Cost and Quality Assurance in Crowdsourcing Workflows

Loïc Hélouët, Zoltan Miklos, Rituraj Singh
loic.helouet@inria.fr,zoltan.miklos,rituraj.singh@irisa.fr

Despite recent advances in artificial intelligence and machine learning, many tasks still require human contributions. With the growing availability of Internet, it is now possible to hire workers on crowdsourcing marketplaces. Many crowdsourcing platforms have emerged in the last decade: Amazon Mechanical Turk, Figure Eight[2], Wirk[3], etc. A platform allows employers to post tasks, that are then realized by workers hired from the crowd in exchange for some incentives [3, 19]. Common tasks include image annotation, surveys, classification, recommendation, sentiment analysis, etc. [7]. The existing platforms support simple, repetitive and independent *micro-tasks* which require a few minutes to an hour to complete.

However, many real-world problems are not simple micro-tasks, but rather complex orchestrations of dependent tasks, that process input data and collect human expertize. Existing platforms provide interfaces to post micro-tasks to a crowd, but cannot handle complex tasks. The next stage of crowdsourcing is to build systems to specify and execute complex tasks over existing crowd platforms. A natural solution is to use workflows, i.e., orchestrations of phases that exchange data to achieve a final objective. Figure 1 is an example of complex workflow depicting the image annotation process on SPIPOLL [5], a platform to survey populations of pollinating insects. Contributors take pictures of insects that are then classified by crowdworkers. Pictures are grouped in a dataset D_{in}, input to node p_0 . D_{in} is filtered to eliminate bad pictures (fuzzy, blurred,...) in phase p_0. The remaining pictures are sent to workers who try to classify them. If classification is too difficult, the image is sent to an expert. Initial classification is represented by phase p_1 in the workflow, and expert classification by p_2. Pictures that were discarded, classified easily or studied by experts are then assembled in a result dataset D_{out} in phase p_f, to do statistics on insect populations.

Workflows alone are not sufficient to crowdsource complex tasks. Many data-centric applications come with budget and quality constraints: As human workers are prone to errors, one has to hire several workers to aggregate a final answer with sufficient confidence. An unlimited budget allows hiring large pools of workers to assemble reliable answers for each micro-task, but in general, a client for a complex task has a limited budget. This forces to replicate micro-tasks in an optimal way to achieve the best possible quality, but without exhausting the given budget. The objective is hence to obtain a *reliable* result, forged through a *complex orchestration*, at a *reasonable cost*.

Several works consider data centric models, deployment on crowdsourcing platforms, and aggregation techniques to improve data quality (see [11] for a more complete bibliography). First, coordination of tasks has been considered in languages such as BPMN [18], BPEL [17], or workflow nets [23], a variant of Petri nets dedicated to business processes. They allow parallel or sequential execution of tasks, fork and join operations to create or merge a finite number of parallel threads. Some works propose empirical solutions for complex data acquisition, mainly at the level of micro-tasks [7, 15]. Crowdforge uses Map-Reduce techniques to solve complex tasks [13]. Turkit [16] builds on an imperative language, that allows for repeated calls to services provided by a crowdsourcing platform. Turkomatic [14] implements a Price, Divide and Solve loop, that asks crowd workers to split task into orchestrations of subtasks, and repeats this operation up to the level of micro-tasks.

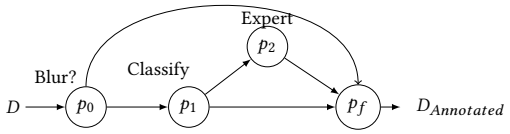

Figure 1: A workflow from SPIPOLL

In this work, we assemble answers returned by workers with so-called *aggregation techniques*. The simplest aggregation is majority voting (MV), a mechanism that considers the most returned answer as ground truth. MV can be improved by giving more weight to competent workers. Other approaches use Expectation Maximization (EM), and consider workers competences to synthesize the most probable correct answer. Competences are expressed in terms of *accuracy* (ratio of correct answers) or in terms of *recall* and *specificity* (that considers correct classification for each possible type of answer). It is usually admitted [26] that recall and specificity give a finer picture of worker's competence than accuracy. Zencrowd [6] uses EM to aggregate answers, and defines competences via accuracy. Workers accuracy and ground truth are hidden variables that must be discovered in order to minimize the deviations between workers answers and aggregated conclusion. D&S [4] uses EM to synthesize answers that minimize error rates from a set of patient records. It considers recall and specificity, but not the difficulty of tasks. [12] proposes an algorithm to assign tasks to workers, synthesize answers, and reduce the cost of crowdsourcing. It assumes that all tasks have the same difficulty, and that workers reliability is a static probability to return a correct value (i.e., the ground truth) that applies to all types of tasks. EM is used by [20] to discover recall and specificity of workers, discover the best experts, and estimate the ground truth. Most of the works cited above consider expertise of workers but do not address tasks difficulty. Approaches such as GLAD [25] or [2] also estimate tasks difficulty to improve quality of answers aggregation on a single batch of Boolean tagging tasks.

A few papers on data aggregation focus on costs optimization. CrowdBudget [22] is an approach that divides a budget B among K existing tasks to replicate them and then aggregate answers with MV. Crowdinc [21] is an EM-based aggregation technique that considers task difficulty, recall and specificity of workers to realize a single batch of micro tasks with a good trade-off between costs and

data quality. It computes accuracy of an aggregation, and launches new tasks dynamically.

Some works consider deployment of tasks, i.e., synthesis of strategies to hire workers and parallelize realization of batches of tasks. The objective is to improve costs and latency, i.e., the time needed to treat a complete batch with an optimal deployment. CLAMSHELL [10] focuses on latency improvement. It affects workers to batches of tagging tasks and detects staggers. To speed up tasks completion, some batches are replicated. Pools are assembled and maintained by rewarding workers for waiting. This approach improves latency, but increases costs. [8] proposes to compute the best static deployment policies in order to achieve an optimal utility (i.e., a weighted sum of overall cost and accuracy) using sequencing or parallelization of tasks. This approach uses a costly exhaustive search which limits the size of deployment problems that can be considered. [24] is a recommendation technique for deployments, that allows parallelization of tasks, sequential composition, and use of machines to solve open tasks such as translation or text writing. This approach builds on optimization techniques to find deployments that reduces latency and improves quality of data.

In this work we propose a solution for the efficient realization of complex tasks. We use a workflow model to orchestrate complex tasks, replicate/distribute them and aggregate the returned results before passing the forged dataset to the next tasks. We extend the complex workflow model of [1], and use the aggregation technique of Crowdinc [21] to forge reliable answers. Our workflow model orchestrates tasks and work distribution according to a dynamic policy that considers confidence in aggregated data and the cost to increase this confidence. A workflow can be seen as an orchestration of phases, where the goal of each phase is to tag records from its input dataset. The output of a phase is used as input for the next ones in the workflow. A complex task terminates when the last of its phases has completed its tagging. For simplicity, we consider simple Boolean tagging tasks that associate a tag in $\{0, 1\}$ to every record in a dataset. Each tagging task on each record is performed by several workers to reduce errors, and the answers are assembled using an aggregation technique. We assume that workers are uniformly paid. For each record, one of the possible answers (the *ground truth*) is correct, and an aggregated answer is considered as reliable if its probability to be the ground truth (computed with aggregation) is high. Hiring more workers to tag records increases the reliability of the aggregated answer. The overall challenge is hence to realize a workflow within a given budget B_0, while guaranteeing that the final dataset forged during the last phase of the workflow has a high probability to be the ground truth.

Design choices influence realization and quality of workflows realization. First, the chosen aggregation technique influences the quality of the final results. Furthermore, the mechanisms used to hire workers impacts costs and accuracy of answers. The simplest way to replicate micro-tasks is *static execution*, i.e., affect an identical fixed number of workers to each micro-task in the orchestration without exceeding budget B_0. On the other hand, one can allocate workers to tasks *dynamically*. One can wait in each phase to achieve a sufficient reliability of answers for all records of the input before forwarding data. This is called a *synchronous* execution of a workflow. Last, one can eagerly forward records with reliable tags to the next phases without waiting for the total completion of a phase. This is called an *asynchronous* execution.

We then study execution strategies for complex workflows in different contexts. We consider several types of workflows, different aggregation mechanisms (namely Majority Voting (MV) and Expectation Maximization (EM) [9]), several distributions of data, difficulty of tasks and workers expertize. We evaluate the cost and accuracy of workflows execution in these contexts under static, synchronous and asynchronous assignment of workers to tasks. Unsurprisingly, dynamic distribution of work saves costs in all cases. A more surprising result is that synchronous realization of complex tasks is in general more efficient than asynchronous realization.

REFERENCES

[1] P. Bourhis, L. Hélouët, Z. Miklos, and R. Singh. Data centric workflows for crowdsourcing. In *Proc. of Petri Nets 2020*, pages 46–61, 2020.

[2] P. Dai, C. H. Lin, and D. S. Weld. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202:52–85, 2013.

[3] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Comput. Surv.*, 51(1):7, 2018.

[4] A.Ph. Dawid and A.M. Skene. Maximum likelihood estimation of observer errorrates using the em algorithm. *J. of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

[5] N. Deguines, R. Julliard, M. De Flores, and C. Fontaine. The whereabouts of flower visitors: contrasting land-use preferences revealed by a country-wide survey based on citizen science. *PloS one*, 7(9):e45822, 2012.

[6] G. Demartini, D.E. Difallah, and Ph. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proc. of WWW 2012*, pages 469–478. ACM, 2012.

[7] H. Garcia-Molina, M. Joglekar, A. Marcus, A. Parameswaran, and V. Verroios. Challenges in data crowdsourcing. *Trans. on Knowledge and Data Engineering*, 28(4):901–911, 2016.

[8] T. Goto, S.and Ishida and D. Lin. Understanding crowdsourcing workflow: Modeling and optimizing iterative and parallel processes. In *Proc. of HCOMP 2016*, pages 52–58. AAAI Press, 2016.

[9] M.R. Gupta and Y. Chen. Theory and use of the em algorithm. *Foundations and Trends in Signal Processing*, 4(3):223–296, 2011.

[10] D. Haas, J. Wang, E. Wu, and M.J. Franklin. Clamshell: Speeding up crowds for low-latency data labeling. *Proc. VLDB Endow.*, 9(4):372–383, 2015.

[11] L. Hélouët, Z. Miklos, and R. Singh. Cost and Quality Assurance in Crowdsourcing Workflows. Extended Version, October 2020.

[12] D.R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *Proc. of NIPS'11*, pages 1953–1961, 2011.

[13] A. Kittur, B. Smus, S. Khamkar, and R.E. Kraut. Crowdforge: Crowdsourcing complex work. In *Proc. of UIST'11*, pages 43–52. ACM, 2011.

[14] A. Kulkarni, M. Can, and B. Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *Proc. of CSCW'12*, pages 1003–1012. ACM, 2012.

[15] G. Li, J. Wang, Y. Zheng, and M.J. Franklin. Crowdsourced data management: A survey. *Trans. on Knowledge and Data Engineering*, 28(9):2296–2319, 2016.

[16] G. Little, L.B. Chilton, M. Goldman, and R.C. Miller. Turkit: tools for iterative tasks on Mechanical Turk. In *Proc. of HCOMP'09*, pages 29–30. ACM, 2009.

[17] OASIS. *Web Services Business Process Execution Language*. 2007.

[18] OMG. *Business Process Model and Notation (BPMN)*. OMG, 2011.

[19] A.J. Quinn and B.B Bederson. Human computation: a survey and taxonomy of a growing field. In *Proc. of SIGCHI'11*, pages 1403–1412, 2011.

[20] V. C. Raykar, S. Yu, L.H. Zhao, G.H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *J. of Machine Learning Research*, 11(Apr):1297–1322, 2010.

[21] R. Singh, L. Hélouët, and Z. Miklós. Reducing the cost of aggregation in crowdsourcing. In *Proc. of ICWS'20*, 2020.

[22] L. Tran-Thanh, M. Venanzi, A. Rogers, and N.R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proc. of AAMAS'13*, pages 901–908, 2013.

[23] W. Van Der Aalst, K. van Hee, A. ter Hofstede, N. Sidorova, H Verbeek, M. Voorhoeve, and M. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.

[24] D. Wei, S.B. Roy, and S. Amer-Yahia. Recommending deployment strategies for collaborative tasks. In *Proc. of SIGMOD'20*, pages 3–17. ACM, 2020.

[25] J. Whitehill, T. Wu, J. Bergsma, J.R. Movellan, and P.L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proc. of NIPS'09*, pages 2035–2043, 2009.

[26] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proc. of VLDB Endowment*, 10(5):541–552, 2017.