# Chapter 1

# Introduction

The Internet has evolved into a platform on which large numbers of individuals take action and join in collaborations. Just a decade or two ago, the Internet was used primarily as a source of information. Today, the Internet is a center for social and economic activity. In social computing systems such as Wikipedia and on crowdsourcing platforms such as Amazon Mechanical Turk, large numbers of individuals contribute to problem-solving efforts as volunteers and as paid workers. On social media services such as Facebook and Twitter, friends and followers communicate news, share thoughts and ideas, and engage in social and political action. In electronic markets such as eBay, Etsy, and Amazon, consumers make purchases and help one another with purchasing decisions by contributing ratings and reviews. Over the years more and more activity is taking place online, and this trend promises to continue as the Internet continues to evolve.

In hosting platforms and services, the Internet is a virtual space on which designers build systems in which participants take action. From the designer's perspective, the

goal of a system is to attract participants and promote particular desired actions and outcomes. For example, Wikipedia seeks to attract contributors to write, review, and edit articles. Requesters on Mechanical Turk want to recruit workers to complete tasks well and on-time. Facebook and Twitter want users to contribute content, communicate with other users, and generally make use of available features.

I refer to the problem of designing social and economic systems on the Internet to promote desired actions and outcomes as *computational environment design*. The designer's role is to construct the *decision environment* in which participants take action. The decision environment may include interfaces, workflows, feedback to users, incentives, constraints on actions, rules and policies, and so forth. Participants have their own knowledge and abilities, interests and motivations, availability, and decision-making processes. Together with the decision environment, all of these elements influence participants' decisions about what actions to take in a system.

To elicit desired behaviors, a designer must construct a decision environment with which to drive participant actions. For example, an effective decision environment may include tools that enable collaboration among contributors, monetary and social rewards for taking desired actions, or interface elements that display relevant information for decision-making. Depending on the decision environment being constructed, effective designs may draw on principles and methods from fields such as human-computer interaction, artificial intelligence, decision science, psychology, sociology, and economics.

Notable abilities afforded by the Internet are creating new opportunities and inspiring new methods for computational environment design. One example is the

ability to recruit a crowd of individuals to join in problem solving and discussion. By employing workers in an online labor market such as Amazon Mechanical Turk and reaching out to friends and followers through social media services such as Facebook and Twitter, one can now draw on a crowd to contribute to seemingly arbitrary tasks of interest. Taking advantage of this ability, *crowdsourcing* and *human computation* systems are attracting large numbers of participants to solve large-scale problems. While individuals in the crowd may only be involved briefly, and while any given individual may provide noisy inputs, we are beginning to develop mechanisms for coordination and quality control that enable a crowd to provide useful solutions in a variety of settings.

One can envision a future in which the *distributed intelligence* of humans and machines across networks are brought together to tackle complex problems, with streams of tasks flowing seamlessly to the people who are most willing and able to contribute. Despite individual limitations, crowds of humans and machines may be able to perform complex tasks that cannot be solved by humans or machines working independently. A key challenge in realizing this vision is understanding how to design decision environments that help to recruit individuals with relevant expertise to join a problem-solving effort, and that enable effective coordination and collaboration.

Another notable ability afforded by the Internet with implications for computational environment design is the ability to engage in a data-driven, iterative design process. Designers can experiment with alternative designs by modifying some aspects of a system, and track complex individual and group behaviors across multiple interactions to get rapid feedback on designs. For example, tools for A/B testing are

enabling designers to iteratively make changes to their web services to better promote desired outcomes by putting hypotheses to the test and objectively measuring the outcomes of competing designs.

The ability to track complex behaviors and redesign easily not only allows designers to compare alternative designs, but also allows them to gain new insights into participants' motivations and decision-making processes. Currently, the iterative design process is largely manual and ad hoc. Designers come up with alternative designs themselves, and the experimentation process is aimed at "hill-climbing" to a local maximum. This process is not only tedious for the designer, but may miss out on parts of the design space where better solutions exist. As improved models of participant behavior and computational tools for understanding participants from data become available, one can envision a future in which we can design decision environments *automatically* by systematically discovering interventions tailored to the preferences and capabilities of participants. Such methods aim to provide for more efficient iterative design processes, that seamlessly combine domain knowledge with machine-driven processes that refine models based on observed behavior.

In this dissertation, I introduce principles and methods for crowdsourcing complex tasks and for automated environment design. I demonstrate how to discover solutions to computational environment design problems manually and automatically. A common thread in my approach is to construct effective designs by *reasoning and learning about characteristics of participants and how these characteristics interact with the decision environment to influence behavior.* By reasoning, I mean the action of thinking about participants and a design problem using available knowledge. By learning,

I mean the acquisition of knowledge about participants through experience or study that informs design decisions.

In the first part of the dissertation, I show how reasoning about crowd abilities and limitations can lead to designs that make crowdsourcing complex tasks feasible, effective, and efficient. I demonstrate a number of design patterns that allow the crowd to effectively coordinate and contribute to complex tasks, and provide incentive mechanisms that encourage individuals to both contribute to a task and route the task to others who can further contribute.

In the second part of the dissertation, I provide a general framework for *automated environment design* that learns relevant characteristics of participants based on observations of behavior and optimizes designs based on learned models. I prove theoretical properties of a method inspired by this framework, and demonstrate the feasibility and effectiveness of automated design procedures for automatically designing crowdsourcing tasks and synthesizing crowd workflows.

## 1.1 Crowdsourcing Complex Tasks

### 1.1.1 Human Computation and Crowdsourcing

Over the last decade, *human computation* [95, 52] has established itself as a powerful paradigm for incorporating human intelligence in problem-solving efforts in which machines cannot yet tackle the problem alone. Such systems take advantage of human abilities—e.g., in vision, natural language, and pattern recognition—to handle instances and aspects of problems that are difficult for computers. The ESP game [97],

FoldIt [15], and reCAPTCHA [98] are a few examples of successful systems that draw on human contributors and machine computations to tackle problems in image labeling, protein folding, and text digitization.

Many human computation systems treat humans as processors in a distributed system, each performing a small part of a massive computation [95]. But unlike computers, humans require an incentive to contribute to a computational effort. This incentive may be in the form of monetary rewards, a sense of duty or purpose, or enjoyment of the task. For example, a human computation system may recruit a crowd of paid workers through an online labor market such as Amazon Mechanical Turk, in which workers receive small amounts of money for completing "microtasks." Another system may draw on a crowd of friends or followers on social media services such as Facebook and Twitter, who may be willing to contribute based on their relationship with the requester or other contributors. Yet another system may attract a crowd of users of a web service who are willing or required to contribute to a task, e.g., either because the task itself has been made enjoyable as in games with a purpose like the ESP game [97], or because completing the task is required for accessing content of value, as in reCAPTCHA [98].

Human attention is limited, and the incentive for individuals to contribute to a task is also limited. Practically, attracting a large crowd to perform an arbitrary task often implies that individuals in the crowd may only be involved briefly, and that any given individual in the crowd may provide noisy inputs. This is in contrast to the way work is performed in traditional firms and also in social computing systems such as Wikipedia, where employees and dedicated volunteers are available over time and

can be relied upon to work on larger problems, keep track of context, identify issues, and solve problems that arise. To handle short periods of work and noisy inputs, human computation systems aim to break down large problems into smaller tasks, and provide means of quality control for synthesizing noisy inputs from large numbers of individuals.

### 1.1.2 Human Computation Algorithms

While simple tasks may be easy to parallelize across individuals, *complex tasks* require more sophisticated coordination and optimization. Over the last several years, there has been a rise of *human computation algorithms*, or *workflows*, that decompose a task into more manageable, self-contained subtasks. Human computation algorithms aim to allow individuals to contribute to small subtasks independently, without having to reason about other subtasks or the task at large.

By drawing on core computational principles, a number of *design patterns* have emerged that serve as the basic building blocks of human computation algorithms. For example, Little et al. [60] introduced an iterative design pattern in which each member of the crowd improves upon the previous solution. Zhang et al. [105], Bernstein et al. [5], and Kittur et al. [46] demonstrated how divide-and-conquer can be applied to the crowd in which crowd workers decompose problems, solve subproblems, and recompose subproblems into a solution. Methods for quality control can be incorporated at various points within an algorithm to promote high quality results.

Given a particular problem, crowdsourcing a solution may involve constructing a human computation algorithm that utilizes multiple design patterns. In Chapter

2, I will present an approach for formulating effective workflows by reasoning about characteristics of the crowd. I will demonstrate how this approach leads to design patterns, algorithms, and frameworks that combine existing design patterns to enable the crowd to reach expert level performance on complex tasks such as audio transcription and nutrition analysis.

### 1.1.3 Human Computation with Global Constraints

An important class of underexplored human computation tasks are those in which the solution must satisfy a set of global requirements. For example, in leveraging the crowd to write an essay, a requester may want to specify requirements on the desired tone, tense, length, structure of arguments, and style of exposition that must hold consistently throughout a piece of writing. Some requirements, e.g., presenting a balanced perspective on a situation, touch upon different components of the essay and depend on the essay as a whole.

As another example, consider the problem of crowdsourcing itinerary planning. Planning events such as vacations, outings, and dates often involve an *itinerary* which contains an ordered list of activities that are meant to be executed in sequence. People going on a trip have preferences and constraints over the types of activities of interest (e.g., "I am interested in history museums"), how long to spend on different activities (e.g., "I want to spend at least 2 hours hiking"), the composition of activities (e.g., "I want to focus on art galleries and museums for the day"), the budget, and the total time available, which define a set of global requirements that an itinerary should satisfy.

For these and other tasks that involve global constraints, good solutions rely on the composition of different contributions as a whole, with interdependence among solution components. As such, it is not clear how to break down these tasks into smaller tasks for individuals in the crowd to complete independently.

In Chapter 3, I will introduce *crowdware* as an approach for tackling human computation tasks with global constraints. Crowdware draws inspiration from *groupware* [25], which suggest principles and ideas on communication and collaboration within a shared context that help a group accomplish a joint task. Crowd workers differ from groups in that individuals may only be involved briefly, may be less willing to spend time grasping the solution context or taking meta-level actions, and may not fully consider the overall objective nor the aims of other crowd workers when making decisions. To address this challenge, crowdware provides mechanisms in which the system (indirectly) coordinates the problem-solving effort by focusing the crowd's attention on what needs work. I will present a system for crowdsourcing itinerary planning called Mobi, to illustrate this concept.

### 1.1.4 Harnessing Crowd Abilities: Control and Synthesis

Human computation algorithms tend to define an explicit sequence of steps in which individuals in the crowd are recruited to complete subroutines within this predefined process. However, there are also opportunities for the crowd to contribute to a problem-solving effort by guiding the control flow of an algorithm or even generating plans that define the problem-solving process. Taking this broader perspective, I explore in Chapter 4 different ways in which the crowd can contribute to a human

computation process, leading to new applications and more efficient forms of problem solving.

An example of having the crowd guide the problem-solving process is *task routing*. Task routing aims to harness the ability of people to both contribute to a solution and to route tasks to others who they believe can effectively solve and route. Task routing provides an interesting paradigm for problem solving in which individuals become engaged with tasks based on their peers' assessments of their expertise. On the task level, effective task routing aims to take advantage of participants' individual expertise as well as participants' knowledge about others' abilities to contribute. On the organizational level, task routing can provide a means for bringing tasks to individuals effectively, where participants' routing decisions take into account not only an individual's expertise on a particular task, but also their ability to contribute as a router.

In Chapter 5, I will introduce incentive mechanisms that reward individuals for solving and routing tasks. An interesting problem that arises is that incentives need to take into account limitations on individuals' knowledge about the knowledge of others. For example, in a social network setting where individuals may only know about the expertise of those that are close to them in social distance (e.g., their friends, and possibly friends of friends), the would-be optimal incentive mechanism designed under the assumption that everyone knows everyone else's expertise may not work as desired. I will show how we can design incentive mechanisms that are sensitive to such limitations, while still promoting effective routing decisions.

## 1.2 Automated Environment Design

Many tools on the Internet facilitate data-driven, iterative design processes. Web analytics software tracks complex individual and group behaviors over time, and provides summary information on trends and patterns in the data. Frameworks, style sheets, and content management systems make it easier to modify or extend existing designs. Tools for A/B testing allow designers to compare alternative designs based on defined objectives.

Despite having a rich set of tools, identifying effective designs to elicit desired behaviors remains a process that is largely manual, tedious, and ad hoc. One potential solution is to automate the environment design process to systematically explore a design space in a principled, data-driven manner. Such an approach may be able to discover effective designs quickly, while requiring less manual effort. An automated environment design system may take as input a set of available interventions, the objective of the designer, and a model of the interaction among environment, participants, and behaviors, and provide as output an intervention that promotes actions and outcomes meeting the objective whenever such interventions exist.

One challenge in building such a system is that models of behavior are imperfect and incomplete. We have limited knowledge of users' preferences and decision-making processes, and this private information is difficult to elicit directly. But in online settings where the designer can track individual and group behaviors, such information can be indirectly inferred from observing actions over repeated interactions. For example, one can infer from observing consumer purchasing decisions and worker performance on tasks the underlying preferences and abilities that guide their deci-

sions and the actions observed. These observations can be used to refine existing models and drive better design decisions.

While observing user behavior can provide some information, observations of user behavior under any particular design will only provide partial information about users' underlying preferences or abilities. For example, a consumer's purchase decision does not completely reveal their underlying value for a good (which may be at or above the purchase price), and a worker's performance on a task does not reveal exactly how the worker will perform on a different task. But given the ability to experiment with alternate designs and receive rapid feedback on different designs, it may be possible to iterate and refine our understanding of participants over time.

In Chapter 6, I will present a general approach for automated environment design that draws on these observations. I will provide an *active, indirect elicitation* framework that automatically drives an objective-based iterative design process by interweaving indirect learning of model parameters with optimization to determine effective interventions based on the current model. In Chapter 7 and 8, I will show how to apply ideas from automated environment design to automatically design crowdsourcing tasks and synthesize crowd workflows.

## 1.3 Limitations

Approaching computational environment design problems by reasoning and learning about characteristics of participants leads to solutions that are tailored to the participants. As such, specific results generalize only as far as the assumed characteristics of participants hold true across domains. For example, in addressing problems in

crowdsourcing complex tasks, I propose designs that take into account some assumed characteristics of crowd workers (e.g., they may only make small contributions) and demonstrate that the designs are effective for participants satisfying such characteristics. But in other settings, these particular assumptions about the crowd may be false, and other characteristics of participants such as their intrinsic motivation may require very different designs.

Related to this, I assume throughout the dissertation that characteristics of participants remain more or less constant, and do not reason explicitly about how these characteristics may change over time or how potential changes may affect design decisions.

The automated environment design framework is most applicable for learning and reasoning about the characteristics of participants, and using this to parametrize designs, rather than for discovering effective design patterns in the first place. For complex domains, proposing a design space that includes effective designs may require significant amounts of domain knowledge. Automated design tools can help to refine existing models through experimentation and provide new insights, but are not yet a replacement for human ingenuity.

## 1.4 Thesis and Contributions

My thesis statement is:

> By reasoning and learning about characteristics of participants and how these characteristics interact with the decision environment to influence behavior, we can design environments that elicit desired actions and outcomes.

My contributions span several areas. Contributions related to crowdsourcing complex tasks include:

- Design patterns, algorithms, and frameworks that effectively utilize combinations of existing design patterns to enable the crowd to reach expert level performance on complex tasks. Constructed workflows coordinate the contributions from the crowd, and are effective in applications to crowdsourced audio transcription and crowdsourced nutrition analysis based on food photographs.

- A crowdware design pattern that enables a crowd to effectively resolve global constraints. Crowdware provides a shared, collaborative workspace through which individuals in the crowd contribute opportunistically based on the current solution context, and in which the system indirectly coordinates the problem-solving effort by alerting crowd workers to what needs work. This design pattern is demonstrated through a system called Mobi, in which the crowd generates custom itineraries for day trips.

- Methods and design elements that leverage the crowd's ability to control an algorithmic procedure and generate plans that define the problem-solving process. An experiment on the 8-puzzle that involves sharing problem-solving strategy, and a system called CrowdPlan that produces simple plans in response to high-level search queries, demonstrate that these methods and design elements can enable effective problem solving and novel applications.

- *Routing scoring rules* for prediction tasks that properly incentivize participants to jointly contribute to a task and route the task to others for further contribu-

tions. Theoretical results characterize the family of routing scoring rules that promote tractable routing decisions based only on local information.

Contributions related to automated environment design include:

- A model of the computational environment design problem.
- A general framework for active, indirect elicitation for automated environment design.
- Efficient algorithms for active, indirect elicitation in sequential decision making settings in which the principal can modulate costs and rewards, along with theoretical results about convergence.

Contributions that relate to both crowdsourcing and automated environment design include:

- Crowdsourcing applications that demonstrate the effectiveness of applying the automated design approach to designing an image labeling task and to synthesizing sorting algorithms tailored to crowd abilities.
- An automated design framework and associated learning and optimization algorithms for synthesizing crowd workflows.

## 1.5 Thesis Overview

This dissertation consists of two major components. The first component demonstrates how we can reason about crowd abilities and limitations to discover effective designs for crowdsourcing complex tasks:

- Chapter 2 introduces design patterns from the literature that serve as the basic building blocks for designing human computation algorithms. The chapter shows how to combine these design patterns to construct human computation algorithms for audio transcription and nutrition analysis.

- Chapter 3 shows how to tackle human computation tasks that are difficult to decompose. The chapter introduces Mobi, a system for crowdsourced itinerary planning. Mobi illustrates a novel *crowdware* design for tackling complex tasks with global constraints by using a shared, collaborative workspace. Experiments and user studies show that Mobi enables the crowd to effectively resolve violated constraints, and generates itineraries that satisfy users' stated requirements.

- Chapter 4 provides an overview of the different ways in which crowds can contribute to a problem-solving process by guiding the control flow of an algorithm and generating plans that define the problem-solving process. As examples, the chapter shows how passing around context can enable a crowd to more effectively solve a version of the 8-puzzle, and introduces a system called CrowdPlan, that leverages a crowd to generate simple plans for accomplishing high-level tasks.

- Chapter 5 describes methods for task routing that aim to harness people's ability to both contribute to a solution and to route tasks to others who they believe can further contribute. Focusing on prediction tasks, the chapter introduces routing scoring rules that reward effective contributions via solving and routing. The chapter also identifies a family of *local routing rules*, which isolate simple

routing decisions in equilibria that are invariant to non-local information while still promoting effective routing and information aggregation.

The second component focuses on constructing automated procedures that can automatically derive effective designs by reasoning and learning about participants:

- Chapter 6 introduces a general approach for automated environment design and describes an active, indirect elicitation framework for iteratively refining designs. As an illustrative example, the chapter introduces the problem of *policy teaching*, in which the goal is to elicit a desired policy from a single agent in sequential decision domains modeled as Markov Decision Processes. Theoretical results provide conditions under which an algorithm applying the active, indirect elicitation framework is guaranteed to discover an effective intervention after a small number of interactions.

- Chapter 7 explains how to automate the design of human computation tasks. Using image labeling as an example, the chapter shows how to learn models of crowd performance as a function of design parameters and derive new designs by optimizing over learned models. Experimental results demonstrate that optimized designs collect significantly more high quality labels than baseline designs at the same rate of pay.

- Chapter 8 explains how to automatically synthesize crowdsourcing workflows. The chapter introduces an active, indirect elicitation based approach that selects experiments to refine current models of the crowd's performance on tasks in order to synthesize algorithms that optimize desired objectives given resource

constraints. The approach is demonstrated to be effective through human sorting tasks that leverage the crowd to determine the ordering among objects.

Chapter 9 concludes the dissertation with a summary of contributions and a discussion of future research directions.

## 1.6 For the Reader

Most of this dissertation is intended to be readable by a general audience with interest in the design of social and economic Internet systems. The most technical chapters are 5, 6, and 8, where some familiarity with game theory (Chapter 5), decision science (Chapters 6 and 8), and artificial intelligence (Chapters 6 and 8) is assumed. Technical details in these chapters may be skipped with little loss to understanding the principles and ideas being introduced, nor the overarching message of approaching computational environment design problems through reasoning and learning about participants.

## 1.7 Bibliographic Notes

Chapter 2 includes work on iterative dual pathway structure [56] with Beatrice Liem and Yiling Chen, and work on PlateMate [69] with Jon Noronha, Eric Hysen, and Krzysztof Gajos. Chapter 3 presents work on Mobi [106] with Edith Law, Rob Miller, Krzysztof Gajos, David Parkes, and Eric Horvitz. Chapter 4 discusses work on crowdsourcing general computation [105] with Eric Horvitz, Rob Miller, and David

Parkes, and work on CrowdPlan [53] with Edith Law. Work on task routing [104] in chapter 5 was conducted with Eric Horvitz, Yiling Chen, and David Parkes.

Chapter 6 discusses work on automated environment design [103, 107, 108] with David Parkes and Yiling Chen. Chapter 7 presents work on automated task design [38] with Eric Huang, David Parkes, Krzysztof Gajos, and Yiling Chen.