Automated Workflow Synthesis

Haoqi Zhang
Northwestern EECS
Evanston, IL 60208
hq@northwestern.edu

Eric Horvitz
Microsoft Research
Redmond, WA 98052
horvitz@microsoft.com

David Parkes
Harvard SEAS
Cambridge, MA 02138
parkes@eecs.harvard.edu

Abstract

By coordinating efforts from humans and machines, human computation systems can solve problems that machines cannot tackle alone. A general challenge is to design efficient human computation algorithms or workflows with which to coordinate the work of the crowd. We introduce a method for *automated workflow synthesis* aimed at ideally harnessing human efforts by learning about the crowd's performance on tasks and synthesizing an optimal workflow for solving a problem. We present experimental results for human sorting tasks, which demonstrate both the benefit of understanding and optimizing the structure of workflows based on observations. Results also demonstrate the benefits of using value of information to guide experiments for identifying efficient workflows with fewer experiments.

Introduction

Human computation (von Ahn 2005; Law and von Ahn 2011) centers on models and methods for incorporating humans in problem-solving efforts when machines cannot tackle the problem alone. In the last several years, there has been a rise in *human computation algorithms* or *workflows* that enable a crowd to tackle complex problems by decomposing them into more manageable, self-contained tasks and coordinating contributions across tasks. Workflows are powering crowdsourcing applications for problems such as copy editing (Bernstein et al. 2010), nutrition analysis (Noronha et al. 2011), and article writing (Kittur et al. 2011).

A workflow can draw on a crowd to perform a variety of tasks. For example, a workflow for nutrition analysis may ask the crowd to identify food items in a photograph, describe them in natural language, match descriptions to a food database, and measure portion sizes. Since responses from the crowd are inherently noisy and submitted answers can be incorrect, the quality of task solutions is non-deterministic. A workflow may apply quality control mechanisms that use redundancy or voting to mitigate potential errors in tasks. Such approaches help to mitigate errors in the final solution, but incur costs of additional effort.

A general challenge for human computation is to design workflows that make ideal use of human effort. This involves reasoning about how to decompose a problem into a set of tasks, how much effort to devote to each task, and how to coordinate among the inputs and outputs of the tasks. Making good decisions relies on understanding human performance on specific tasks, and on understanding how potentially noisy and erroneous outputs from each task influence the final solution, either directly or through other tasks that follow in a chain of analysis. The space of possible workflows for solving a problem is large even for a handful of configurable tasks. Thus, learning and reasoning about the efficiency of workflows is a difficult challenge. Designers may test only a small set of designs, make ad-hoc decisions, and ultimately deploy inefficient workflows.

We introduce a general method for *automated workflow synthesis*, centered on constructing models of human performance on tasks to provide guidance on enhancing the efficiency of workflows. Over repeated interactions, an automated system selects experiments to refine current models, with the intent of discovering an efficient workflow constructed of a set of tasks that meets desired objectives and satisfies resource constraints. To reason about the effect of task performance on the overall performance of a workflow, we develop a simulation-based approach that uses learned models to estimate the cost and solution quality associated with a workflow. To synthesize more efficient workflows after fewer experiments, we develop a *value-of-information* elicitation strategy that chooses the next experiment based on which experiment is expected to best inform the choice of the optimal workflow.

We illustrate the effectiveness of the approach in a case study on *human sorting tasks*, in which human judgment is needed to compare the objects to be sorted. We focus on the design of workflows within a class of quicksort algorithms in which *pivot selection* and *pairwise comparison* tasks are performed by the crowd. Experimental results demonstrate that optimized workflows achieve a 13% reduction in errors over baseline comparisons at the same level of effort. Results also show that the value-of-information elicitation strategy reveals better workflows more quickly than selecting experiments that uniformly reduce uncertainty across models. More broadly, the results illustrate how a system can *automatically synthesize efficient workflows by understanding the interplay between workflow structure and human performance on tasks.*

Related Work

Program synthesis considers the use of appropriate design tactics to systematically derive a program based on a problem specification. In the context of sorting, Darlington (1978) and Smith (1985) demonstrated how to derive sorting algorithms using logical transformations and reductions. Closer to our work, Li et al. (2007) demonstrated how to synthesize sorting algorithms that are optimized for particular computer architectures. Because humans can make mistakes, work on automated workflow synthesis must tackle the added challenge of learning and reasoning about human performance on different kinds of tasks.

By synthesizing workflows involving heterogeneous tasks, we extend previous work by Huang et al. (2010) on automatically designing human computation tasks with identical, parallel subtasks. Our work also extends previous work by Shahaf and Horvitz (2010), which introduced formulations for sequencing tasks optimally. By using simulation approaches, we can reason about complex workflows that extend beyond sequencing tasks to cases where the relationship between the crowd's performance on tasks and the quality of the final solution is difficult to capture analytically. Furthermore, while Shahaf and Horvitz focused primarily on the optimization challenge, we address simultaneously the problem of learning about human performance on tasks.

We seek in automated workflow synthesis to efficiently discover workflows that embody effective structures and strategies for solving a problem (e.g., the choice of which tasks to include, the allocation of effort to tasks, and the design of the control flow). In optimizing over a space of possible workflows, our work stands in contrast to previous efforts on optimizing a particular workflow through decision-theoretic optimization (Dai, Mausam, and Weld 2010; 2011; Kamar, Hacker, and Horvitz 2012). Given a predefined workflow, these earlier approaches make online decisions at run time on whether to request additional work based on the current state of problem solving. They focus on making local adjustments and do not consider changes to the larger problem-solving strategy. Furthermore, by reasoning about workflows constructed from a set of tasks, the methods we describe can optimize the interplay among tasks within workflows. Such consideration of the fundamental *structure* of workflows comes in distinction to the work by Lin et al. (2012), which considers optimizing the switching among a fixed set of workflows.

Several studies to date have focused on enabling efficient computation in human-powered database systems that recruit a crowd to perform operations such as filters, sorts, and joins. For example, Marcus et al. (2011b; 2011a) introduces a declarative workflow engine called Qurk, and proposes optimizations for sorts and joins such as batching tasks, using numerical ratings, and pre-filtering tables before joins. Venetis et al. (2012) introduces local search procedures for optimizing workflows for retrieving the maximum item from a set, which we adapt for optimizing sorting workflows. In contrast to these efforts, which focus on optimizing for a specific setting, we present a unifed approach to automated workflow synthesis that simultaneously considers tasks, models, and optimization procedures within an automated experimentation process that can discover efficient workflows for problems automatically.

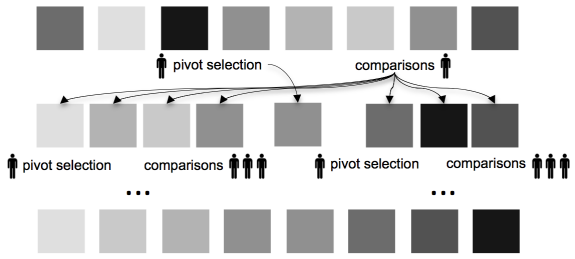

Figure 1: Human quicksort applied to ordering grayscale tiles from light to dark. A workflow specifies the number of workers to allocate to each pairwise comparison and pivot selection task at each level of recursion.

In artificial intelligence, the study of *metareasoning* (Horvitz, Cooper, and Heckerman 1989; Russell and Wefald 1991) seeks to enable agents with bounded time and computational resources to make intelligent decisions about how to reason, what to reason about, how long to deliberate for, and when to take action. Due to the cost of human effort, our system for automated workflow synthesis faces a similar problem in that it must decide which experiments to conduct, how much resources to devote to experimentation, and when to stop experimenting. In using information value to inform elicitation decisions, we adopt a decision-theoretic framework that draws on principles introduced by Horvitz (1987; 1990) for decision-theoretic metareasoning.

Human Sorting Tasks

As an illustrative example, we consider as a case study the problem of finding efficient workflows for *human sorting tasks*. In a human sorting task, human perception and judgment are used to determine the ordering among objects. Examples of human sorting tasks include sorting objects by their visual appeal (Little et al. 2010), ranking edited versions of a paragraph by writing quality (Bernstein et al. 2010), and ranking web pages by their relevance to a query (Alonso, Rose, and Stewart 2008).

Workflows for human sorting tasks may recruit workers from the crowd to perform a variety of tasks, such as comparing pairs of elements, grouping elements into buckets, finding outliers, checking if a list is sorted, or even sorting a small list directly. The effectiveness of a workflow depends on its problem-solving strategy and on how well the crowd performs the tasks it calls upon. Since people can make mistakes, solutions may not be perfectly sorted, and redundancy may be needed to achieve good solutions.

We focus on the example problem of automatically synthesizing a workflow from a class of workflows based on quicksort, using the crowd to perform pairwise comparison and pivot selection operations. We consider how much redundancy to require for each pairwise comparison and pivot selection task that is assigned to the crowd at different points in the computation (see Figure 1). These decisions influence the quality of the solution, as well as the number of operations and thus the cost required to compute a solution.

We consider optimizing over two sets of parameters, r_d and k_d, that determine the number of people to recruit for identifying the median of three randomly chosen elements as the pivot (r_d), and the number of people to recruit for comparing a pair of objects (k_d), at the d-th level of recursion. For any task, the workflow takes the majority answer from people recruited to perform the task as output, breaking ties randomly as needed. In cases where $r_d = 0$, a random element is chosen as the pivot.

The performance of a workflow in this class depends on how well the crowd can identify the median and perform pairwise comparisons, and on the implications of the crowd's performance on the quality of the solution and the cost incurred. We assume that each call to a pairwise comparison or pivot selection task incurs known costs c_k and c_r, which are additive and independent of task input. The quality of the crowd's work is assumed to be unknown a priori, and can only be learned through experimentation.

To evaluate solution quality, we consider two standard measures of sortedness. Given a list $\{l_1, \ldots, l_n\}$ that should be sorted in ascending order, the number of *inversions* is the number of pairwise elements that are out of order, which occurs whenever $l_j < l_i$ for $j > i$. The number of *runs* is the number of adjacent pairwise elements that are out of order, such that $l_{i+1} < l_i$. *The goal is to find parameters values r_d, k_d such that human quicksort based on these values produces solutions with few inversions or runs on average, while staying within a cost budget C.*

Inversions and runs provide global and local measures of sortedness respectively; workflows optimized for the two objectives may be qualitatively different. For example, a workflow optimized for runs may allocate more effort at deeper levels of recursion where more local elements are sorted. In addition to the objective, the crowd's performance on tasks may also affect the choice of workflow. For example, if the crowd makes more mistakes when elements being compared are close in value, shifting effort to deeper levels of recursion where elements being compared are likely to be closer may be a good problem-solving strategy.

Automated Workflow Synthesis

Abstracting from human sorting tasks, we now introduce an automated system to identify efficient workflows for solving a problem. We consider a finite set $S = \{s_1, \ldots, s_m\}$ of base-level tasks that can be used as part of a workflow. An instance of a task represents a single piece of work that can be performed by an individual crowd worker.[1] Tasks differ in the type of work or in the details of how the work is requested, such as the user interface or the instructions.

We assume a worker-independent *task function* f_s for each task $s \in S$. The task function maps an input instance to a probability distribution over the possible responses that workers may provide for the task-input pair, some of which may be incorrect. We assume that each instance of a task incurs a known cost, which may be monetary or be based on a measure of the time or effort required to complete the task instance. We assume that the system does not know how well the crowd performs each task *a priori*. That is, f_s is imprecisely known and more can be learned about this function via experimentation.

The workflow design space considers a (possibly infinite) set of workflows $\mathcal{A} = \{A_1, A_2, \ldots\}$. Workflows differ in the tasks they include, their allocation of effort to tasks, or their control flow. We let $S_i \subseteq S$ denote the set of tasks in A_i. Given a problem instance as input, an *algorithm function* F_i provides a distribution over the possible solutions returned by running workflow A_i. The algorithm function considers workers performing instances of tasks as they are assigned and models the effect of workers' responses on the solution returned by a workflow. In a generative sense, the solution distribution from F_i is constructed by simulating A_i and sampling from the output distribution provided by f_s whenever the workflow calls on an instance of task $s \in S_i$.

An efficient workflow for solving a problem may be tailored to the distribution of problem instances and to the crowd's abilities. To learn about the crowd's performance on tasks, the system can experiment with different tasks and observe the crowd's outputs. At any time, the system can select from a set of possible *experiments* $E = \{e_1, \ldots, e_m\}$, each of which corresponds to a particular task-input pair that can be presented to a crowd worker. Since the crowd's responses are non-deterministic, running the same experiment twice may result in different observations.

Given a distribution over problem instances and a measure of the solution quality, *the system aims to identify a workflow $A^* \in \mathcal{A}$ with the maximum expected solution quality while satisfying cost constraints.*[2] A general goal is to discover, after a small number of experiments, an efficient workflow that obtains high quality solutions and satisfies cost constraints.

An Active, Indirect Elicitation Approach

In order to synthesize workflows that are tailored to the crowd's performance on tasks, we introduce *active, indirect elicitation* (Zhang and Parkes 2008) as a general approach for automated workflow synthesis. For each task $s \in S$, we construct a *task performance model* $\hat{f}_s$ to predict the output from the task function f_s. Proceeding in rounds, an *elicitation strategy* selects which experiments to conduct. An *inference procedure* updates $\hat{f}_s$ based on observed outputs from experiments to refine the system's knowledge of the crowd's task performance. This allows the system to better predict the performance of different workflows under consideration.

At any time during the experimentation process, the system uses its current knowledge to optimize the choice of workflow and to select subsequent experiments. For any task s, we assume that the system can use the current task performance model $\hat{f}_s$ to estimate a distribution over outputs for any input to f_s. Under this assumption, the system can simulate a workflow A_i on a machine by sampling from the output distribution provided by $\hat{f}_s$ whenever the algorithm

[1] For example, on Amazon Mechanical Turk the base-level tasks are the human intelligence tasks (HITs) assigned to workers.

[2] Our approach is agnostic to details of the objective. We can also consider optimizing for more complex preference models that define explicit tradeoffs between solution quality and cost.

makes a call to task $s \in S_i$. This allows the system to estimate a distribution over possible solutions for any workflow applied to a problem instance. Simulations can thus be used to estimate the solution quality and costs for any workflow based on current knowledge. Simulations can also be used to evaluate workflows under different hypotheses about f_s; this may be helpful for choosing an experiment.

Elicitation Strategy: Value of Information

An automated workflow synthesis problem may consider a large space of possible workflows that each draw on diverse tasks. We would like to determine which experiment to conduct at any time that would best improve the choice of workflow. Since the goal is ultimately to discover efficient workflows and not to learn about the crowd's performance on tasks *per se*, it is not necessarily best to learn about task-input pairs on which the fewest experiments have been conducted. For example, if the system already knows that a task is unlikely to help produce high-quality solutions, learning about it is unlikely to improve the choice of workflow.

Following this intuition, we consider a *value-of-information* elicitation strategy that selects experiments based on which task-input pair is most likely to reveal information that improves the choice of the optimal workflow. Let $A^*_{\hat{f}}$ denote the optimal choice of workflow based on current task performance models. That is, $A^*_{\hat{f}}$ achieves the highest average solution quality across all workflows that satisfy cost constraints when simulated using $\hat{f}_s$ for task s on problem instances drawn from a known distribution. For each experiment $e \in E$ that involves the task s_e, let $O_e = \{o_e^1, \ldots, o_e^k\}$ denote the set of potential outcomes from experiment e based on $\hat{f}_{s_e}$. We let $\hat{f}^{o_e^i}$ denote the updated task performance models under the assumption that we conduct experiment e and observe outcome o_e^i, and let $A^*_{\hat{f}^{o_e^i}}$ denote the optimal choice of workflow with respect to $\hat{f}^{o_e^i}$.

The difference in solution quality between $A^*_{\hat{f}}$ and each of the workflows $A^*_{\hat{f}^{o_e^i}}$, evaluated with respect to our knowledge after observing o_e^i, captures the expected value to be gained if we were to update our choice of workflow to deploy after conducting a single experiment e. By comparing experiments in this way, we can find the experiment that myopically maximizes the expected value of information (VOI) by solving the following optimization problem:

$$max_{e \in E} \sum_{o_e^i \in O_e} \Pr(o_e^i | \hat{f}_{s_e}) [v(A^*_{\hat{f}^{o_e^i}} | \hat{f}^{o_e^i}) - v(A^*_{\hat{f}} | \hat{f}^{o_e^i})] \quad (1)$$

$\Pr(o_e^i|\hat{f})$ is an estimate of the likelihood of observing outcome o_e^i when conducting experiment e based on the task performance model $\hat{f}$, and $v(A|\hat{f})$ is a measure of the expected quality of solutions provided by workflow A based on task performance model $\hat{f}$.

Since an individual experiment only obtains a single output from the crowd, it may not contain enough information to change the decision about the best workflow. For this reason, in practice we may wish to conduct experiments in batch. Outside of any computational concerns, the elicitation strategy remains essentially the same, but with each experiment representing a set of experiments.

Beyond informing the system about the best experiments to conduct, VOI can be used to determine whether experimentation should be halted. We halt when the expected value of an experiment is less than its cost. For simplicity, we will not model the cost of experimentation explicitly and focus on discovering efficient algorithms after few experiments.

Case Study on Human Quicksort

Having introduced a general approach for automated workflow synthesis, we return to our case study on human sorting tasks and demonstrate how to synthesize human quicksort workflows that are tailored to the crowd's abilities.

Task Performance Models

We first construct models for the pairwise comparison and pivot selection tasks. Since it is infeasible to learn a model for every combination of input values separately, we consider grouping sets of input values into *clusters*, and learning a model for each task-cluster pair. In the simplest instantiation, there may only be a single cluster per task, and the model may only attempt to learn an input-independent probability distribution over outputs. We can consider arbitrarily more complex models by considering finer-grained clusters.

For each model, we use the Beta and Dirichlet distributions to represent our knowledge and uncertainty over the actual output distributions for pairwise comparison and pivot selection tasks, respectively. For example, for pairwise comparisons, a Beta distribution's α and β parameters can capture the number of correct and incorrect answers respectively, and be updated by incrementing α by 1 if the answer to an experiment is correct or incrementing β by 1 otherwise. For pivot selections, a Dirichlet distribution with three parameters maintains counts over the frequency of the correct output and the two possible incorrect outputs, and can be similarly updated. These distributions are reasonable representations of uncertainty as sampling processes progress and have conjugate properties that allow for efficient updating with observations (Howard 1970).

Optimizing Human Quicksort

For the human quicksort workflows we consider, the number of possible workflows is exponential in the assignment of effort to tasks at different levels of recursion. For tractability, we use local search to find workflows that are approximately optimal with respect to task performance models. We do this by adapting to our setting the local search procedures introduced by Venetis et al. (2012) for optimizing workflows for finding the maximum element in a set.

To perform the search, we assume that there is a fixed, finite set of possible values to assign to parameters r_d and k_d. Given task performance models, we first compute the optimal *constant sequence* workflow, which selects fixed values for r^* and d^* such that $r_d = r^*$ and $k_d = k^*$ for all recursion levels d. Better workflows may exist that allocate effort non-uniformly at different levels of recursion. For simplicity, we fix the number of people to assign to pivot selection

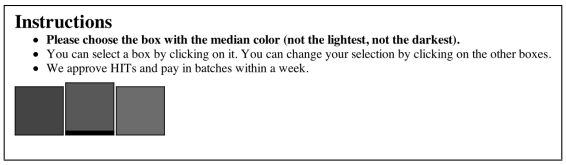

Figure 2: Example HIT for the pivot selection task.

tasks (r_d) and consider a greedy hill-climbing procedure that iteratively varies the number of people to assign to pairwise comparison tasks (k_d). For every pair of parameter values k_i and k_j for which $k_i > 1$, we consider the effect of decrementing k_i and incrementing k_j up to the point that the resulting workflow *just* satisfies cost constraints. Given that swaps improve the solution quality, we apply the best such swap, and repeat the process to incrementally improve the choice of workflow until no such improvements exist.

Experimental Setup

We consider a human sorting task that requires people to sort a list of grayscale tiles from light to dark. We choose this example because comparisons are objective, tasks are easy to describe, and tasks may vary in difficulty (e.g., based on the closeness of tiles' grayscale values). The straightforward nature of the task makes it easy for us to evaluate answers, and allows for exploration of models that depend on characteristics of specific task instances. To study human performance on tasks in this domain, we recruited workers from Amazon Mechanical Turk (Turkers) to complete pairwise comparison and median-of-three pivot selection tasks (see Figure 2).[3]

To simplify the evaluation, we use the Turkers' responses to construct *ground truth models* of task functions. Each ground truth model maintains a distribution over answers based on the empirically observed answers from Turkers. When evaluating the active, indirect elicitation approach, instead of actually posting jobs on Mechanical Turk for experiments that an elicitation strategy chooses, we instead sample from the ground truth distribution to simulate the answers that the crowd would provide in an experiment. If the models are accurate, results of the simulated experiments would approximate the crowd's actual performance, but with evidence obtained *a priori* to allow for a simpler evaluation.

For both the ground truth and task performance models, we cluster inputs to tasks based on the closeness of objects being compared. For pairwise comparison tasks, we consider clusters based on the distance in grayscale value between pairs of objects. For pivot selection tasks, we consider clusters based on the *minimum* distance between the grayscale value of the median element and a non-median element. We consider five clusters per task, for distances of 1, 2, 3, 4, and 5+.

[3]For pairwise comparison tasks, we posted 100 HITs each with 10 assignments. We sampled pairs of grayscale values at random, restricting the difference in grayscale values to between 1 and 10. We used a scale with 128 values, which we chose so that minimal differences in darkness are barely distinguishable. For pivot selection tasks, we also posted 100 HITs each with 10 assignments. We sampled three grayscale values for tiles at random, restricting the difference in value between the median element and the other 2 elements to between 1 and 10. Workers were required to have a 98% approval rating, and were paid $0.01 per HIT.

	Pairwise	Pivot	
Difference	Pr(correct)	Pr(median)	Pr(closer)
1	0.74	0.585	0.27
2	0.87	0.69	0.16
3	0.94	0.74	0.13
4	0.98	0.82	0.10
$\geq$5	0.997	0.85	0.08

Table 1: Ground truth models based on Turkers' performance on pairwise comparison and pivot selection tasks as a function of the (minimum) difference in grayscale value.

We maintain a model for each task-cluster pair. For pivot selection, the Dirichlet models maintain counts for selecting the median (α_1), the element closer to the median (α_2), and the element farther from the median (α_3). As priors, we initialize all pairwise comparison models with $(\alpha, \beta) = (4, 1)$, and all pivot selection models with $(\alpha_1, \alpha_2, \alpha_3) = (6, 1, 1)$.

In the active, indirect elicitation process, the set of possible experiments includes an experiment for each task-cluster pair. We batch experiments to sets of five observations each, such that any update to a model is based on five samples drawn from the ground truth distribution. We compare the VOI elicitation strategy with a *uniform* strategy that chooses the next experiment based on the model that has been experimented on the fewest times thus far. We hypothesize that learning will typically lead to better workflows regardless of elicitation strategy, but that the VOI strategy will identify better workflows after fewer experiments.

We consider optimizing with respect to random permutations of a list with 20 tiles holding grayscale values 1 through 20, with costs $c_k = c_r = 1$ and budget $C = 250$. We consider $k_d \in \{1, 3, 5, 7\}$ and $r_d = r \in \{0, 1, 3\}$ as possible values for k_d and r_d, where $d \in \{1, 2, 3, 4, 5, 6+\}$.

Results

From the Mechanical Turk experiment, we found that people make more mistakes in pairwise comparison tasks when tiles are closer in value (see Table 1). We observe that when tiles differ in value by 1, the crowd makes twice as many mistakes (26% error rate) as when tiles differ in value by 2 (13%), and about four times as many mistakes as when tiles differ in value by 3 (6%). For pivot selection tasks, we found that people make more mistakes when one or more of the non-median elements is close to the median.

Table 2 shows the average configuration and solution quality of workflows synthesized for minimizing runs and inversions. To compare workflows synthesized before and after learning, we optimize with respect to both the prior distributions and the ground truth distributions. The best constant sequence workflow serves as a baseline for what a designer may deploy without reasoning about the specifics of the objective or learning about human performance on tasks.

When minimizing runs, we observe that both prior to learning and with knowledge of the ground truth, synthesized workflows outperform the baseline by allocating more

	k_1	k_2	k_3	k_4	k_5	k_{6+}	r	error
Runs								
Baseline	3	3	3	3	3	3	3	3.24
Prior distribution	1.9	3.4	4.2	4.3	4.5	4.0	2.3	2.97
Ground truth	2.0	3.3	4.6	5.3	5.1	4.9	1.3	2.74
Inversions								
Baseline	3	3	3	3	3	3	3	4.23
Prior distribution	3.5	3.0	3.0	3.0	4.3	3.8	2.9	4.21
Ground truth	3.0	3.3	3.8	4.3	4.3	4.4	1.4	3.66

Table 2: Average configuration and solution quality of optimized workflows. The best constant sequence workflow serves as a baseline. Values are averaged over 500 trials.

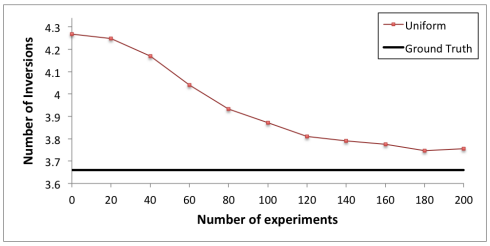

Figure 3: Performance of workflows optimized through the process of learning. Values are averaged over 500 trials.

effort to pairwise comparisons at deeper levels of recursion. Since in quicksort nearby elements are sorted at these deeper levels, optimizing for runs shifts effort to these levels even when errors are assumed to be uniform. We observe that, with knowledge of the ground truth, even more effort is spent on pairwise comparisons at deeper levels to account for people making more errors when elements are close.

When minimizing inversions, optimization does not lead to effective workflows in the absence of accurate error models. Assuming that errors are uniform, workflows synthesized prior to learning allocate slightly more effort upfront in order to avoid large errors that may result in multiple inversions. But since errors are most likely to occur when elements being compared are close at deeper levels of recursion, that policy turns out to be a poor strategy. We observe that workflows synthesized based on ground truth models allocate more effort at deeper levels (but not as much as for minimizing runs), which results in 13% fewer inversions.

These results show that understandings of a workflow's structure and human performance on tasks jointly contribute to synthesizing better workflows. For runs, knowing that quicksort compares closer items at deeper levels of recursion led to an improved workflow even prior to learning. For inversions, recognizing that people make more errors when items are close was crucial for finding an effective workflow.

Experiments reduce uncertainty in task performance models and inform decisions for synthesizing workflows. Focusing on the influence of learning on minimizing inversions, Figure 3 shows that workflows optimized using current task performance models improve over time as the system conducts more experiments using the uniform strategy.

Figure 4 compares the average number of inversions when using workflows that are optimized based on updated models following each experiment for the VOI and uniform strategies. We observe that after the early rounds of learning, workflows optimized based on information gathered using the VOI strategy outperform workflows optimized based on information gathered using the uniform strategy. Comparing the best workflows from those synthesized after each experiment conducted thus far, Figure 5 shows that the VOI strategy discovers, after fewer experiments, workflows that lead to fewer inversions.[4] The difference is most significant before all of the models become refined, suggesting that the VOI strategy may be particularly useful when models are complex and experimentation is costly.

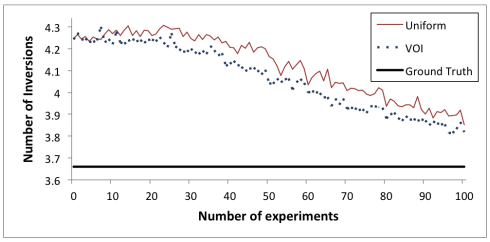

Figure 4: Comparison of the performance of workflows optimized following each experiment for the VOI and uniform elicitation strategies. Values are averaged over 500 trials.

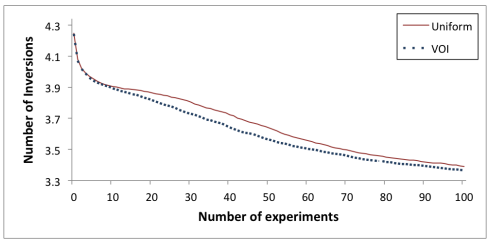

Figure 5: Comparison of the performance of the best workflows discovered thus far for the VOI and uniform elicitation strategies. Values are averaged over 500 trials.

Conclusion

We introduced a general methodology for automated workflow synthesis. Experimental results showed that reasoning about workflow structure and learning about human task performance can lead to more efficient workflows for human computation. Results also highlighted the benefits of using VOI to discover efficient workflows after fewer experiments.

Research directions on automated workflow synthesis include developing procedures that optimize workflows over richer design spaces composed of larger sets of tasks and more varied control structures. The design space may also involve decisions about how machine competencies can be harnessed to complement human work. Finally, we are interested in synthesizing workflows that include components that perform decision-theoretic control at one or more levels of analysis. Such an extension will require procedures for learning and reasoning about high-level problem-solving strategy and finer-grained control structures simultaneously.

[4]Because the local search optimization may search over different sets of workflows for different models, we observe in Figure 5 that the best workflows discovered through learning can sometimes outperform the workflows synthesized based on the ground truth.

References

Alonso, O.; Rose, D. E.; and Stewart, B. 2008. Crowdsourcing for relevance evaluation. *SIGIR Forum* 42(2):9–15.

Bernstein, M. S.; Little, G.; Miller, R. C.; Hartmann, B.; Ackerman, M. S.; Karger, D. R.; Crowell, D.; and Panovich, K. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, 313–322. ACM.

Dai, P.; Mausam; and Weld, D. S. 2010. Decision-theoretic control of crowd-sourced workflows. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, AAAI '10, 1168–1174.

Dai, P.; Mausam; and Weld, D. S. 2011. Artificial intelligence for artificial artificial intelligence. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, AAAI '11, 1153–1159.

Darlington, J. 1978. A synthesis of several sorting algorithms. *Acta Informatica* 11:1–30. 10.1007/BF00264597.

Horvitz, E.; Cooper, G.; and Heckerman, D. 1989. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, IJCAI '89, 1121–1127.

Horvitz, E. J. 1987. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence*, 429–444.

Horvitz, E. J. 1990. *Computation and Action Under Bounded Resources*. Dissertation, Stanford.

Howard, R. 1970. Decision analysis: Perspectives on inference, decision, and experimentation. *Proceedings of the IEEE* 58(5):632–643.

Huang, E.; Zhang, H.; Parkes, D. C.; Gajos, K.; and Chen, Y. 2010. Toward automatic task design: A progress report. In *Proceedings of the 2nd Human Computation Workshop*, HCOMP '10.

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. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Kittur, A.; Smus, B.; Kraut, R.; and Khamkar, S. 2011. Crowdforge: Crowdsourcing complex work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11.

Law, E., and von Ahn, L. 2011. *Human Computation*. Morgan & Claypool Publishers.

Li, X.; Garzaran, M. J.; and Padua, D. 2007. Optimizing sorting with machine learning algorithms. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, IPDPS '07.

Lin, C.; Mausam; and Weld, D. 2012. Dynamically switching between synergistic workflows for crowdsourcing. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, AAAI '12.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on user interface software and technology*, UIST '10, 57–66. New York, NY, USA: ACM.

Marcus, A.; Wu, E.; Karger, D.; Madden, S.; and Miller, R. 2011a. Human-powered sorts and joins. *Proceedings of the VLDB Endowment* 5(1):13–24.

Marcus, A.; Wu, E.; Karger, D. R.; Madden, S.; and Miller, R. C. 2011b. Crowdsourced databases: Query processing with people. In *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research*, CIDR '11.

Noronha, J.; Hysen, E.; Zhang, H.; and Gajos, K. Z. 2011. Platemate: Crowdsourcing nutrition analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, 1–12.

Russell, S., and Wefald, E. 1991. Principles of metareasoning. *Artificial Intelligence* 49(1-3):361–395.

Shahaf, D., and Horvitz, E. 2010. Generalized task markets for human and machine computation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, AAAI '10, 986–993.

Smith, D. R. 1985. Top-down synthesis of divide-and-conquer algorithms. *Artificial Intelligence* 27,1:43–96.

Venetis, P.; Garcia-Molina, H.; Huang, K.; and Polyzotis, N. 2012. Max algorithms in crowdsourcing environments. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, 989–998. New York, NY, USA: ACM.

von Ahn, L. 2005. *Human Computation*. Ph.D. Dissertation, Carnegie Mellon University.

Zhang, H., and Parkes, D. C. 2008. Value-based policy teaching with active indirect elicitation. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, AAAI '08, 208–214.