Chapter 9

Conclusion

The Internet today is a center for social and economic activity. Through crowdsourcing, social media, and electronic commerce, social and economic systems on the Internet attract large numbers of individuals to take action and join in collaborations. From a system designer's perspective, a key challenge is understanding how to promote particular desired participant behaviors and outcomes. I call this problem *computational environment design*.

The designer's role is to construct the *decision environment* in which participants take actions. This can include interfaces, workflows, feedback to users, incentives, constraints on actions, rules and policies, and so forth. Participants have their own preferences and capabilities, that together with the decision environment influence their behavior. As the designer can only affect participants' actions and outcomes indirectly through the decision environment, solving computational environment design problems may rely on understanding participants and tailoring designs to the participants.

In this dissertation, I propose an approach for solving computational environment design problems by reasoning and learning about characteristics of participants and how these characteristics interact with the decision environment to influence behavior. By reasoning, I mean thinking about participants and a design problem using available knowledge. By learning, I mean the acquisition of new knowledge about participants that informs design decisions.

I focus on two notable abilities afforded by the Internet that speak directly to the computational environment design problem. The first is the ability to recruit a crowd. Taking advantage of this ability, *crowdsourcing* and *human computation* systems are attracting crowds to solve large-scale problems. From a computational environment design perspective, this presents an exciting opportunity for designers to recruit large numbers of interested participants for the *explicit purpose* of performing useful actions that help to achieve desired outcomes.

A practical challenge that arises when attracting a large crowd to perform an arbitrary task is that individuals may only be briefly involved, and any given individual may provide noisy inputs. Leveraging a crowd to complete a complex task may thus require coordinating small, noisy contributions from large numbers of participants, or identifying and attracting individuals who are most willing and able to contribute.

In the first part of the dissertation, I show how reasoning about crowd abilities and limitations can lead to designs that enable a crowd to effectively contribute to solving complex tasks. In seeking to leverage the *distributed intelligence* of the crowd, I make advances in three core directions. The first direction is the coordination among problem solvers. I demonstrate how existing design patterns can be effectively combined

to construct human computation algorithms and new design patterns that enable the crowd to solve complex problems (Chapters 2). I also introduce a *crowdware* design, that enables the crowd to tackle complex tasks involving global constraints which cannot be easily decomposed and solved using human computation algorithms (Chapter 3).

The second direction is harnessing the *general intelligence* of the crowd. I explore methods and designs that engage the crowd to guide the control flow of an algorithm and generate plans that define the problem-solving process (Chapter 4). In studying effective means for passing solution context in the 8-puzzle and a system called CrowdPlan for generating simple plans to high level search queries, we are beginning to explore principles for *crowdsourcing general computation* that can enable general problem solving via human computation systems.

The third direction is the recruitment of expertise. I study *task routing* as an approach for problem solving in which individuals both contribute to a solution and route to others for further contributions (Chapter 5). Focusing on prediction tasks, I introduce incentive mechanisms that promote participants to honestly report private information and route tasks to people who they believe can best contribute.

In the process of arriving at effective designs, I find that designs that are effective for small groups of people are not necessarily effective for the crowd. Such designs often needed to be rethought and adapted to explicitly take into account crowd abilities and limitations. For example, in studying Mobi, we observed how automatically generated *todo items* are crucial for helping the crowd keep track of violated constraints in the process of generating an itinerary. Given crowd workers who are only briefly

involved and not available over time to keep track of solution context, the todo items take on the role of a "dedicated volunteer," who is there always to provide feedback and make suggestions about how to move toward a solution.

As another example, when constructing *routing scoring rules* for task routing for prediction tasks, we observed that it is possible to construct incentives such that if everyone knows about the network structure and everyone else's expertise, then in equilibrium everyone would route along the optimal path. But in social networks on the Internet, individuals may only know (the expertise of) people within their local neighborhood, which may only include their friends and possibly friends of friends. Implementing the would-be optimal incentive scheme would require people to perform complicated inference and may not work as desired. As a solution, we introduced a class of *local routing rules*, that are designed to explicitly enable equilibrium behavior for which the inference required of participants is local and thus tractable.

The second ability afforded by the Internet with implications for computational environment design is the ability for designers to engage in a data-driven, iterative design process. The Internet provides a wide range of tools for iterative design, that include web analytics software for tracking user behavior; style sheets, frameworks, and content management systems for redesigning easily; and tools for A/B testing for comparing designs based on desired objectives. From a computational environment design perspective, these tools provide a valuable resource for designers of Internet systems to easily experiment with alternative designs, collect rich behavioral data from large numbers of users, and iterate quickly to improve designs over time.

But despite having these tools, the process of designing for effective behavior on

the Internet remains largely manual, tedious, and ad hoc. Experiments are often conducted on alternative designs that consist of small modifications aimed at making incremental improvements against set objectives. Without particular regard to understanding participants and their behavior, this can lead to a design process that hill-climbs toward a solution at a local rather than global maximum. Designers may miss out on better designs, and ultimately fail to promote desired behaviors and outcomes.

In the second part of the dissertation, I introduce principles and methods that enable an automated system to systematically explore a design space to elicit desired behavior by reasoning and learning about participants. In *automated environment design*, a system takes a model of the interaction among decision environment, participants, and behaviors and seeks to quickly identify an effective intervention from a space of possible interventions. We introduce an *active, indirect elicitation* framework that drives an objective-based, iterative design process (Chapter 6). The framework makes use of an *inference procedure* and an *elicitation strategy*. The inference procedure uses observations of participant actions to learn about participants and refine existing models of behavior. The elicitation strategy complements the inference procedure by designing experiments to refine existing knowledge.

We find through applications to crowdsourcing that an automated system using observations of participant actions to refine a model of behavior can discover effective designs tailored to the participants that achieve significantly better outcomes than designs available prior to learning. In *automated task design*, we learned models of the quality of worker output to an image labeling task as a function of task design param-

eters and used learned models to construct optimized designs that are demonstrated to be more effective at the same rate of pay (Chapter 7). In *automated workflow synthesis*, we learned models of the crowd's performance on pairwise comparison and pivot selection tasks and used learned models to better allocate effort within a human quicksort algorithm to achieve high quality solutions (Chapter 8).

In general, a design space may be very large, and exploring it blindly may not lead to effective designs. By learning about participants from observing their behavior in response to different designs, we can effectively narrow the space of possible designs we need to consider. This is because our knowledge of participants gives us a better sense of which designs may be effective or ineffective. In the extreme case where we have a perfect, known model of how participants make decisions with respect to different designs, identifying the best design becomes an optimization problem with known parameters. When studying the problem of *policy teaching*, we take advantage of this insight and develop a centroid-based elicitation strategy that is guaranteed to elicit the desired behavior after few interactions (Chapter 6). The elicitation strategy does this by basing incentives on hypotheses that, if correct, will elicit the desired behavior, and if incorrect will lead to an observation that significantly narrows the space of agent rewards that are consistent with observed behavior.

In addition to exploring a design space in a principled manner based on models of participant behavior, to be practically useful, automated environment design systems need to be able to discover effective designs quickly. In the context of computational environment design, the goal is not to learn about participants for learning's sake but rather to elicit desired behaviors and outcomes quickly. Focusing on this, in studying

automated workflow synthesis, we introduced a *value of information* based elicitation strategy that selects experiments by estimating the expected value that can be derived from potential improvements to the current choice of algorithm as the result of new information (Chapter 8). By incorporating the objective of the designer directly into the elicitation strategy, a value of information approach focuses the learning effort on exploring parts of the design space where learning is most likely to matter.

In both manual and automated approaches to solving computational environment design problems, reasoning and learning about participants allows us to discover effective solutions that are tailored to the participants. In the rest of the chapter, we briefly review the main contributions and results, and present directions for future work.

9.1 Brief Review

The first part of the dissertation focused on human computation and crowdsourcing and introduced design patterns and methods for recruiting and coordinating a crowd to tackle complex tasks.

Chapter 2 studied the design of human computation algorithms that enable the crowd to contribute effectively to complex tasks. Through the problem of crowdsourcing audio transcription, I discovered an *iterative dual pathway structure* that effectively combines the output-agreement design pattern with the iterative design pattern to encourage contributors to provide accurate improvements. This design pattern eliminates the need for explicit quality control via voting or grading and focuses the crowd's effort on improving solutions instead. I then considered the

problem of crowdsourcing nutrition analysis from food photographs. I introduced a system called PlateMate, whose workflow consists of multiple, heterogeneous tasks that request from the crowd diverse contributions such as tagging food items, describing ingredients, and measuring portions. PlateMate is built on the management framework inspired by the structure of human organizations, which provides effective support for managing complex workflows involving heterogeneous tasks.

Chapter 3 presented a crowdware design that enables a crowd to tackle complex tasks with global constraints through a shared, collaborative workspace. Focusing on crowd itinerary planning as a case study, I introduced a system called Mobi. Mobi presents a single interface through which individuals in the crowd can see the current solution and all ideas generated thus far and contribute freely. To guide the crowd towards useful contributions, Mobi displays automatically generated todo items that alert crowd workers of unresolved constraints. The design takes advantage of the crowd's ability to process context and contribute where they are best able to. It also addresses the crowd's limited attention span by bringing to their attention via todo items where contributions are most needed. Experiments and user studies showed that the design is effective in helping workers to resolve global constraints and that the crowd-generated itineraries satisfied users' stated mission requirements.

In crowdware and Mobi, the crowd is allowed to shape the problem solving process directly. That is, the process of computation is no longer fixed by an algorithm ahead of time and is instead defined by the crowd in the process of problem solving. Expanding on this view, Chapter 4 explored opportunities for involving the crowd in control and synthesis. I presented an experiment on the 8-puzzle that demonstrated

how passing a small amount of context can enable more effective problem solving in an iterative task. I also introduced a system called CrowdPlan, that leveraged a crowd to generate simple plans that help users to approach and accomplish high-level tasks.

Solving a complex problem requires not only effective coordination but recruiting individuals who are willing and able to effectively contribute. Chapter 5 proposed methods for task routing on a social network that harness people's ability to both contribute to a solution and route tasks based on their knowledge of others' expertise. Focusing on prediction tasks, I introduced routing scoring rules that properly incentivize participants to honestly update probability assessments and route tasks to people who they believe can best contribute. Taking into account that individuals may only know about people within a local neighborhood, I identified a family of local routing rules which isolate simple routing decisions in equilibria while still promoting effective information aggregation.

Understanding participants and their behavior is crucial for designing any social or economic Internet system that aims to elicit desired behaviors and outcomes. To enable designers to discover more effective designs more quickly and with less manual effort, the second part of this dissertation focused on constructing automated procedures that discover effective designs by reasoning and learning about participants.

Chapter 6 introduced a general approach for automated environment design. I provided a model of the automated environment design problem and presented an active, indirect elicitation framework that drives an objective-based, iterative design process. As an illustrative example, I introduced the problem of policy teaching, in

which the goal is to discover rewards that induce an agent to follow a desired policy. I showed that, even with a large number of possible designs and little prior information about the agent's reward function, an algorithm based on the active, indirect elicitation framework is guaranteed to discover an effective reward intervention after a small number of interactions.

Chapter 7 presented an approach for automating the design of human computation tasks. Using image labeling as an example, I learned models of the crowd's performance by observing the crowd's outputs under different task designs and used learned models to optimize designs based on desired objectives. Experimental results showed that simple models can accurately predict work quality and that optimized designs outperformed baseline designs at the same rate of pay.

While Chapter 7 focused on the design of human computation tasks with identical, parallel subtasks, Chapter 8 considered the more general challenge of automating the synthesis of workflows that involve heterogeneous tasks. By adapting the active, indirect elicitation framework of automated environment design, I introduced a general framework for automated workflow synthesis. I presented an elicitation strategy that decides which task to experiment on at any given point by estimating the expected value that can be derived from new information. Learned models are used to synthesize and tune algorithms to optimize desired objectives subject to resource constraints. In experiments on human sorting tasks, I showed that this elicitation strategy is effective in helping to discover better algorithms with less experimentation.

9.2 Research Directions

9.2.1 Crowdsourcing and Human Computation

In crowdsourcing complex tasks, there are opportunities to develop other crowdware systems along with theoretical models, in order to fundamentally understand the spectrum between crowdware and workflow paradigms. An interesting challenge is scale. As the number of ideas and the size of the solution grows, it becomes difficult if not impossible for any given individual to keep track of the entire solution context and reason about all aspects of the problem. For problems that are difficult to decompose, managing problem-solving context becomes difficult and crucial for effective problem solving. Problems embodying this challenge include enabling a crowd to write a novel or a large piece of software, and involving hundreds or thousands of individuals in planning real world events and executing their plans.

One idea for overcoming the challenge of scale is to present solution context at different levels of detail and abstraction. We can create *task platforms* that generalize both crowdware and workflow paradigms. By presenting context at the right level of detail, individuals can be prompted to make effective local contributions while being aware of the effect of their actions on the global solution. For example, in writing a story, someone working on the plot may need to be aware of the impact of his contributions on character development, but can otherwise contribute freely. In cases where relevant views of the solution may not already exist, such views may need to be explicitly constructed by the crowd to facilitate effective problem solving. For example, a crowd writing a story may need to produce plot summaries and character

profiles to help people working on different aspects of the problem be aware of relevant changes that require attention and thus be able to contribute more effectively.

From the workflow perspective, task platforms are algorithms that determine and display at any given time a set of available tasks, and for any selection construct an interface that provides the necessary context and functionality for that task. From the crowdware perspective, task platforms consist of multiple workspaces that cover different but interdependent aspects of the problem. There are opportunities to explore both perspectives, and to develop frameworks, methods, and applications that leverage this concept.

Moving from the task level to the organizational level, we can envision a future in which the *distributed intelligence* of humans and machines across networks are brought together to tackle complex problems. In the context of task routing, there are opportunities to develop general principles and methods that effectively and efficiently harness the diverse expertise of participants in a system. As online labor markets and online platforms for collaborative problem solving [2, 84] develop, it will become increasingly important to make efficient use of people's expertise. This includes recognizing people's changing levels of attention, motivation, and availability, and the corresponding need for balancing the load across participants. For example, a task should not always be routed to the individual with the most expertise, simply because that individual may already be engaged in another task. As effective problem solving may rely on the *joint* characteristics of participants involved, I am interested in exploring settings in which *ad hoc teams* [89] of human and machine problem solvers connected through networks form spontaneously to tackle problems as they

arise, and expect that reasoning and learning about the *collective intelligence* [101] of such teams may affect solutions and outcomes.

As we continue to explore crowd problem solving in the context of complex and creative tasks, we will inevitably encounter or create scenarios in which the crowd has an intrinsic interest in the solution. In other words, the outcome of the crowd's collaboration may matter to the crowd, and this creates new opportunities and challenges. One possible issue that can arise is that while some individuals may only be briefly involved, other individuals may return to a task over time and claim particular tasks as their responsibility or make demands about some aspect of the solution. For example, an individual contributing to writing a story may attempt to steer the plot toward a certain direction, and individuals planning a large-scale event may not agree on the best course of action.

One approach for resolving differences in opinion and making key decisions is to consider differentiating members within a crowd, such that some contributors may be given special powers and privileges based on their experience or expertise, and can serve as moderators or decision makers should conflicts arise. This is common on the Web, and is used in social computing systems like Wikipedia and in forums to resolve disputes and maintain the quality of content. In the case of a crowd, a hierarchy among contributors may emerge either organically or based on rules and policies set by the designer. For example, within a task platform, it is possible that some tasks are at a higher level than other tasks (e.g., decision about a key aspect of a plot), with these tasks only accessible to those who have already contributed significantly to other tasks within the platform. Understanding how to design such rules and policies

in order to create and maintain cultural norms through which the best contributors can emerge organically is an interesting area for future work.

An alternative approach is to design affordances that promote effective crowd decision making, but otherwise leave the decision to the crowd. For example, to settle differences, members in the crowd may vote on the best path forward, with the system automatically enforcing and imposing that choice unless the results from a subsequent vote suggests a different path forward. In the context of planning a real-world event, this may mean voting on a course of action and sticking to it unless the crowd collectively prefers something else. The crowd can also decide to split up into smaller crowds, each pursuing their own direction forward. From the computational environment design perspective, understanding how to design effective mechanisms for joint decision making within crowds that promote effective outcomes is an interesting area for future work.

9.2.2 Automated Environment Design

For automated environment design, a key next step is applying the active, indirect elicitation framework to a wide range of real world scenarios in which automation may help to discover more effective designs more quickly and with less manual effort. In the near term, there are opportunities to automate the design of websites and web pages to promote desired usage patterns, for example to increase the levels of contribution, comprehension, and awareness. In the longer term, there may be opportunities to apply automated design techniques in the physical world, where advances in ubiquitous sensing and the increasing digitization of real world spaces have the po-

tential to enable interactions in which spaces can automatically configure themselves to promote desired behaviors and outcomes.

An ongoing challenge for automated environment design will be the availability of models that capture how characteristics of participants interact with the decision environment to influence participant behavior. But as improved models and computational tools for understanding participants from data become available, automated environment design tools and methods will naturally play an increasingly important role in how we approach the design of social and economic systems.

Of course, human ingenuity will also continue to play an important role in designing social and economic systems for many years to come. An interesting direction is to explore opportunities for tight-knit collaboration between humans and machines in the process of identifying an effective design for solving a computational environment design problem. As an example, consider the following interaction. An automated system forms hypotheses and suggests experiments on alternative designs on its own. A designer can at any time ask questions about how the process is going, provide feedback by identifying particular neighborhoods to focus the search, and introduce additional features and parameters for the system to incorporate in its automated design process. The system may likewise provide feedback on a designer's hypotheses and make suggestions based on its knowledge. While the example may seem somewhat futuristic, given the extent to which automated tools for simplifying the design of social and economic systems on the Internet are already utilized and continue to be developed, such interactions may not be so far fetched, and point to a future in which effective collaboration among human and machine designers become commonplace.

9.3 One More Thing

One can envision a future in which a crowd is more connected, more intelligent, and generally more capable of handling a situation or task than an individual. One can also envision a future in which automated systems are more adept at understanding us and shaping the environment around us.

But there is one more thing I want to discuss. It is about why social and economic systems on the Internet exist in the first place.

A computational environment design problem is intrinsically a human problem. It's about designers with their own interests and motivations constructing decision environments in which participants with their own interests and motivations take action. The environment exists to advance the interests of both parties. Otherwise, a designer would likely modify the environment he controls or participants would leave and new environments would likely form.

Given this, what is perhaps most important is for designers to adopt a way of thinking in which truly advancing the interests of both the designer and the participants is paramount over any narrower objective that can be formed. Without regard to this way of thinking, designers may be content with constructing environments that lead to desired behaviors in the short term but that are ultimately unsustainable. For this reason, a designer may need to continuously reassess specific objectives and designs to ensure that they indeed advance the interests of both parties. Such awareness will require that we develop the ability to reason and learn about fundamentally what it is that we as designers aim to do, why is it that we do what we do, and whether doing what we do indeed makes things better.