Efficient Exploration of the Crowd Process Design Space

De Boer, Patrick ; Bernstein, Abraham

Efficient Exploration of the Crowd Process Design Space

PATRICK M. DE BOER, University of Zurich
ABRAHAM BERNSTEIN, University of Zurich

Crowd sourcing is increasingly adopted as an approach to tap into the skillset of a vast amount of workers that are available on demand. In paid crowd sourcing portals, workers flock to fulfil tasks of requesters. Tasks include data labelling, object tagging, as well as more complex tasks, where crowd workers need to collaborate. As is common in any work environment involving collaborating people, workers need to be coordinated to complete tasks. Numerous patterns have been proposed on how to structure tasks, such that workers can solve them in the "best" way possible, whereas "best" can be measured by a utility function defined by the requestor. Popular patterns towards this goal include Majority voting, Iterative Refinement [Little et al. 2010], and Find Fix Verify [Bernstein et al. 2010].

Unfortunately, the performance of these patterns varies by problem setting and it is hard to know ex-ante how well a pattern will work for a given problem. Due to the vast number of patterns available and the expenses of implementing (and evaluating) crowd processes for them, picking the wrong one may prove costly.

To this end, we proposed PPLib [de Boer and Bernstein 2016], a method and Open Source system,[1] that, given a problem definition, leverages process recombination [Bernstein et al. 1999] to automatically generate candidate crowd processes for it. Provided with a user-defined utility function, PPLib can also evaluate the performance of these candidates and find a suitable process for the provided problem definition.

In order to evaluate a candidate crowd process using utility function, previous versions of PPLib included a naïve Auto-Experimentation Engine, which executed each candidate crowd process[2] (multiple times) and let the user-defined utility function assess its performance. However, if many candidate crowd processes fit a problem definition – and therefore need to be executed for their evaluation – finding a well-performing process could become prohibitively expensive and would only pay off in large crowd sourcing projects, where potential cost savings from highly optimized crowd processes balance the expenses for finding them.

The core idea of this abstract is to reframe the problem of identifying well-performing crowd processes for a given problem from a set of candidates *as a convex optimization problem*. Specifically, given the user-defined, performance-measuring utility function, optimization can be employed to find the highest value of this utility function, whilst minimizing the number of costly samples (i.e., crowd-process executions) necessary.

For this abstract, we developed and evaluated a prototype implementing Bayesian Optimization [Mockus 2012] to guide PPLib's Auto-Experimentation Engine. Thus, the main contribution of this extended abstract is the presentation of a cost-efficient Bayesian Optimization guided Auto-Experimentation method (BOA) to identify well-performing crowd processes for a problem at hand.

Using a simulated crowd sourcing setting, we find that a BOA setup can *lower the cost of finding one of the best performing crowd processes by ~100x in 87% of the cases*, whilst finding a good one in the other cases.

[1] https://github.com/uzh/PPLib

[2] In order to evaluate crowd processes, they are executed on live crowd sourcing platforms (such as Amazon Mechanical Turk) on a sample task in the specified problem definition. For example, if the goal is to find a well-performing crowd process to translate a book, a sample task could be to translate a single paragraph of that book.

EVALUATION

In order to be able to compare results between PPLib's naïve Auto-Experimentation and the new BOA, we evaluated both methods using a simulation,[3] where crowds are asked multiple choice questions with different levels of bias inherent their possible answers. Specifically, we wanted to ascertain, if the Bayesian-guidance can identify well-performing crowd processes with significantly fewer samples (i.e., lower cost) compared to the naïve Auto-Experimentation Engine proposed earlier.

1.1 Experimental setup

Mirroring many crowd computing settings, we chose a simulation that included a set of multiple choice questions Q, where each question $q \in Q$ had a set of possible choices C_q. The simulated crowd workers would pick each choice $c_q \in C_q$ with a prior probability $p(c_q)$, whereas the goal of the crowd process was to reliably identify the choice with the highest prior for each question.

To model different degrees of bias in the crowd, we evaluated our prototype with 4 different questions, each having different distributions of prior probabilities (see Table 1), where each succeeding question simulated a less unanimous case. For example, the 1st-3rd options (c_1 -c_3) of Question Q1 have a 1% probability of being picked by a simulated crowd worker, while the forth option has a 97% chance of being picked, making Q1 simulate the almost unanimous condition.

Experiment	$p(c_1)$	$p(c_2)$	$p(c_3)$	$p(c_4)$
Q1	1%	1%	1%	97%
Q2	10%	10%	10%	70%
Q3	20%	20%	20%	40%
Q4	24%	24%	24%	28%

Table 1: Prior Probabilities for options to be selected in the different experiments

A set of executable candidate crowd processes for each experimental condition (Q1-Q4) can be obtained semi-automatically through PPLib's Recombination mechanism: Following the approach proposed in [de Boer and Bernstein 2016], a process designer starts by defining the crowd process search space through basic operators describing the problem deep structure [Chomsky 1965] (i.e. the most abstract representation of the problem in terms of its major building blocks). In this case, each experiment condition only involves obtaining the answer to a single multiple choice question. This decision can be modelled as a single DECIDE-type operator in PPLib. Using this search space definition, PPLib then recombines suitable candidate processes using it's repository of crowd process fragments. For each experiment condition (Q1-Q4), 208 candidate crowd processes have been generated.

In order to evaluate each candidate crowd process' performance, it needs to be executed and its results assessed using a utility function that is able to quantify a crowd process designer's key requirements. In this case, we defined our utility function to reward (i) if the question answer selected by the process was the "correct" one (i.e. the one with the highest prior in the respective experimental condition) and (ii) low process execution cost. Specifically, we used the utility function $\lambda(c_s, k_s) = -(\max_i p(c_i) - p(c_s) + k_s)$, where c_s is the multiple choice item selected by the crowd after executing the process and k_s is the total execution cost of the candidate process. Hence, the more the selected answer's probability deviates from the answer with the highest prior, the larger the penalty induced by the first term. The same applies to the cost of a process.

[3] We chose to run a simulation rather than a field experiment due to the overwhelming cost that would be involved when employing both search algorithms on a paid platform. Specifically, at least $100 \cdot 208 \cdot 5$ crowd processes would have to be executed to draw 100 samples from the naïve Auto-Experimentation Engine with the default of 5 repetitions for stability.

1.2 Results

The naïve Auto-Experimentation Engine applied this utility function to all 208 generated candidate crowd processes, while we configured BOA's convergence criterion as the inability to find a new optimal observed λ for 20 iterations. The best-performing crowd processes are then identified as the most stable crowd process among the candidates leading to the 5% lowest values of λ.

In order to assess the significance of the cost difference and a possible quality trade off, both, the naïve and the optimized Auto-Execution Engine are executed and compared after drawing 100 samples in experimental condition Q1-Q4. We established a ground truth for the quality of each candidate process by re-running it 1000x in each setting and ranking it according to its average utility λ and its standard deviation thereof (lower is better).

When running the experiment, we discovered, that the BOA *costs ~100x less* (on average) to find crowd processes performing well in each experimentation setting (across Q1-Q4 μ=117.9x less, σ=30.5). In 87% of the cases, BOA nominated a process, which was part of the top-10% best-performing processes as identified by the ground truth.

We also found that *each experimental-condition* (Q1-Q4) *yielded different optimal processes*), which emphasizes the need for systematic crowd process design in practice and confirms a similar finding using real-world data in [de Boer and Bernstein 2016].

A t-test comparing the experimental conditions additionally shows, that each experimental condition results in different Auto-experimentation costs ($P < 0.001$). Conditions with a small difference in priors (Q3,Q4) are generally a lot more expensive than the conditions with high difference thereof (Q1,Q2). In addition to a higher cost in Auto-experimentation, the nominated crowd processes are also more expensive. This makes sense intuitively: The more ambiguous the correct answer to a question, the lower the share of crowd workers answering correctly – therefore requiring more work to obtain the correct answer.

LIMITATIONS

The experiment has focused on a setting, where it is assumed that the majority of crowd workers know the "correct" answer to a question. This assumption does not generalize in practice. However, we didn't aim to provide a solution to this interesting problem; instead, our main goal was to the validate our method for crowd process selection in a common, simple crowd task.

Another important point is to evaluate BOA in a real-world setting, with actual crowd workers, rather than a simulation. At the time of writing, preliminary real-world results indicate that the performance is indeed comparable to what we have observed in the simulation.

CONCLUSION

This abstract proposes a method and prototype to efficiently explore the design space of crowd processes for a problem at hand using Bayesian optimization. We showed in a simulation, that our prototype is capable of finding well-performing processes in a fraction of the cost of the current state-of-the art (~100x less costly).

Our findings provide first evidence that Bayesian Optimization guided Auto-Experimentation makes the exploration of a large design space for crowd processes feasible. This may pave the way for better designed crowd processes in the future.

REFERENCES

Abraham Bernstein, Mark Klein, and Thomas W. Malone. 1999. The process recombinator: a tool for generating new business process ideas. In *ICIS '99 Proceedings of the 20th international conference on Information*. 178–192.

Michael S. Bernstein et al. 2010. Soylent : A Word Processor with a Crowd Inside. In *UIST*.

Patrick M. de Boer and Abraham Bernstein. 2016. PPLib: Towards the Automated Generation of Crowd Computing Programs using Process Recombination and Auto-Experimentation. *ACM Trans. Intell. Syst. Technol.* , Special Issue: Crowd in Intelligent Systems (2016).

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*,

Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 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. DOI:http://dx.doi.org/10.1145/1866029.1866040

Jonas Mockus. 2012. *Bayesian approach to global optimization: theory and applications* Vol. 37. S., Kluwer Academic Publishers.