Improving Impact Sourcing via Efficient Global Service Delivery

Michael Borokhovich
The University of Texas at Austin
michaelbor@utexas.edu

Avhishek Chatterjee
The University of Texas at Austin
avhishek@utexas.edu

Jason Rogers
Samasource
jason.rogers@samasource.org

Lav R. Varshney
University of Illinois at Urbana-Champaign
varshney@illinois.edu

ABSTRACT

Impact sourcing outsources tasks to people living in poverty in underdeveloped regions, allowing them to earn the dignity of work and a living wage. Samasource is a leader in impact sourcing, decomposing and encapsulating data projects into microwork for global service delivery at centers around the world. By using microwork centers instead of large for-profit vendors, Samasource calculates customers can get jobs done for 30% to 40% less, which is important since cost is the major driver of new contracts (rather than social mission). A key part of operations is routing and scheduling work, currently performed manually. This paper first derives novel data-driven, queuing-theoretic algorithms for this problem and then demonstrates on real-world data that they can considerably improve global service delivery efficiency. This reduces turnaround time and costs, thereby increasing the scope of social impact. The path to deploying these algorithms in practice is also discussed.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; C.2 [**Computer Systems Organization**]: Computer Communication Networks

General Terms

Performance

Keywords

global service delivery, impact sourcing, routing, scheduling, social good

1. INTRODUCTION

Samasource is a mission-focused nonprofit company that connects people in poverty to life-changing digital work. Workers are primarily women and youth without a college degree and having never held a job before, located in underdeveloped regions such as Haiti, Ghana, Uganda, Kenya, and India. Since 2008, Samasource has paid out over $3.5 million in direct wages and benefits to 6,527 workers that also collectively support 20,161 income dependents, for a total of 26,688 people lifted out of poverty. Evaluation data has demonstrated the income increase of workers directly improves their quality of life as measured by spending in five key categories: shelter, food, education, local remittances, and savings. Unlike charitable giving which involves direct wealth transfers, so-called *impact sourcing* allows people to earn a living wage and to establish a dignified life for themselves and their dependents. Human resource development and worker churn is part of the model: 89% of workers pursue additional means of formal employment and/or education after working for Samasource.

Samasource approaches impact sourcing—essentially business process outsourcing to boost economic development—as follows. They provide requesters of digital work with project management and consultation on task design while connecting them with workers. Partner agencies in underdeveloped regions provide delivery centers, which are physical locations with required facilities (e.g., managers, computers, stable electricity) that workers go to for work [4, 7].

By impact sourcing rather than using large for-profit vendors, Samasource calculates customers can get jobs done for 30% to 40% less, which is important since cost is the major driver of new contracts (rather than social mission). Customers that request work include many large multinational corporations such as eBay, Getty Images, Google, TripAdvisor, Walmart, and Microsoft.

The basic work model is to break down and encapsulate large data projects into smaller *microwork* tasks that can be distributed to global work centers through a web-based platform called SamaHub. This approach of work decomposition, encapsulation, distribution, and recombination has emerged throughout the economy [1] and is very much in line with *global service delivery* systems that large for-profit companies such as Wipro and IBM have also deployed [5, 6, 9].

A key to efficient global service delivery is reducing coordination costs [3]. Encapsulation of microwork [8] enables optimal routing and scheduling algorithms to coordinate the skills required for doing work with the skills that workers possess [2, 10]. Samasource, however, uses manual assignment of work to work centers. The purpose of this paper is to develop coordination algorithms for the SamaHub plat-

Data for Good Exchange 2015 New York, New York USA

Table 1: Time Zones of Work Center Countries. Actually, Jason told us that they have the following time zones: (EAT, IST, EDT, GMT/UTC) which are: (3,5.5,-4,0)

Country	Time Zone
Haiti	UTC –05:00
Ghana	UTC +00:00
Uganda	UTC +03:00
Kenya	UTC +03:00
India	UTC +05:30

form to make it more efficient, to reduce costs, to drive more contracts, and thereby make greater social impact.

The data-driven, queuing-based algorithm we develop...

Demonstrate 3.5x improvement on all work, and 2x improvement on real-time work.

2. SYSTEM MODEL AND NOVEL ALGORITHMS

Tasks arrive in impact sourcing platforms over time. Each task has multiple steps and each step requires different skills and time. Similarly, agents arrive and leave the system (depending on working hours) and different agents have different skills and availability. The platform allocates tasks or steps at a regular interval of time to the available agents at that time. Steps of a task have precedence or ordering constraints among them between them, i.e., certain steps can be allocated only if the preceding steps have been completed. A step can only be allocated if we have an agent or a group of agents who have the required skills and the time to serve the step. A step can either be flexible or inflexible. A group of agents can pool their time and skills to serve a flexible step, whereas for an inflexible step we need to find an agent that has all the skills and time.

We designed queuing theoretic policies that are optimal in the sense that they can support maximum supportable task loads into the system. We proposed computationally efficient algorithms to implement the policies. Finally, we modified the policies to suit the special needs of impact sourcing platforms: (i) fast, simple and decentralized, (ii) customers' freedom to choose agents without hurting resource utilization and (iii) good performance in terms of backlog of tasks and total system time of a job which we call turn-around-time (TAT).

2.1 A Note on Theoretical Results

Our queuing theoretic algorithms are provably optimal for dynamical impact sourcing platforms where tasks and agents arrive over time according to stochastic processes. Optimality of the algorithms are in the sense of maximal system stability, a well known notion in dynamical and queuing systems. We also showed good backlog performance of our proposed schemes, specifically we show that for a broad class of task arrival processes the total backlog in the system at any time scales no faster than logarithm in the system size with very high probability. Details of the mathematical model of the system, algorithms and the theoretical results are in [].

3. SIMULATION FRAMEWORK

Figure 1: A sample black and white graphic (.eps format).

4. RESULTS

5. EVALUATION

In sections **??**–**??** we have characterized limits of different types of crowdsourcing systems, proposed efficient computational methods for centralized optimal schemes and designed decentralized schemes with provable bounds on task-backlog while offering freedom of choice to customers. This section complements the results presented in previous sections. Here we study crowdsourcing systems using real data from a non-profit crowdsourcing company and Monte Carlo simulations. We consider much simplified (in terms of computation and implementation) of the proposed decentralized algorithms above and study their performances on real as well as synthetic data.

We first consider performance evaluation on real data. In the real data that we have there are tasks with $2-?$ steps and both agents and steps are flexible. For this system, we implement a very simple version of STEP_FLEX where we prioritize the steps with higher precedence to choose agents greedily with random tie-breaking.

Let us first describe the evaluation of the Samasource's scheduling using its real data. The dataset contains 9.3M tasks and each belongs to a specific project. Some of the projects are regarded as *real-time* which means that they have higher priority. The overall number of tasks that belong to the real-time projects is about 4.2M. Each task is comprised of 1 or 2 steps which in turn comprised of a single sub-step. Some of the tasks have strict step ordering, i.e., the previous step must be completed before the next could be scheduled. Average sub-step working time requirement is $340sec$. From the dataset we were able to calculate the turn-around time (TAT) for each task, i.e., the time since the task arrived to the system until the time its last step was completed. The cdf of TAT for all projects and for real-time projects only can be found in Figure 2.

In order to compare the Samasource scheduling with our approach, we used the Samasource dataset as an input to our algorithm STEP_FLEX. We didn't have exact information about workers availability in the Samasource system, thus the following assumptions were made (and approved to be reasonable). The number of active workers in the Samasource system is 625, where each worker works every day from $9am$ to $5pm$ and they are evenly distributed across the four timezones: $-4, 0, 3, 5.5$. Each worker possesses the skills required for any sub-step in the dataset. In Figure 2 we compare the CDF of tasks turn-around time of the Samasource scheduling (calculated from the data) and our approach STEP_FLEX (simulated with the data as an input). We can see that our algorithm substantially outperforms the Samasource scheduling. The average TAT for all projects is $\times 6.5$ better and for the realtime projects is more than $\times 8$.

In Figure 3 we can see how the algorithm STEP_FLEX performs as a function of number of workers. As the number of workers grows, the tasks turn-around time decreases (see Figure 3(a)). The benefit of adding more workers to the system can be seen even clearer when we analyze the backlog, i.e., the average number of steps that entered the system but

were not scheduled yet. Figure 3(b) shows that the backlog queue substantially decreases when the number of workers increases.

Next we turn to compare our algorithms on synthetic data. For evaluation on synthetic data we consider flexible agents and flexible steps, and flexible agents and inflexible steps. Algorithm STEP_FLEX mentioned above is used for the first system. For the second system we use a simplified version of Restricted Greedy scheme where we prioritize steps with higher skill requirements and allocate among them greedily. In addition, we consider another scenario which is in between flexible and inflexible steps, here each sub-step has to be allocated to a single agent, but different sub-steps of a step can be allocated to different agents. For this system we consider STEP_SEMIFLEX where steps allocate themselves greedily while ensuring that a sub-step gets all service from an agent. While it is expected that STEP_FLEX will outperform STEP_INFLEX, we found, somewhat surprisingly, that STEP_FLEX and STEP_SEMIFLEX have very similar performance.

The first set of generated data included tasks with up to three steps in each and with strict ordering. Each step was comprised of one to three random sub-steps out of five possible types. Working time requirement for each sub-step was uniformly distributed between 60 and 600 seconds. Each worker in the system has daily availability from $9am$ to $5pm$ and and they are evenly distributed across the four timezones: $-4, 0, 3, 5.5$. A worker possesses a random set of skills that enables him to work on up to three (out of five) sub-step types. For each of our three algorithms we compare three metrics: TAT, backlog queue and workers utilization. The experiment simulated a single run over a timespan of 40 days.

In Figure 4 we can see that algorithms STEP_FLEX and STEP_SEMIFLEX outperform STEP_INFLEX for both cases: 500 workers in the system and 700 workers. When the load on the system is 150 tasks per hour and the number of workers is 500 we can see that algorithm STEP_INFLEX is substantially worse since it became unstable for this load. Notice that STEP_FLEX and STEP_SEMIFLEX perform very similar which can be explained with relatively short sub-step work time requirement in which case necessity to splitting it becomes a rear event. Another interesting observation is that the workers utilization of STEP_INFLEX is not much worse than of the other algorithms. This can be explained by the long backlog queue of STEP_INFLEX. Though it is harder for STEP_INFLEX to find a worker that is capable to work on on the whole step, when the backlog becomes large, the probability that a given worker will be assigned to some whole step is growing.

The last set of results uses the same synthetic data as the previous with a small change: working time requirement for each sub-step was uniformly distributed between 600 and 6000 seconds. In this experiment (see Figure 5) we can see a slight advantage of STEP_FLEX upon STEP_SEMIFLEX. This is due to the longer working time requirements per sub-step, since now the cases in which a sub-step may be split to improve scheduling are more probable. In this scenario, disadvantage of STEP_INFLEX is even more obvious: already for the load of 50 tasks per hour and 1200 workers, its TAT and backlog are very large and unstable.

To summarize, our approach substantially outperforms the scheduling scheme currently used by Samasource. While STEP_FLEX achieves best performance in terms of TAT and backlog, STEP_SEMIFLEX may be a good alternative. Its performance is almost the same but it does not require ability to split sub-steps among different workers, and it is computationally lighter.

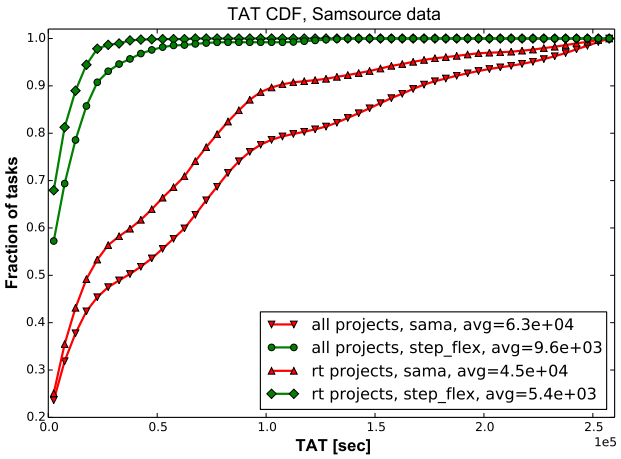
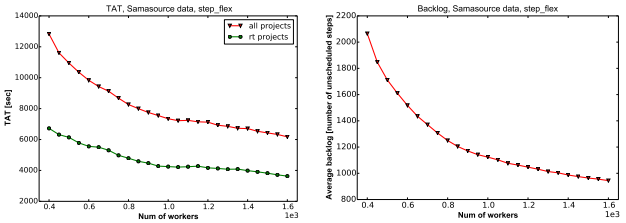


Figure 2: CDF of tasks turn-around time (TAT) using Samasource dataset. Samasource scheduling "sama" vs our algorithm STEP_FLEX.



(a) (b)

Figure 3: Performance of our STEP_FLEX algorithm on Samasource data, as a function of number of workers. (a) Tasks turn-around time (TAT). (b) Average backlog (number of unscheduled steps in the system).

6. CONCLUSION

We presented a set of novel algorithms, which are detailed elsewhere.

Demonstrated the possibility of significant improvement on real-world data

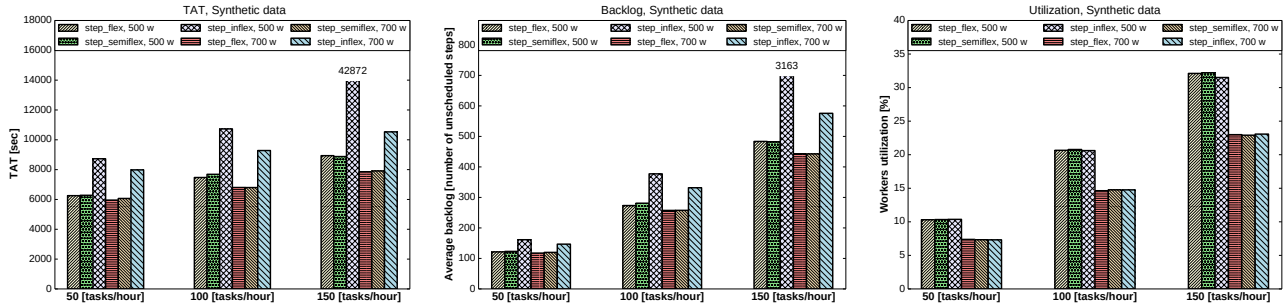
To deploy things in SamaHub, we would...

7. ACKNOWLEDGMENTS

We thank Sriram Vishwanath for his support in pursuing this project.

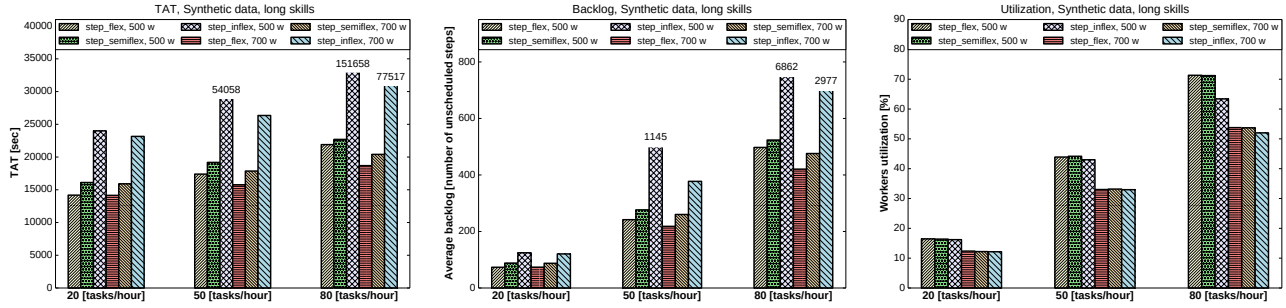
8. REFERENCES

[1] D. Bollier. *The Future of Work: What It Means for Individuals, Businesses, Markets and Governments.* The Aspen Institute, Washington, DC, 2011.



(a) (b) (c)

Figure 4: Performance of our algorithms on synthetic data with *short* sub-steps ($60 - 600$ sec), as a function of load. (a) Tasks turn-around time (TAT). (b) Average backlog (number of unscheduled steps in the system). (c) Workers utilization.



(a) (b) (c)

Figure 5: Performance of our algorithms on synthetic data with *long* sub-steps ($600 - 6000$ sec), as a function of load. (a) Tasks turn-around time (TAT). (b) Average backlog (number of unscheduled steps in the system). (c) Workers utilization.

[2] A. Chatterjee, L. R. Varshney, and S. Vishwananth. Work capacity of freelance markets: Fundamental limits and decentralized schemes. In *Proc. 2015 IEEE INFOCOM*, Apr. 2015. to appear.

[3] M. Ehret and J. Wirtz. Division of labor between firms: Business services, non-ownership-value and the rise of the service economy. *Service Sci.*, 2(3):136–145, Fall 2010.

[4] F. Gino and B. R. Staats. The microwork solution. *Harvard Bus. Rev.*, 90(12):92–96, Dec. 2012.

[5] IBM Global Business Services. Application assembly optimization: A new approach to global delivery, Aug. 2009.

[6] IBM Global Business Services. Application assembly optimization: A distinct approach to global delivery, Mar. 2010.

[7] A. Marcus and A. Parameswaran. *Crowdsourced Data Management: Industry and Academic Perspectives*. 2015. in preparation.

[8] D. Oppenheim, S. Bagheri, K. Ratakonda, and Y.-M. Chee. Coordinating distributed operations. In E. M. Maximilien, G. Rossi, S.-T. Yuan, H. Ludwig, and M. Fantinato, editors, *Service-Oriented Computing*, volume 6568 of *Lecture Notes in Computer Science*, pages 213–224. Springer, Berlin, 2011.

[9] D. M. Upton and V. A. Fuller. Wipro technologies: The factory model. Harvard Business School: 9-606-021, Oct. 2005.

[10] L. R. Varshney, S. Agarwal, Y.-M. Chee, R. R. Sindhgatta, D. V. Oppenheim, J. Lee, and K. Ratakonda. Cognitive coordination of global service delivery. arXiv:1406.0215 [cs.OH]., June 2014.