Computational Division of Labor with Human and AI Workers

Atsuyuki Morishima, Masaki Matsubara, Kei Wakabayashi, Nobutaka Suzuki, Hiroyoshi Ito
University of Tsukuba
{mori, masaki, kwakaba, nsuzuki, ito}@slis.tsukuba.ac.jp

Abstract

Computational division of labor addresses the design and analysis of algorithms for division of labor problems and will be one of the key issues in Future of Work. The problems deal with interactions among multiple worker and task classes in task decomposition, worker recruitment and education, and task assignments to AI and humans, for efficiently completing given tasks. We survey some of the related literature and discuss challenges and open problems.

1 Introduction

Division of Labor is known as an essential factor for achieving wealth of human beings. In division of labor, individuals and organizations acquire specialised capabilities and play different roles in a system, and either form combinations or trade to take advantage of their and others' capabilities.

In the recent years, more and more information on tasks is being circulated on the Internet, and there are many labor resources accessible through it. This allows us to take the computational approach to division of labor, in which AI and algorithm agents assign appropriate tasks to workers to achieve specific goals. There are many studies that address topics related to division of labor. For example, many papers on crowdsourcing address algorithms for task assignment considering given objective functions, such as price and required time. However, division of labor is different from just "decomposing into microtasks." Task decomposition itself does not necessary cause division of labor, if we obtain a set of tasks each of which requires workers to have the same set of capabilities represented by their attributes (such as skills and locations of workers). We need *task classes* that require different capabilities and *worker classes* that have workers with different capabilities. For example, assume that we have a set of sentences in English that needs to be translated into Braille in Japanese. Decomposing into a set of tasks each of which translates one sentence into Braille does not cause a division of labor, because we have only one task class that require the same skills. In contrast, decomposing them into a set of translation tasks from English to Japanese and another set of translation tasks from Japanese into Japanese Braille result in division of labor; then we have two task classes that require different specialized capabilities for translation. In addition, we may be able to further decompose the obtained task classes. For example, we can find a *subclass* under the English-Japanese translation task class in which the sentences require detailed knowledge of a particular domain (e.g., Japanese pop stars). Then, workers who have the knowledge can be assigned to the tasks in the subclass. Note that how to decompose task classes depends on the availability of workers in the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

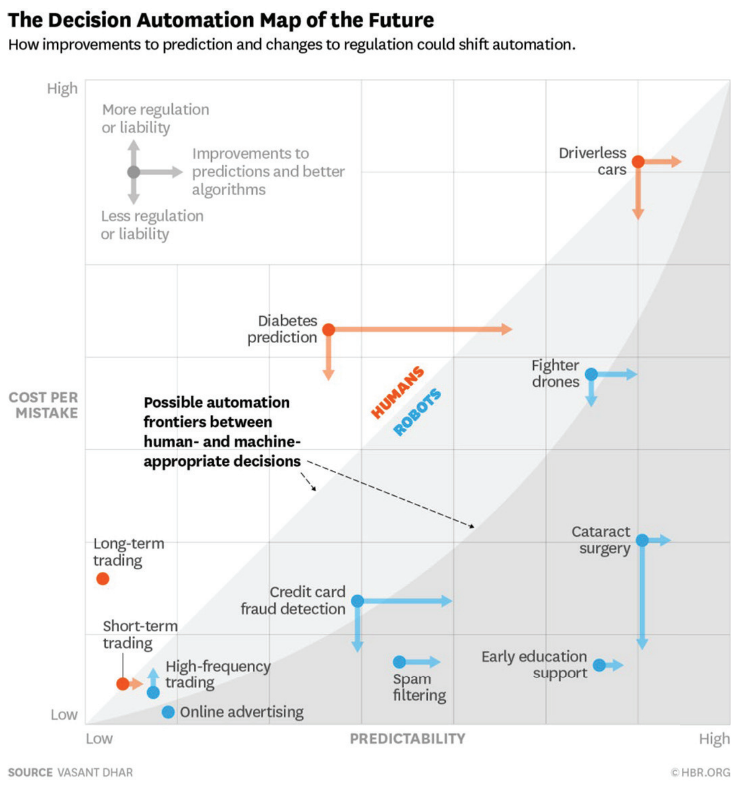

Figure 1: Whether decisions in particular areas should be made by humans or AI [4]. The x-axis is the predictability by AI. The y-axis is the cost per mistake. This figure states that decisions can be made by AI if it has high predictability and low cost per mistake. We already have many things for which we can use AI (dots) and we expect to have more things as we have improvement in predictions and algorithms (horizontal arrows), and regulations or liability changes (vertical arrows) in the near future.

worker pool. If we had a lot of people who are good at translating English Braille into Japanese Braille, we would have different subclasses: Translating English into Braille and translating it into Japanese one.

Adam Smith pointed out that the efficiency of division of labor comes from the following benefits: Increase of workers' capabilities, lower switching cost of tasks, and machines taking place of manual labor [27]. Taking different roles and trading or combining their abilities and products has a dramatic effect; it would be impossible for us to obtain many things today, let's say a smartphone, in exchange for the work for several to tens of hours, if each of us made our smartphone ourselves, without having a large amount of people who produce materials and semiconductor chips, design the electronic circuit and user interface, develop software, etc. Computational division of labor will be one of the important issues in Future of Work [1]. However, as we will show, this area is still in its infancy and there are a lot of things we can do.

While we have an environment that makes us ready to pursue the computational division of labor today, there is the demand for the computational approach. We recently noticed that worker availability can be changed in a disease pandemic. However, one of the most important factors is the rapid growth of AI. Figure **??** shows whether decisions in particular areas should be performed by humans or AIs at present and the potential changes in the near future. It depends not only the predictability but also cost per mistake. As shown in the figure, we already have a lot of things we can rely on AI, and there will be more and more in the near future, as we see improvement to predictions and better algorithms, and regulations or liability changes. Many new kinds of commercial services that employ new combinations of AI and human resources are emerging at a great speed. This means that AI is important not only for implementing solutions of computational division of labor, but also as *AI workers*,

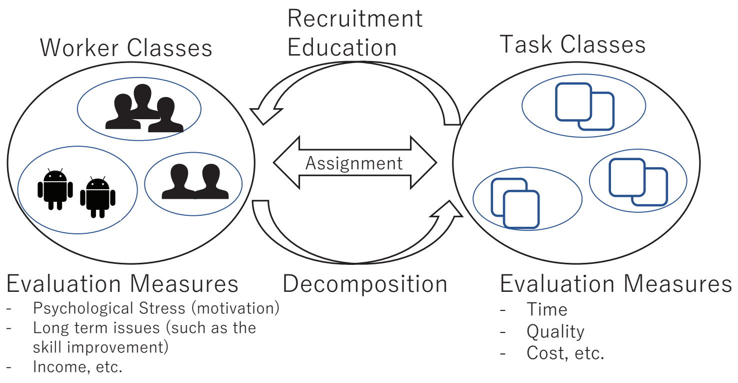

Figure 2: Framework for computational division of labor

that work in collaboration with human workers. In addition, the change spreads world wide in a short period of time; we see accelerating technology change [16] and new services and industries are deployed at a large scale internationally. Such a *scale* and *speed* of change caused by the *rapid growth of AI technology* requires the computational approach to the division of labor problems.

Note that there are a variety of tasks beyond decision tasks, which often require other capabilities than high predictability such as flexible response, some of which are difficult for the AI today, although the situation may change in the near future. For example, think about a conveyor belt sushi restaurant which is popular in Japan. Although making oval-shaped rices for sushi used to be considered as an expert task that is allowed to be performed by experienced staff members only, the advance of technology allowed machines to make the rices in conveyor belt sushi restaurants. Even fish slices to be put onto the rice are being made by machines with AI. However, there are many other tasks that are being performed by humans workers. They *develop* and *teach* AI, communicate with customers, cook special kinds of sushi, and deal with non-routine issues, all of which are difficult tasks for AI today.

In this article, we investigate problems on computational division of labor and look at the current status of research on crowdsourcing and human-in-the-loop systems on the Internet from the division-of-labor perspective. We chose these areas because in the near future, we expect that many jobs will be supplied by human-in-the-loop online job platforms [8].

Paper Outline. The rest of the paper is organized as follows. In Section 2, we explain what computational division of labor problems deal with and introduce a set of dimensions and terms to classify existing solutions for problems related to computational division of labor. In Section 3, we look into some of existing literature and investigate the current status. Section 4 discusses challenges and open problems.

2 Computational Division of Labor

The "computational division of labor" is not a new concept, and has been addressed in many related areas, notably in crowdsourcing research, although the state is still in its infancy as we will show in the next section.

Figure 1 shows the framework for computational division of labor. It has two key components: *task classes* and *worker classes*. A task class defines a set of task instances that require a specific set of capabilities in order to perform them. A worker class defines a set of workers having a specific set of capabilities. The task and worker classes have *mutual dependence*; i.e., good task decomposition results in

$$\arg\max_{T \in \mathcal{T}} \ \vec{f}(\mathcal{M}(W, T)),$$

where W is a set of workers, $\mathcal{T}$ is the collection of feasible task decompositions for W, $\mathcal{M}$ is the appropriate

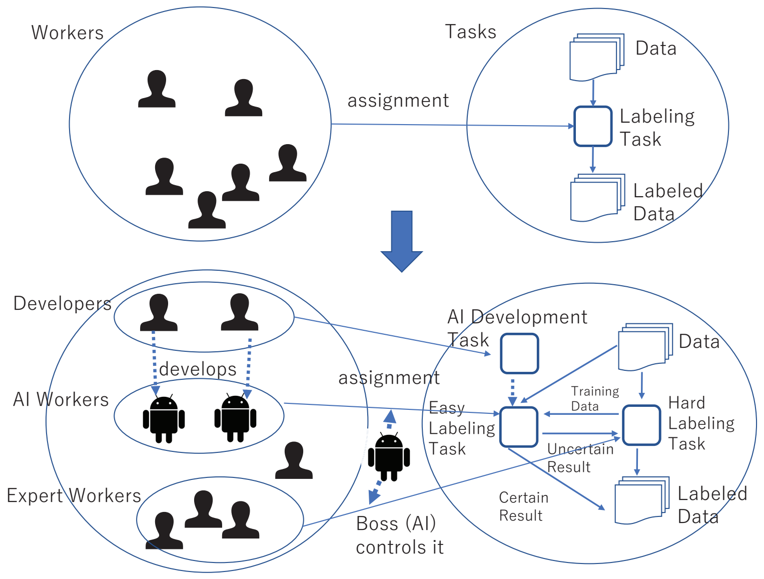

Figure 3: Example of dividing tasks and workers into multiple classes. If we know we have software developers in the worker pool, we can decompose the original task into a set of tasks consisting of AI development task, AI labeling task, and human labeling task. Since the tasks include tasks for AI workers, we recruit software developers so that AI workers are added to the worker pool.

assignment from tasks to qualified workers, and $\vec{f}$ is the multi objective functions that define what is good assignment. The functions may include a variety of things, including the result quality, the expected cost due to mistakes, and human factors related to workers [2]. In the opposite direction, what skills workers should obtain and what kinds of AI workers should be developed are described by

$$\arg\max_{W \in \mathcal{W}} \ \vec{f}(\mathcal{M}(W, T)),$$

where $\mathcal{W}$ is the collection of sets of workers augmented by T, i.e., each $W \in \mathcal{W}$ can be a set of workers with new or better skills, or an extended set of workers with newly developed AI workers. In addition, the set of skills the decomposed tasks require will affect how we should recruit workers.

Figure 3 gives an example process of computational division of labor for labeling tasks. If we know that we have software developers in the worker pool, we can decompose the original task into a set of tasks consisting of AI development tasks, easy labeling tasks (to be assigned to AI workers) and hard labeling tasks (assigned to human workers). Since the tasks include tasks for AI workers, we recruit software developers so that AI workers are added to the worker pool.

Assignment of workers to tasks is often determined by considering not only short-term benefits (such as time, quality, cost) for requesters, but long-term benefits, such as social sustainability and inclusion, with the three important benefits (skill improvement, low switching cost, and AI utilization) of division of labor considered.

The essential part of division of labor in the framework is that there must be a variety of task classes, each of which requires different expertise or abilities to complete them. Division of labor can lead to efficient society by exploiting the following advantages [27]:

Increase of Workers' Capabilities Some tasks may require special expertise and workers need to have experiences or be trained for doing a good job on the task. This sometimes requires long-term commitment of workers to a set of tasks that require particular capabilities.

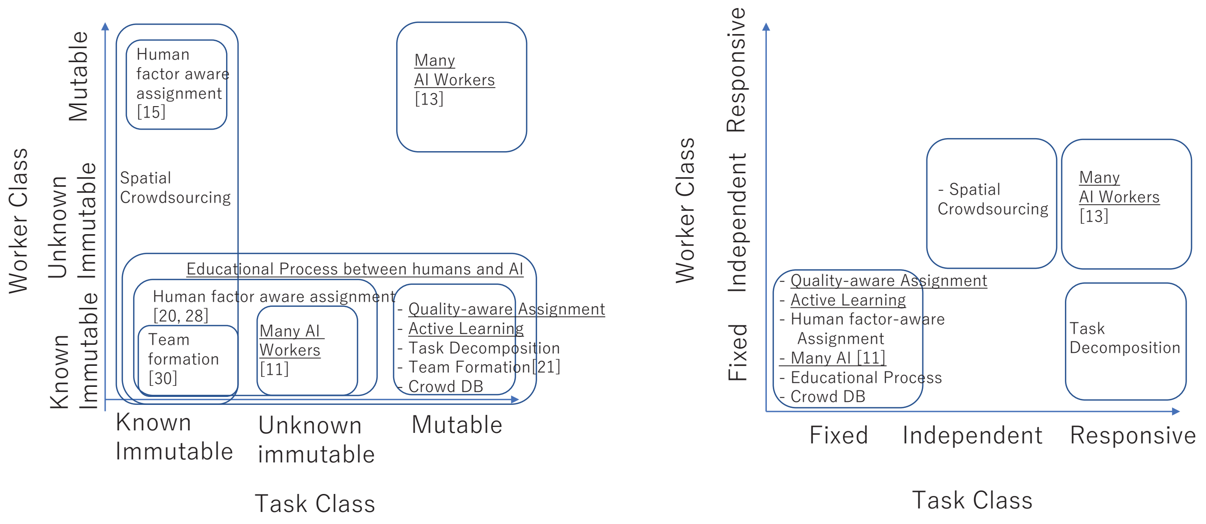

Figure 4: Two views of a solution space of computational division of labor and the current status of solutions in related topics. On the left is the view with instnace-level mutability class dimentions. On the right is one with class-level dynamicity dimensions. Underlined Topics deal with both of human and AI workers, while the others deal with human workers only.

Lower Switching Cost Cost for switching into completely new tasks is generally high for humans. Taking into this factor when assigning tasks to human workers increase the efficiency of manual labor.

Machine (AI) Taking Place of Manual Labor Dividing the task into sub-tasks can increase the opportunities for machines (AI workers) to do the task, if the task is appropriately extracted so that the AI worker is capable of doing the task. In order to achieve this, we need to address meta-level algorithms that explicitly deal with capabilities of available AI workers.

3 Related Research and Current Status

There is a lot of research done related to computational division of labor. This section tries to organize relevant topics especially in the area of human-in-the-loop database systems, crowdsourcing and machine learning. Although crowdsourcing research generally focuses on the case where workers are human workers and the objective functions are given in terms of each workflow (e.g., quality, time, money), there are papers that address important components of division of labor such as task and worker classes and the three benefits. Human-in-the-loop database systems combine human and AI workers in the way that human workers collect data that are not stored in the current snapshot of database. Regarding the division of labor between humans and AI workers, machine learning plays an essential role in some division of labor problems. Some studies such as supervised learning focus on addressing how to replace human labor by teaching AIs via a dynamic interaction between AIs and human workers.

3.1 Dimensions and Overview

Existing solutions for problems related to computational division of labor can be seen from the division of labor perspective by placing them in a space with the dimensions related to task and worker classes, such as follows:

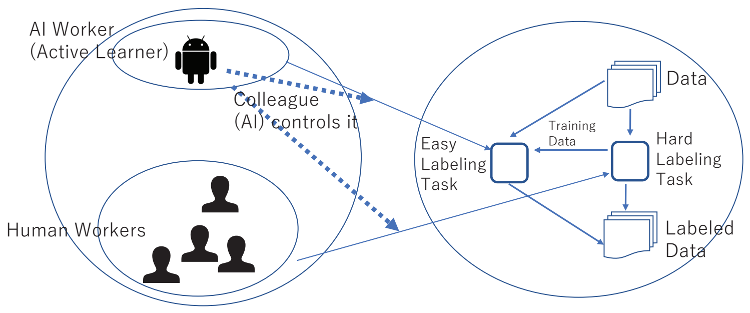

Figure 5: Active learning from the perspective of computational division of labor. The active learner is a colleague AI worker of human workers, who controls the task assignment.

A. Instance level mutability of Task and Worker Classes. The dimentions represents the interaction between task (or worker) classes and their instances with the three categories:

Known Immutable The solution assumes that we already know which class each task or worker belongs to, and the instance-of relationship does not change.

Unknown Immutable The solution does not know which class each task or worker belongs to at first, but the instance-of relationship does not change once it decides the membership.

Mutable The solution changes the instance-of relationship according to a change of situation. Thus the number of instances of each class changes.

For example, some research papers classify crowd workers into novice and expert workers. Other papers classify them into groups each of which contains similar workers in terms of accuracy per labels. If a method assumes that we know who are novice and expert workers in advance, it is labeled with “Known Immutable.” If a method measures the workers’ skills and put them into classes only once, it is labeled with “Unknown Immutable.” If a method regularly checks the skills and changes the memberships accordingly, it is labeled with “Mutable.” An example with task classes is as follows: If a complex workflow is required in an application, the workflow may contain different kinds of tasks, such as find, fix and verification tasks [3], which are connected to each other in the workflow. If the solution takes as input such a workflow and does not decompose it, it is labeled with “known Immutable.” If a method decomposes the tasks into smaller ones, it is labeled with “Mutable.” Another example is the case where we have a set of data labeling tasks and find a subset of the data labeling tasks appropriate for training AI workers. Then the set of tasks will be a new task class. A method to find such a subset is either “Unknown Immutable” or “Mutable” depending on whether it updates it during the execution or not.

B. Class-level Dynamicity of Task and Worker Classes. The dimensions represent the interaction between task and worker classes with the three categories:

Fixed The solution assumes that the set of task or worker classes is fixed and always contains their instances.

Independent The solution allows that task (or worker) classes can be added or deleted during the execution, and such operation is done independent of worker (or task) classes.

Responsive The solution adds or deletes tasks or worker classes during the execution, and the action is affected by the situation in their counterpart (e.g., changes in worker classes cause the division of task classes).

For example, a task decomposition method may not look at available workers at all. Such a method is labeled with “Independent.” Another may consider the current availability of workers before the task decomposition.

Such a method is labeled with "Responsive." Responsiveness is definitely the potential benefit of computational division of labor, which will be useful in situations such as COVID-19 pandemic where we encounter a sudden change in labor resources.

Figure 4 puts some of the related topics (details will be shown in Figure 6) in the three dimension space. The results show that, at this moment, there are a limited number of studies that deal with responsive class generation. They deal with the two cases where (1) they assume human workers only and do not deal with automatic task decomposition and workflow optimization, and (2) they deal with human and AI workers for simple data labeling tasks. In contrast, there are few studies that deal with dynamic task and worker classes containing AI workers. Many studies focus on dynamicity of either worker or task classes only. Among the three benefits of division of labor, few studies focus on the problem of lowering switching cost. Some address the problem of improving workers' skills, and having Machine (AI) take place of manual labor, but the they addressed the problems independently. Most of the objective functions are defined in terms of short-term, requester-centric views on each workflow.

C. Controller of Division of Labor. In addition, there are different approaches on who controls the division of labor. The followings are potential subjects that control the division of labor process.

Boss There are approaches that assume a subject other than workers, who mainly takes care of the task decomposition, assignment, recruitment, etc. The boss can be a human, an AI agent, or a human-in-the-loop algorithm.

Colleague There are some cases where one of the workers decides who perform what tasks. For example, we can view active learning methods in the division of labor perspective as follows. We have two types of workers (a machine learner and humans), where the decision maker who assigns tasks for data labeling is the learner (Figure 5).

No one There is nobody who explicitly controls the division of labor. Rather, how it goes is incorporated in the design of the framework, such as incentive design, to implement "invisible hands." The task decomposition, assignment, education, recruitment, etc. are implemented as the result of every participant's action in the process of pursuit of their own gain.

3.2 Computational Division of labor View of Existing Studies

Figure 6 shows some of the related works that satisfy one of the following conditions: (1) The work deals with more than one task or worker classes (2) the work addresses issues related to benefits of division of labor - increasing workers' capabilities, lower switching cost, and AI workers taking place of manual labor.

Quality-Aware Microtask Assignment. For finite pool data categorization, there are approaches to assign appropriate tasks to different classes of workers to improve the result quality. For example, [18] discusses how to assign the categorization tasks to two categories of human workers, namely, experts and crowd workers, and an AI worker (a classifier model), in order to achieve high quality results. In the method, the AI worker is considered as a worker that responds to all the tasks that have been assigned to no human workers when the budget is run out. The method dynamically estimates the result quality in a situation that we train the AI worker with tasks labeled by humans and assign the rest of tasks to the trained AI worker. In terms of class-level dynamicity, it does not change task or worker classes during the execution. Furthermore, it always assumes to use a particular AI worker whose algorithm is known and does not accept AI workers that are developed by crowd workers.

Spatial Crowdsourcing. In spatial crowdsourcing, tasks and workers are associated to locations. Logically, we can think of many subclasses of each of the task and worker class, defined by their locations. The (sub-)class of each task or worker is known because we know their locations. In some settings, workers move thus the the

Topic	Examples	Task Classes and Relationship with their Instances	Worker Classes and Relationship with their Instances	Class-level Dynamicity and Interaction	Objective Functions and Constraints	Controller
Quality-Aware Microtask Assignment	[18]	Three subclasses (Crowd, Expert, AI) of a data labeling task class. Mutable	Three classes (Crowd, Expert, AI). Known Immutable	No change at task and worker class levels.	Better quality with a limited budget	Boss (AI)
Spatial Crowdsoucring Task Assignment	[29]	Subclasses of a spatial task class with different locations. Known Immutable	Classes for human workers in different locations. Known Immutable or Mutable	Task and Worker classes can be added and deleted dynamically, but independent of each other.	Maximizing total number of assignments, Optimized for average performance, Maximizing total payoff, etc.	Boss (AI)
Active Learning	[31, 22, 32, 9]	Subclasses (easy and hard) of a data labeling task. Mutable	Two classes (AI and Human) known Immutable.	No change at task and worker class levels.	Better machine learning models	Colleague (AI)
Working with a Large Number of Data Labeling AI Workers	[13]	Subclasses of a data labeling task class for human and AI workers. Mutable	One human worker class and many AI worker classes for different skills. Unknown Immutable	Automatic subclass generation and assignment triggered by dynamic estimation of AI worker performance	Quality and speed of task completion	Boss (AI)
	[11]	Two classes for easy and hard labeling tasks. Unknown Immutable.	Two classes for AI and Human. Known Immutable	Task classes are computed once based on a task prioritization algorithm before assignment	Better performance of AI Workers	Boss (AI)
Task Decomposition	[14]	Many dynamically-generated classes. Mutable.	One (human workers). Known immutable.	Task classes are dynamically generated, according to workers' judgement.	Completing tasks with the crowd. Each task must be done for a fixed price	Boss (human workers)
Team Formation	[30]	Many classes with different roles. Anyone can propose new role and edit role structure. Mutable.	Two (Team leader and team members). Known Immutable	Task classes are first defined by the leader, but can be reconfigured manually by anyone during the execution.	Organizing the teams to accomplish complex work with deadline of six weeks and budget.	Colleague (Human) and Boss (Human)
	[21]	Complex workflow consists of several tasks that require different capabilities. Known Immutable	Many classes with different capabilities and wage expectation. Known Immutable	No change at task and worker class levels. The algorithm optimally forms groups with available workers pool	Worker-worker affinity and upper critical mass with skill and cost constraint	Boss (AI)
Human-Factor Aware Microtask Assignment	[15]	One for speech transcription. Known Immutable	N subclasses. Workers are divided into each subclass based on current skill distribution. Mutable	No change at task and worker class levels.	Skill, Psychological Stress	Boss (AI)
	[28]	Each task will be broken down by the mentor into several subtasks only once. Unknown Immutable.	Two (Mentor and Intern worker). Known Immutable.	No change at task and worker class level.	Better learning of intern worker	Boss (Human)
	[20]	Many: each task has different content and requirements associated to keywords. Known Immutable	Many: each worker has different interests associated to keywords. Known Immutable	No change at task and worker class level. However, correspondences between task and worker classes are dynamically changed.	Worker Motivation, Task relevance, and Task diversity	Boss (AI)
Educational Process between Human and AI	[24, 17, 5, 10]	Two classes (tasks for education and not) of data labeling tasks. They are Known Immutable in [10], and Mutable in the others.	Two classes (Human and teaching AI). Known Immutable.	No change at task and worker class level.	AI learns effective teaching schemes and humans get higher ability to the task.	Colleague (AI)
Crowd Databases	[6, 19]	Many Tasks (filtering, join, etc.). Mutable	Two (Humans and DBMS). Known Immutable	Task classes are determined in the optimization phase based on the data statistics and fixed before the execution.	Monetary cost, Quality	Colleague (AI)

Figure 6: Division-of-Labor view of some of related research topics that deal with at least more than one worker or task class or address issues related to benefits of division of labor

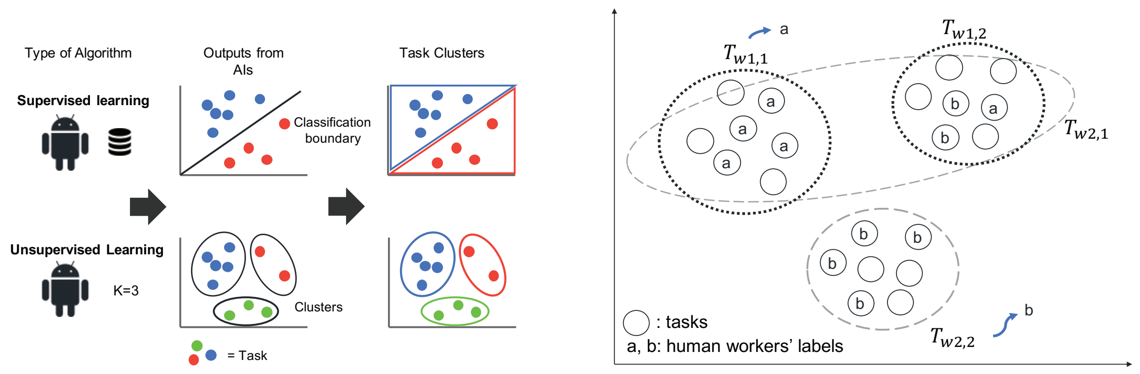

Figure 7: The method proposed in [13]. Each AI worker is assumed to cluster tasks, regardless of how it is implemented (left). The method conducts statistical tests to find whether all tasks in each task cluster is associated to a particular label, by looking at the labels given by human workers (Right). Here, $T_{w_{i,j}}$ means the jth task cluster of AI worker i. a and b are labels given by human workers.

worker classes they belong to are mutable. The task (worker) classes can be added and deleted dynamically as their instances appear and disappear, independently of workers (tasks).

Active Learning. Active learning interacts with human workers to (i) ask humans to make training data for data instances specified by *an AI worker* and (ii) receive the training data from humans to better learn the way to work [26]. Some active learning methods are aware of the performance of each worker and assign tasks to specific workers [22, 31]. In this discussion, we need to clearly distinguish the two roles of AI workers; a controller of task allocation to task classes and a task executor. The task allocation controller is a subject that organizes the dynamic allocation of data labeling tasks to the task class for human workers (i.e., hard task class), while a task executor is a subject that accomplishes data labeling tasks that could be assigned to human workers. From the viewpoint of the division of labor between humans and AIs, it is helpful to recognize that these active learning systems play two different roles because we potentially can find more flexible way to design AI components, e.g., working with a large number of task executor AIs, which we explain later. While the typical goal of active learning is to take over the human labor on data labeling tasks as a task executor, the allocation function is elaborated in some active learning models proposed so far. For example, Fang et al. [5] proposed an active learning model that encourages human learning by selecting a pair of workers having different skills to work together on the same data labeling task. Another example can be found in active learning that dynamically estimates the difficulty of tasks and assign only difficult tasks to workers of domain expert for minimizing the labor cost [9, 18, 32]. In terms of class-level dynamicity, no change happens on the task an dworker classes in active learning systems.

Working with a Large Number of Data Labeling AI Workers. There are studies on labeling a finite number of data items with not only human workers but also *a large number of* AI workers. The paper [13] proposes HACTAP (Human+AI Crowd Task Assignment Problem) that allows black box AI workers to join the workflow during its execution and assigns tasks to them if the high quality result is expected. Figure 7 shows the proposed method shown in [13]. Their method does not assume any particular model implemented in each AI worker. Rather, they assume that each AI worker outputs task clusters (which correspond to task (sub) classes in this paper) (Figure 7(left)). A task cluster will be meaningful if all tasks in each task cluster are associated to the same label. Therefore, the method conducts statistical tests to know whether each cluster corresponds to a particular label, by looking at the labels given by human workers. If the task cluster passes the test, all tasks in the cluster will be assigned to the AI worker. The method allows dynamic task assignment according to the available AI workers at each time, but the workflow is limited to a simple labeling task. The paper [11] presents a batch prioritization of data labeling tasks that allows a large number of black box AI workers to be efficiently trained. It statically assigns tasks to humans in advance before the task execution so that it effectively train AI workers independent of their underlying models. Thus, it can train a large number of blackbox AI workers with different underlying models in parallel. In terms of class-level dynamicity, there are only two fixed classes for tasks and workers. However, their hard labeling tasks are carefully chosen so that the tasks are effective for training AI workers with any underlying models.

Task Decomposition. There are studies to ask human workers to do task decomposition. For example, [14] proposes the PDS (Price-Divide-Solve) algorithm to ask crowd workers to decomposed tasks into smaller ones. The result workflow contains a diverse set of microtasks whose results are merged to produce the final product. Worker classes are not explicitly dealt with in the algorithm, but they assume that each task is done with the fixed payment (20 cents, in their implementation). This implies that it assumes that the task is easy enough so that we can easily find workers that do the task with the payment in crowdsourcing platforms.

Team Formation. There are studies on how to configure teams to solve complex problems. [30] proposed flash-organization which enables us to hire expert crowd workers into role structures and dynamically reconfigure the structure via version control. [21] proposed an optimization model for task assignment in a collaborative crowdsourcing environment and proposed optimization algorithms with theoretical guarantees. From the aspect of the computational division of labor, [30] allows human workers to change task classes, but it is not automatic, while [21] performs automatic task assignment at class-level, but does not allow dynamic change at the class level. Assignment to AI workers is not discussed in both of them.

Human-Factor Aware Microtask Assignment. While many crowdsourcing studies have assumed that anonymous workers have the same role, every human being is different, i.e. what tasks the workers are good at, what motivates them, and what they are doing the task for, are different for each person. Thus, considering the human rights of workers, it is natural to take into account the worker's perspective, in other words, it is important to consider who, when, and which task should be performed by whom. Therefore, the number of worker class is usually plural in this topic. Recently, the importance of human-factor in crowdsourcing has been argued [2], and task assignment research has been addressed in line with this perspective. For example, [36] considers psychological stress, [20] considers motivation, and [28] considers worker's learning. Class-level dynamicity is not addressed in this topic.

Educational Process between Humans and AI. Assigning a task to workers with appropriate ability is important for efficient problem solving. Increasing workers' skills is an important issue in division of labor, from the viewpoint of obtaining high-quality task results in the long term. For this problem, there are studies dealing with the interactive educational process between humans and AI. In this process, the AI learns the optimal task assignment to maximize the learning effect for human workers. These researches give us new insight into the task division from the viewpoint of the cultivation of expert human workers. However, the existing studies discuss improving skills *within* a particulars task class. From the division of labor view, it is important to address the problem of improving skills across task classes in different workflows.

Crowd Databases. Crowd Databases such as CrowdDB[6] and Deco[19] employ human workers to obtain data that are not stored in the current snapshot of database in the storage. The workflow contains a variety of tasks such as data entry, selections, join, ordering, while it does not explicitly deal with worker classes and attributes. The optimization of workflow is based on the data statistics, rather than worker availability and their skills.

4 Challenges and Open Problems

As shown in Section 3, although there have been many studies that address topics relevant to computational division of labor, this area is still in its infancy. In most studies, problems such as task decomposition, worker recruitment and education, are discussed with particular assumptions on available workers and decomposed tasks. There are only a few studies that deal with dynamic interactions between the skills of available workers and task decomposition, and that focus on benefits of division of labor in their objective functions. Given the current status, this section discusses challenges and some of open problems in computational division of labor with human and

AI workers.

4.1 Workers to Tasks

Knowing Relevant and Qualified AI Workers. The decomposed tasks include those to be performed by human and AI workers. However, AI workers are more diverse to each other in the skills than human workers. If a task is given, finding AI workers that we can be employed in its decomposed tasks will be a challenge.
Worker-Conscious Task Decomposition. In the existing research, the task decomposition is conducted once before the execution, assuming the simple assumption on workers (e.g., there are enough number of workers in the worker pool who are able to perform every task). Task decomposition schemes that are more conscious of the available workers in the pool will be an interesting issue. Another interesting issues is the tailor-made task extraction schemes; if a worker shows an interested in the project, the system extracts a task for him considering her skills and other constraints. If we include AI workers in the worker pool, workers will be more diverse in their speed, skills, and appropriate interfaces. An AI worker handles a bundle of tasks better than performing each task one by one. We need to deal with such a diversity.
On-the-Fly Workflow Switch without Stopping Its Execution. The situation of worker pool sometimes changes as time goes. For example, workers in Japan usually sleep at night in Japan Time and there will be a lower number of workers who can process Japanese. If a pandemic happens, worker distributions will dramatically change. Reassembling tasks and switch to new ones should be done without stopping its execution, while keeping a certain service level. When workers change, it would be inefficient to calculate the optimal solution from scratch each time a change occurs. Therefore, optimization mechanism that adapts to changes of workers and tasks, e.g., incremental algorithm, is becoming more important than before. As a preceding study, an incremental algorithm for finding an optimum worker assignment when a worker set changes is proposed [23].

4.2 Tasks to Workers

Incentive Design for Recruiting and Developing AI Workers. Tasks registered in the task pool for human workers do not necessarily require human workers and sometimes can be processed by AI workers. For example, active learners that are appropriately trained with crowdsourced labels sometimes output good quality results [31]. In some cases, we may be able to ask AI workers to performs most of a tremendous number of tasks. If we gave a good incentive to people, they would search for or develop their AI workers to perform the available tasks. The open question is how to design such an incentive. Effective and fair payment framework for AI developers needs to be investigated.
Psychological Stress Management of Human Workers. Job change is one of the things that give psychological stress to people [7]. Therefore, the responsiveness introduced by computational division of labor may cause additional psychological stress. We need to take into consideration the skill of workers, the types of tasks they have done so far, and their long-term career plans.
Matching of Supply and Demand of Skills. Mismatch of supply and demand of skills cause problems in lack and excess in labor resources. Developing ways to make the demand of the required skills visible will affect workers on choosing skills to learn and designing their long-term career for their future.
Education Strategies for Human and AI Workers. Most existing research for educating people in crowdsourcing settings all targets a particular set of microtasks. Extracting common skills from the task markets and provide educations for workers will be indispensable for the efficient learning and education strategies.

4.3 Holistic Perspective

Integration of Human and AI Worker Results. We naturally obtain diversity with human workers. Therefore, many existing studies on integrating results from crowd workers assumes the diversity. In contrast, a set of AI workers may implement similar algorithm and we may not able to consider them to make completely independent decisions.

Human-in-the-Loop Division of Labor Algorithms. Division of labor itself can be implemented with AI and human computation. Many problems in computational division of labor themselves can be regarded as "tasks." Thus, they can be implemented with AI and human computation. For example, "optimization algorithm for task decomposition" in which workers contribute their computational power to some part of optimization would be an algorithm design of great interest.

Social-Level Objective Functions. Most of related literature that address optimization problems has objective functions at a requester or a worker level. However, optimization focusing on the project-level efficiency only often concludes that working with only a few high performers is the best solution. In addition, platform-based recruitment of workers often cause price discrimination and exclusion of particular groups of workers [25]. Taking care of social-level objective functions, such as achieving inclusive labor markets, will be an important open problem.

5 Conclusion

In the recent years, more and more information on tasks is being circulated on the internet, and there are many labor resources accessible through it. In addition, we see the rapid growth of AI technology that allows us to have them workers for our tasks. This motivates us to take the computational approach to division of labor, in which we use AI agents that implement algorithms to assign appropriate tasks to human and AI workers to achieve specific goals. This paper investigated problems on computational division of labor around crowdsourcing and data-centric human-in-the-loop systems research. We explained what computational division of labor problems deal with and introduced a set of dimensions and terms to classify existing solutions for problems related to this topic. We identified the current status of this topic and showed that there are a number of interesting research challenges.

Acknowledgment

This work was partially supported by JST CREST Grant Number JPMJCR16E3, Japan.

References

[1] S. Amer-Yahia et al. Making ai machines work for humans in fow. *ACM SIGMOD RECORD (to appear)*, 2020.

[2] S. Amer-Yahia and S. B. Roy. Toward worker-centric crowdsourcing. *IEEE Data Eng. Bull.*, 39(4):3–13, 2016.

[3] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, October 3-6, 2010*, pages 313–322, 2010.

[4] V. Dhar. When to trust robots with decisions, and when not to. *Harvard Business Review*, May 2016.

[5] M. Fang, X. Zhu, B. Li, W. Ding, and X. Wu. Self-Taught active learning from crowds. In *2012 IEEE 12th International Conference on Data Mining*, pages 858–863, Dec. 2012.

[6] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 61–72, 2011.

[7] J. GREUBEL and G. KECKLUND. The impact of organizational changes on work stress, sleep, recovery and health. *Industrial Health*, advpub:1102240056–1102240056, 2011.

[8] D. Gross-Amblard, A. Morishima, S. Thirumuruganathan, M. Tommasi, and K. Yoshida. Platform design for crowdsourcing and future of work. *IEEE Data Eng. Bull.*, 42(4):35–45, 2019.

[9] S. Hao, S. C. H. Hoi, C. Miao, and P. Zhao. Active crowdsourcing for annotation. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 1–8. IEEE, Dec. 2015.

[10] E. Johns, O. Mac Aodha, and G. J. Brostow. Becoming the expert - interactive multi-class machine teaching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2624, 2015.

[11] M. Kimura, K. Wakabayashi, and A. Morishima. Batch prioritization of data labeling tasks for training classifiers. In *Proceedings of the Eighth AAAI Conference on Human Computation and Crowdsourcing,HCOMP 2020*, 2020.

[12] M. Kobayashi, H. Morita, M. Matsubara, N. Shimizu, and A. Morishima. An empirical study on short- and long-term effects of self-correction in crowdsourced microtasks. In *Proceedings of the Sixth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2018, Zürich, Switzerland, July 5-8, 2018*, pages 79–87, 2018.

[13] M. Kobayashi, K. Wakabayashi, and A. Morishima. Quality-aware dynamic task assignment in human+ai crowd. In A. E. F. Seghrouchni, G. Sukthankar, T. Liu, and M. van Steen, editors, *Companion of The 2020 Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 118–119. ACM / IW3C2, 2020.

[14] A. Kulkarni, M. Can, and B. Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1003–1012, New York, NY, USA, 2012. ACM.

[15] K. Kumai, M. Matsubara, Y. Shiraishi, D. Wakatsuki, J. Zhang, T. Shionome, H. Kitagawa, and A. Morishima. Skill-and-stress-aware assignment of crowd-worker groups to task streams. In *Proceedings of the Sixth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2018, Zürich, Switzerland, July 5-8, 2018.*, pages 88–97, 2018.

[16] R. Kurzweil. The law of accelerating returns, 2001. kurzweilai.net/the-law-of-accelerating-returns.

[17] W. Liu, B. Dai, A. Humayun, C. Tay, C. Yu, L. B. Smith, J. M. Rehg, and L. Song. Iterative machine teaching. *arXiv preprint arXiv:1705.10470*, 2017.

[18] A. T. Nguyen, B. C. Wallace, and M. Lease. Combining crowd and expert labels using decision theoretic active learning. In E. Gerber and P. Ipeirotis, editors, *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2015, November 8-11, 2015, San Diego, California, USA*, pages 120–129. AAAI Press, 2015.

[19] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 1203–1212, 2012.

[20] J. Pilourdault, S. Amer-Yahia, S. B. Roy, and D. Lee. Task relevance and diversity as worker motivation in crowdsourcing. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 365–376. IEEE, 2018.

[21] H. Rahman, S. B. Roy, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Optimized group formation for solving collaborative tasks. *The VLDB Journal*, 28(1):1–23, 2019.

[22] F. Rodrigues, F. Pereira, and B. Ribeiro. Gaussian process classification and active learning with multiple annotators. 32(2):433–441, 2014.

[23] S. B. Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Task assignment optimization in knowledge-intensive crowdsourcing. *VLDB J.*, 24(4):467–491, 2015.

[24] C. P. G. Runzhe Yang, Yexiang Xue. Pedagogical value-aligned crowdsourcing: Inspiring the wisdom of crowds via interactive teaching. *International Conference on Autonomous Agents and Multiagent Systems(AAMAS)*, 2018.

[25] S. S. Sara C. Kingsley, Mary L. Gray. Monopsony and the crowd: Labor for lemons? In *Proceedings of the Internet, Policy & Politics Conference (IPP2014)*, 2014.

[26] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[27] A. Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations*. McMaster University Archive for the History of Economic Thought, 1776.

[28] R. Suzuki, N. Salehi, M. S. Lam, J. C. Marroquin, and M. S. Bernstein. Atelier: Repurposing expert crowdsourcing tasks as micro-internships. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2645–2656, 2016.

[29] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi. Spatial crowdsourcing: a survey. *VLDB J.*, 29(1):217–250, 2020.

[30] M. A. Valentine, D. Retelny, A. To, N. Rahmati, T. Doshi, and M. S. Bernstein. Flash organizations: Crowdsourcing complex work by structuring crowds as organizations. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 3523–3537, 2017.

[31] Y. Yan, R. Rosales, G. Fung, and J. G. Dy. Active learning from crowds. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1161–1168, 2011.

[32] C. Zhang and K. Chaudhuri. Active learning from weak and strong labelers. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 703–711. Curran Associates, Inc., 2015.