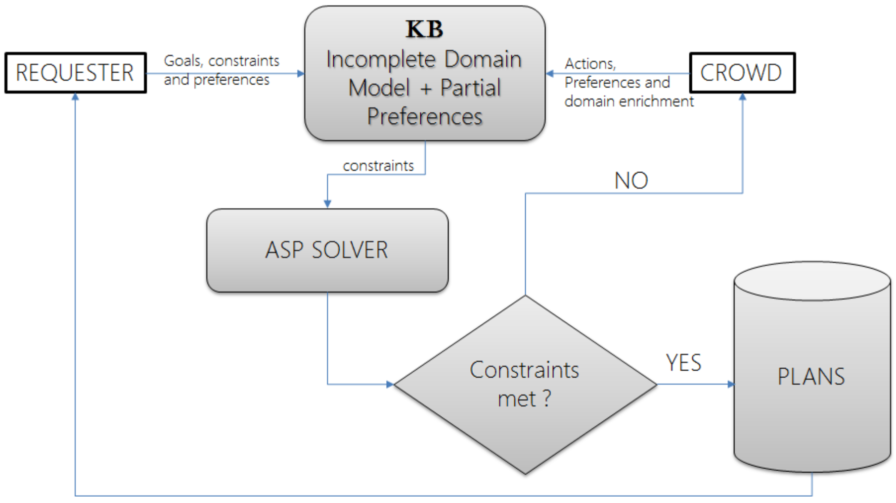

KB
Incomplete Domain Model + Partial Preferences
REQUESTER
Goals, constraints and preferences
Actions, Preferences and domain enrichment
CROWD
constraints
NO
ASP SOLVER
Constraints met ?
YES
PLANS

**Figure 4.** Workflow of our system

The workflow in our system is shown in Figure 4.

## 6. EMPIRICAL STUDY

The overarching aim of our empirical study is to understand the impact of automated planning support on the effectiveness of crowd-sourced planning. Accordingly, our general strategy was to compare the relative performance of different versions of **AI-MIX** with increasing levels of planning support. Thus we start with

i. a pure crowd planning instance,
ii. add constraint checking similar to Mobi's (Zhang et al., 2012) approach,
iii. further add subgoal generation (which exploits the PDDL model), and finally
iv. add the automated plan assembling step.

Our original aim was to compare all these versions of **AI-MIX** on the Amazon MTurk platform. However, we found that MTurk does not allow running external executables. Because of this, we split our experiments into two different setups. One on Amazon MTurk, that compared only the first

[2]http://reasoning.eas.asu.edu/f2lp/

| - | **C2** | **C3** |
|---|---|---|
| minimum | *47 secs* | *59 secs* |
| maximum | *8 mins 26 secs* | *2 mins 43 secs* |
| Average | *3 mins 9 secs* | *1 mins 26 secs* |

**Table 1.** HIT completion times for the two conditions: C2 and C3

three instances of **`AI-MIX`** and another set of experiments on a separate stand-alone platform that we built, to evaluate the **`AI-MIX`** system instance with plan assembling step. The next two sections give the details of the two setups, and the comparison results from the experiments.

## 6.1. Experiments on Amazon Mechanical Turk

In this subsection, we present details about the performance of the proposed system with respect to different experimental conditions that were all deployed on Amazon mechanical turk.

### *Experimental Setup*

For our study, HITs were made available to all US residents (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 generated plans, we compared 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.

We note here that in terms of planning support, Mobi system corresponds to the condition C2 in our framework (as it only supported constraint checking). 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 (only for the New York City) among C1, C2, and C3.

| # | Plan Suggestions | Our Observations |
|---|---|---|
| 1 | **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 $)<br>**Show:** Go to show #show(3 hours)(200.0 $)<br>**Show:** ABSOLUTELY CANNOT go wrong with Phantom of the Opera #show(3 hours)(200.0 $)<br>**Lunch:** Alice's Tea Cup #lunch(20.0 $)<br>**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 $)<br>**Dessert:** Serendipity #dessert(1 hours)(10.0 $) | 1) Redundant suggestions – In this example, the action *show* has multiple similar suggestions<br>2) Shorter plan suggestions<br>3) Vague suggestions – In this example, the action *show* has very high level descriptions to go and watch a show |
| 2 | **piccolo angolo:** Italian in the Village - real deal #italiandinner(2 hours)(60.0 $)<br>**Lombardi's Pizza:** #italian_dinner #italiandinner_todo1<br>**Ice Cream:** http://www.chinatownicecreamfactory.com/ #italiandinner_todo0<br>**#lunch:** Mangia Organics #lunch_todo0<br>**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 $)<br>**watch How to Succeed in Business:** Also a great show, a little less grand than Wicked. #broadwayshow(3 hours)(150.0 $)<br>**Activity Steamer:** #lunch #lunch_todo1<br>**Paradis To-Go:** Turkey & Gruyere is pretty delicious. The menu is simple, affordable, but certainly worth the time #lunch(1 hours)(10.0 $)<br>**cupcakes!:** Magnolia Bakery on Bleecker in the Village #dessert(1 hours)(10.0 $) | 1) Very detailed<br>2) Handling all the requester's preferences<br>3) Better utilization of the day |

**Table 2.** Sample activity suggestions from turkers for the two conditions: C2 (top) and C3 (bottom). For both of these conditions, same amount of money has been paid to each turker.

### *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. To receive responses for a HIT, the turkers have to accept the HIT before providing responses. In our scenario, C1 HITS were not accepted by any turker. 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.

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. We also verified the HIT completion time for a turker that we define as the difference between the time when he accepted the HIT and the time he submitted the same HIT. The average HIT completion time for C2 is 3 minutes 9 seconds where as, for C3 it is 1 minute 26 seconds. For C3, the task latency is higher but HIT completion time is lower as shown in Table 1. These results show that in case of C3, the probability of turker fatigue is low. As shown in the following section, the increased quality of generated plans may justify the task latency.

### *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. We consider the plan comparisons as we are conducting an ablation study to recognize the importance of automated planning technology on two similar systems (C2 and C3). 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 2. This seems to indicate that including a domain description in addition to the simplistic quantity and constraint checks increases the plan quality.

### *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 total of 31 new activity suggestions from turkers, of which 5 violated quantity constraints. The C3 setting attracted 39 responses, compared to 28 for C2, which may indicate that the planner tags encouraged turker participation. As shown by the HIT completion latencies, the planner module helped in reducing the probability of turker fatigue by requesting goal suggestions clearly. On the other hand, even though there is a task latency due to planner generating more subgoals to be fulfilled, the overall quality of the plans is significantly improved.

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. So it is interesting to see how the critiques are received by the other turkers. 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 relevant to the generation of a high quality plan than those suggested by other turkers. Even if we consider alternate explanations (e.g. the critiques of the planner were received better as they were easier to respond to), there is enough evidence to suggest that the presence of an automated system does help to engage and guide the focus of the crowd.

## 6.2. Experiments outside Mechanical Turk

As mentioned earlier, because of the difficulties of running the plan assembling executable (described in Section 5.4), on Amazon Mechanical Turk, we shifted to a stand alone platform to evaluate the full version of **AI-MIX** with all planning capabilities (including plan assembling support). We limited our comparison to the full fledged **AI-MIX** system, and an ablated version that removes the set of features that, taken together, set **AI-MIX** maximally apart from the earlier work on crowd-sourced planning (e.g. (Zhang et al., 2012; Law and Zhang, 2011)). These are:

– **AI-MIX**'s ability to do automatic subgoaling with its planner
– **AI-MIX**'s ability to provide verbal critiques of the turker plans
– **AI-MIX**'s ability to provide an activity plan that is scheduled on the timeline.

Accordingly, we developed an ablated system **AI-MIX**$^-$ that doesn't have these capabilities, but is otherwise identical to **AI-MIX**. This system doesn't use the knowledge base of the domain which was built based on the suggestions or critiques as in **AI-MIX**. Instead, the crowd creates a plan by adding new suggestions, deleting and rearranging the existing suggestions. We compared **AI-MIX**$^-$ to **AI-MIX** to understand the relevance and usability of the plans generated by both of these systems.

### *Experimental Setup*

One important aim of our study is to measure how useful and executable the plans that are generated by **AI-MIX** in real-life settings. To measure these metrics, we deployed another system as mentioned above **AI-MIX**$^-$ and consider the plan generated by this system to the plan generated by **AI-MIX**. We crowdsource both of these plans to obtain more accurate feedback from the humans on the usefulness of these plans obtained. For this analysis, we deployed both the **AI-MIX** and **AI-MIX**$^-$ systems on an independent server where anyone with a web url were given access to this system. We shared this system with 21 graduate students from the computer science department who participated in this experimental analysis. As mentioned, in this subsection we mainly focus on evaluating the usefulness of the generated tour plan from **AI-MIX** in comparison with **AI-MIX**$^-$.

**AI-MIX**$^-$ essentially focuses on generating plans by resolving the various quantitative and qualitative constraints. This system allows workers to specify constraints on different attributes as shown below:

– on the number of activities in each category
– amount of time to spend on activities in each category
– preferred combination of activities in the plan
– time constraints that state the cumulative duration of the activities in the itinerary

**AI-MIX**$^-$ provides a stream of suggestions on the interface where all the suggested activities are present and the workers can view, edit and delete the existing activities based on their opinion. The important missing factor here is that if one user modifies an existing activity, then it gets carried on to the future workers. For example, if a worker chooses to delete an activity from this stream, then there is no possibility to retrieve that deleted suggestion in **AI-MIX**$^-$. Whereas, in **AI-MIX** there is an option to modify an existing suggestion and the planner will decide to consider a given suggestion if it improves the quality of the final plan. Both the systems accept the suggestions till all the constraints are satisfied and there are no more concerns raised by any future worker.

In this study, both the systems ( **AI-MIX** , **AI-MIX**$^-$) were made available to the participants who didn't receive any money[3] for participating in this study. We used the tour planning scenario defined in section 5 for a city in South West United States, which is a familiar city for all the workers who were recruited. There are different ways that these workers contributed towards:

i. provide suggestions
ii. critique the existing suggestions
iii. vote on the best plan

The interface of the **AI-MIX** and **AI-MIX**$^-$ are both the same as shown in Figure 2. There are two main tasks involved here –

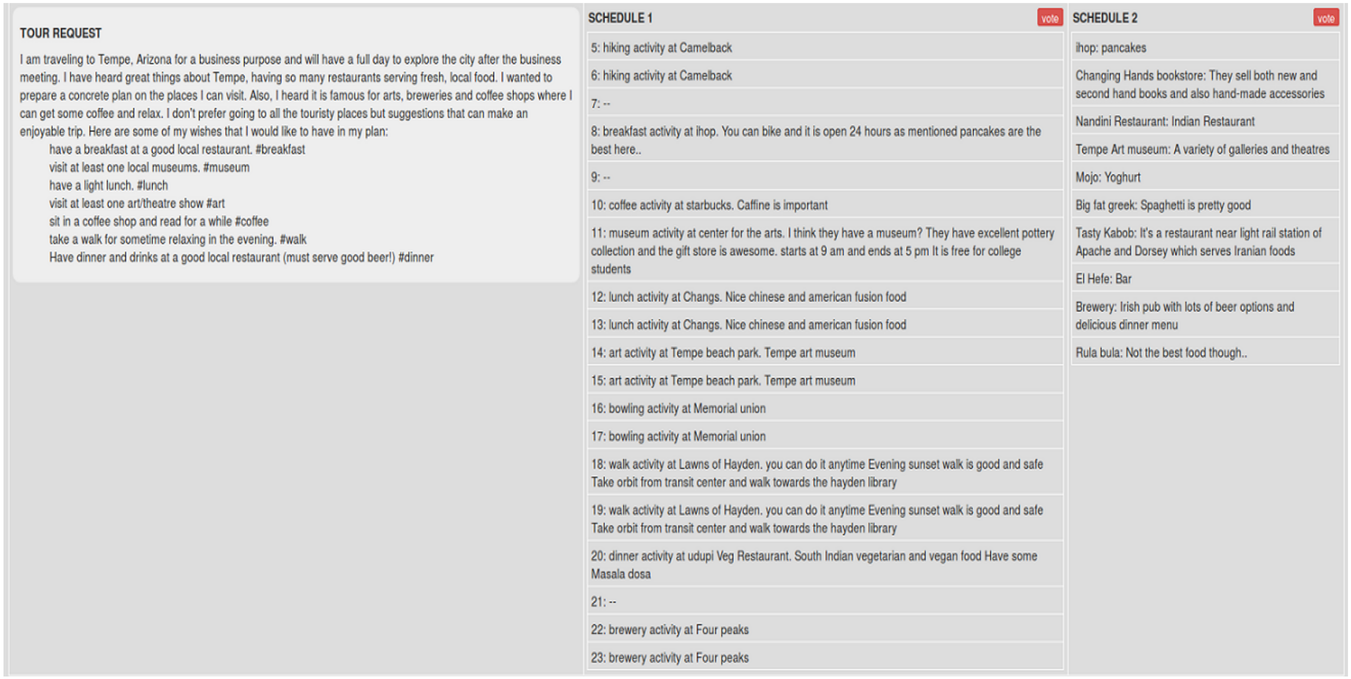
– *Task1* – provide suggestions or critique the existing suggestions
– *Task2* – vote on the best plan

Out of the 21 workers, we had 10 workers work on the first task – *Task1* and the rest of them to work on *Task2*. We separated both of these groups of workers as these tasks cannot be performed parallely if we want to understand the usefulness of the final plan. The interface for *Task2* is as shown in Figure 5.

### *Relevance of the Generated Plan*

As mentioned earlier, we recruited 10 workers (we refer this set of workers as $turk_{plangen}$) to work on *Task1* which is to measure the usefulness of the plans generated by the systems **AI-MIX** and **AI-MIX**$^-$. Subsequently, we asked the other set of 11 workers (we refer this set of turkers as $turk_{vote}$) to work on *Task2* which is to vote on a plan that will be most useful for them based on the tour request. This task mainly focuses on infering the quality of the plan in terms of relevance to the constraints laid by the requester along with the usability of the plan. On average, each worker participated in this experimental evaluation has given **2.4** suggestions for **AI-MIX** and **2.0** for the **AI-MIX**$^-$ system. Users felt that both the systems were fairly easy to work with, and had no issues with the interface while providing or critiquing the suggestions. The interface of the voting system is shown in Figure 5. In this interface, the task description is shown on the left most side and both the plans are shown on the right side of this interface. We can see the plan generated by **AI-MIX** followed by the plan generated by **AI-MIX**$^-$. Turkers can either vote for plan1 (generated by **AI-MIX** ) or plan2 (generated by the **AI-MIX**$^-$) which will be recorded for our analysis. We made sure that the

[3]The direction of whether the quality of completing the task is affected by the crowd's incentives is another research direction which is not the focus of this paper.



TOUR REQUEST

I am traveling to Tempe, Arizona for a business purpose and will have a full day to explore the city after the business meeting. I have heard great things about Tempe, having so many restaurants serving fresh, local food. I wanted to prepare a concrete plan on the places I can visit. Also, I heard it is famous for arts, breweries and coffee shops where I can get some coffee and relax. I don't prefer going to all the touristy places but suggestions that can make an enjoyable trip. Here are some of my wishes that I would like to have in my plan:

have a breakfast at a good local restaurant. #breakfast
visit at least one local museums. #museum
have a light lunch. #lunch
visit at least one art/theatre show #art
sit in a coffee shop and read for a while #coffee
take a walk for sometime relaxing in the evening. #walk
Have dinner and drinks at a good local restaurant (must serve good beer!) #dinner

SCHEDULE 1

5: hiking activity at Camelback
6: hiking activity at Camelback
7: --
8: breakfast activity at ihop. You can bike and it is open 24 hours as mentioned pancakes are the best here..
9: --
10: coffee activity at starbucks. Caffine is important
11: museum activity at center for the arts. I think they have a museum? They have excellent pottery collection and the gift store is awesome. starts at 9 am and ends at 5 pm It is free for college students
12: lunch activity at Changs. Nice chinese and american fusion food
13: lunch activity at Changs. Nice chinese and american fusion food
14: art activity at Tempe beach park. Tempe art museum
15: art activity at Tempe beach park. Tempe art museum
16: bowling activity at Memorial union
17: bowling activity at Memorial union
18: walk activity at Lawns of Hayden. you can do it anytime Evening sunset walk is good and safe Take orbit from transit center and walk towards the hayden library
19: walk activity at Lawns of Hayden. you can do it anytime Evening sunset walk is good and safe Take orbit from transit center and walk towards the hayden library
20: dinner activity at udupi Veg Restaurant. South Indian vegetarian and vegan food Have some Masala dosa
21: --
22: brewery activity at Four peaks
23: brewery activity at Four peaks

SCHEDULE 2

ihop: pancakes
Changing Hands bookstore: They sell both new and second hand books and also hand-made accessories
Nandini Restaurant: Indian Restaurant
Tempe Art museum: A variety of galleries and theatres
Mojo: Yoghurt
Big fat greek: Spaghetti is pretty good
Tasty Kabob: It's a restaurant near light rail station of Apache and Dorsey which serves Iranian foods
El Hefe: Bar
Brewery: Irish pub with lots of beer options and delicious dinner menu
Rula bula: Not the best food though..

**Figure 5.** The voting interface showing the tour request along with plan1 (**`AI-MIX`** ) and plan2 (**AI-MIX**$^{-}$)

| – | **Plan1** | **Plan2** |
|---|---|---|
| #of votes | 7 | 4 |
| Why this plan? | More organized<br>Clarity is much better<br>Better utilization of the day<br>I don't need to think at all<br>Presented variety of activities to-do | Better flexibility in schedule<br>Not regimented |

**Table 3.** Number of votes obtained by each system

turkers are not aware of the mapping between the plan and the system that generated this plan to avoid biases.

Table 3 shows that plan1 was voted for by 63% of workers and the rest of workers chose plan2 which is generated by the **AI-MIX**$^{-}$ system. Workers were asked why they chose a particular plan and the list of answers are compiled in Table 3. We believe that humans tend to have similar preferences which is the motivation to utilize the feedback obtained from the second set of users to evaluate the quality and usefulness of the plan generated from both the systems.

Along with these formal evaluations, we have also *demonstrated* the **AI-MIX** system at both a planning conference and a human computation conference. These "live" demonstrations involved engaging demo visitors in "turker" role so they could experience the **AI-MIX** interface and the crowd planning support it provides. The reception of the system in both venues was generally positive. Indeed, the system was the recipient of the "People's Choice Award for Best Demonstration" at the planning conference.

## 6.3. Discussion

In the first set of experiments where we have different conditions of **AI-MIX** that we tested on mechanical turk, we notice that **AI-MIX** is able to generate better quality plans which are more detailed. The turkers are able to attend to the different critiques issued by the system with relative ease (as shown by the HIT completion time). The other important measure of testing the quality is to make sure the plan suggestions provided span across all the different types of activities the requester has requested. This set of experiments addressed both the challenges of interpretation and steering by utilizing the planning techniques. Automated planning techniques seem to provide a clear additional advantage. As mentioned earlier, in terms of planning support, Mobi system corresponds to the condition C2 in our framework (as it only supported constraint checking).

In the second set of experiments, **AI-MIX** is able to generate a very useful schedule that considered different types of activities. One worker complained that the plan generated by the **AI-MIX**$^{-}$ system mainly focused on food-related activities with a lesser focus on other activities. Even though plan2 is very flexible and is not regimented, plan1 is more detailed and is better at utilizing the entire day while being more organized. This shows that subgoal generation and critiquing at various levels to generate a better detailed plan is very important. **AI-MIX** includes these features which helps in generating better quality plans over the other types of systems. We know that there is incompleteness involved in the model and preferences and there is an impedance mismatch between the planner's and humans' model. We can infer from these results that both of these issues were adequately handled by utilizing the strengths of both machines (automated critiquing and subgoal identification) and humans (providing plan suggestions and critiquing) to generate a better quality plan that considers requester's preferences into account. Though the current system uses only a preliminary form of automated reasoning, this effort can be seen as the first step towards incorporating more sophisticated methods for plan recognition and generation.

We consider that the approach we proposed as part of the system can be generalized to other real-world domains under these two conditions:

i. When the partial model of the domain is provided by the designer.

ii. Turkers or workers are knowledgeable about the domain that is under consideration are available.

## 7. 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 empirical results over the tour planning domain in two different experimental setups, 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. Considering this, as a future work we are interested to see if the existing work on learning domain models (Yang et al., 2007; Tian et al., 2016) can be adapted to allow learning from observing the plans suggested by the crowd. We are also interested in eliciting information (Kaplan et al., 2013) from the crowd in order to go from the current list of activities that are produced as the result of the crowd's planning process, to a more structured *plan* in the traditional sense of the word.

### 7.1. Funding Acknowledgements

This research is supported in part by a Google research award and the ONR grants N00014-15-1-2027, N00014-16-1-2892, N00014-13-1-0176 and N00014-13-1-0519.

## 8. REFERENCES

Brewka, G, Delgrande, J, Romero, J, and Schaub, T. (2015). asprin: Customizing Answer Set Preferences without a Headache. In *AAAI Conference on Artificial Intelligence*.

Dai, P, Lin, C. H, Mausam, , and Weld, D. S. (2013). POMDP-based control of workflows for crowdsourcing. *Artificial Intelligence* 202, 0 (2013), 52 – 85.

Dai, P, Mausam, , and Weld, D. S. (2010). Decision-theoretic control of crowd-sourced workflows. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Dai, P, Mausam, , and Weld, D. S. (2011). Artificial intelligence for artificial artificial intelligence. In *Proc. of AAAI*.

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

Kamar, E, Hacker, S, and Horvitz, E. (2012). Combining Human and Machine Intelligence in Large-scale Crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1 (AAMAS '12)*. 467–474.

Kamar, E and Horvitz, E. (2015). Planning for Crowdsourcing Hierarchical Tasks. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*.

Kamar, E, Kapoor, A, and Horvitz, E. (2013). Lifelong Learning for Acquiring the Wisdom of the Crowd. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13)*. 2313–2320.

Kambhampati, S. (2007). Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain theories. *Proceedings of AAAI 2007* (2007).

Kambhampati, S and Talamadupula, K. (2015). *Human-in-the-Loop Planning and Decision Support*. Technical Report. AAAI 2015 Tutorial. rakaposhi.eas.asu.edu/hilp-tutorial.

Kaplan, H, Lotosh, I, Milo, T, and Novgorodov, S. (2013). Answering planning queries with the crowd. *In Proc. of VLDB Endowment* 6, 9 (2013), 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 von Ahn, L. (2011). Human Computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool* (2011).

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

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

Mao, A, Kamar, E, Chen, Y, Horvitz, E, Schwamb, M. E, Lintott, C. J, and Smith, A. M. (2013). Volunteering vs. Work for Pay: Incentives and Tradeoffs in Crowdsourcing. In *Human Computation Workshop 2013*.

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* (1998).

Miller, G. A. (1995). WordNet:A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

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 (2003), 379–404.

Quinn, A. J and Bederson, B. B. (2011). Human Computation: A survey and taxonomy of a growing field. In *Proceedings of the 2011 annual conference on Human factors in computing systems*. 1403–1412.

Ramírez, M and Geffner, H. (2010). Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *Proc. of AAAI*.

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

Tian, X, Hankui Z., H, and Kambhampati, S. (2016). Discovering Underlying Plans Based on Distributed Representations of Actions. In *Aamas*.

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*.

Weld, D. S, Mausam, , and Dai, P. (2011). Human Intelligence Needs Artificial Intelligence. In *Human Computation Workshop 2011*.

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

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*.

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.

Zhuo, H. H. (2015). Crowdsourced Action-Model Acquisition for Planning. In *Proc. of AAAI*.

Zhuo, H. H, Kambhampati, S, and Nguyen, T. A. (2012). Model-Lite Case-Based Planning. *CoRR* abs/1207.6713 (2012).