1-S

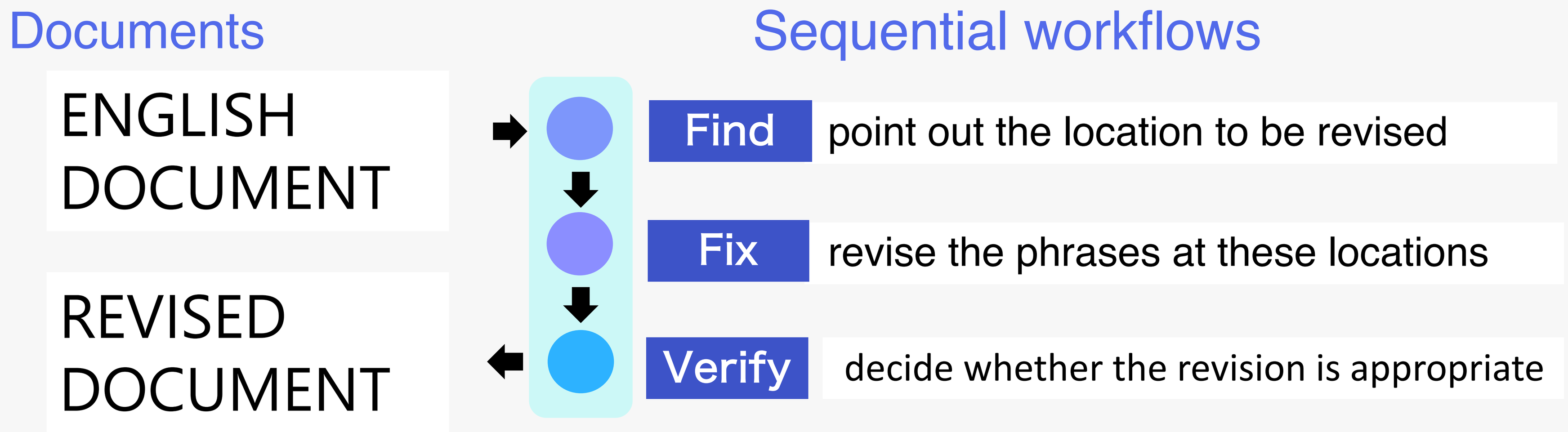
Efficient Pipeline Processing of Crowdsourcing Workflows

Ken Mizusawa[†], Keishi Tajima[‡], Masaki Matsubara[†], Toshiyuki Amagasa[†], Atsuyuki Morishima[†]

† 筑波大学 University of Tsukuba ‡ 京都大学 KYOTO UNIVERSITY JST CREST CyborgCrowd Immediate, United, Optimized

Background

Pipeline Crowdsourcing



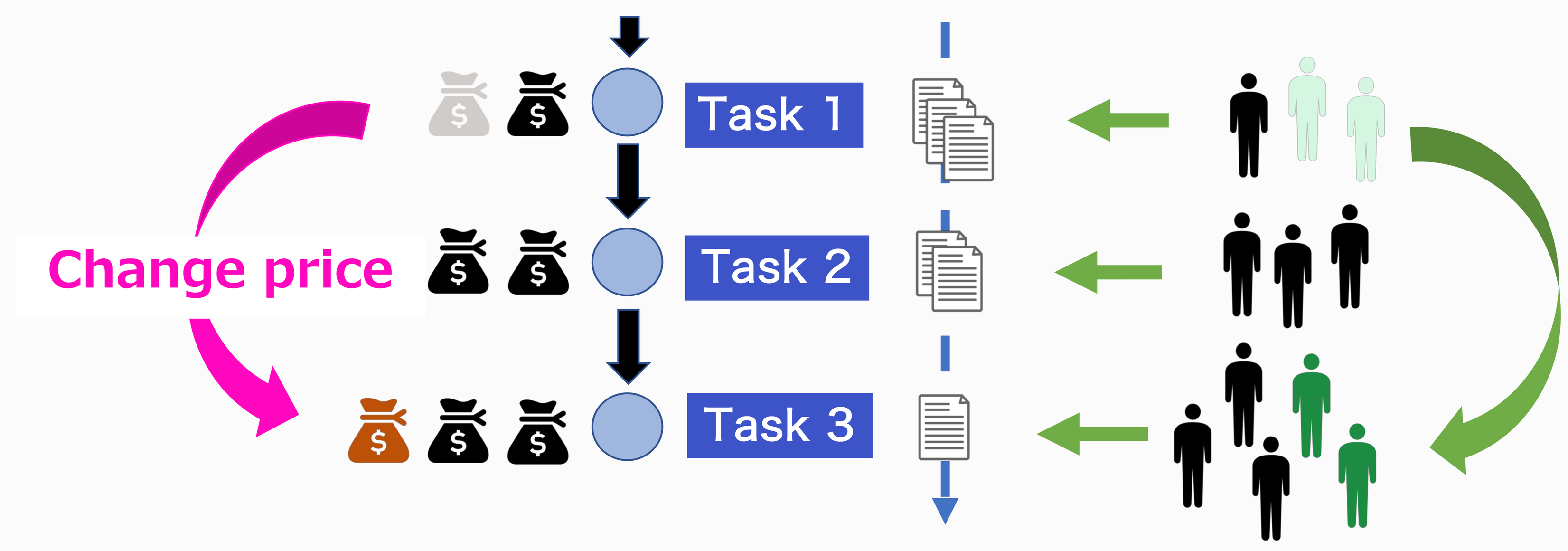
Using workflow to crowdsource a task of revising a document

The problems handled by crowdsourcing are becoming more complex, and we need to use workflows that involve more than one type of subtasks with dataflow among them

Bernstein, Michael S., et al. "Soylent: a word processor with a crowd inside." *Communications of the ACM* 58.8 (2015): 85-94.

Approache

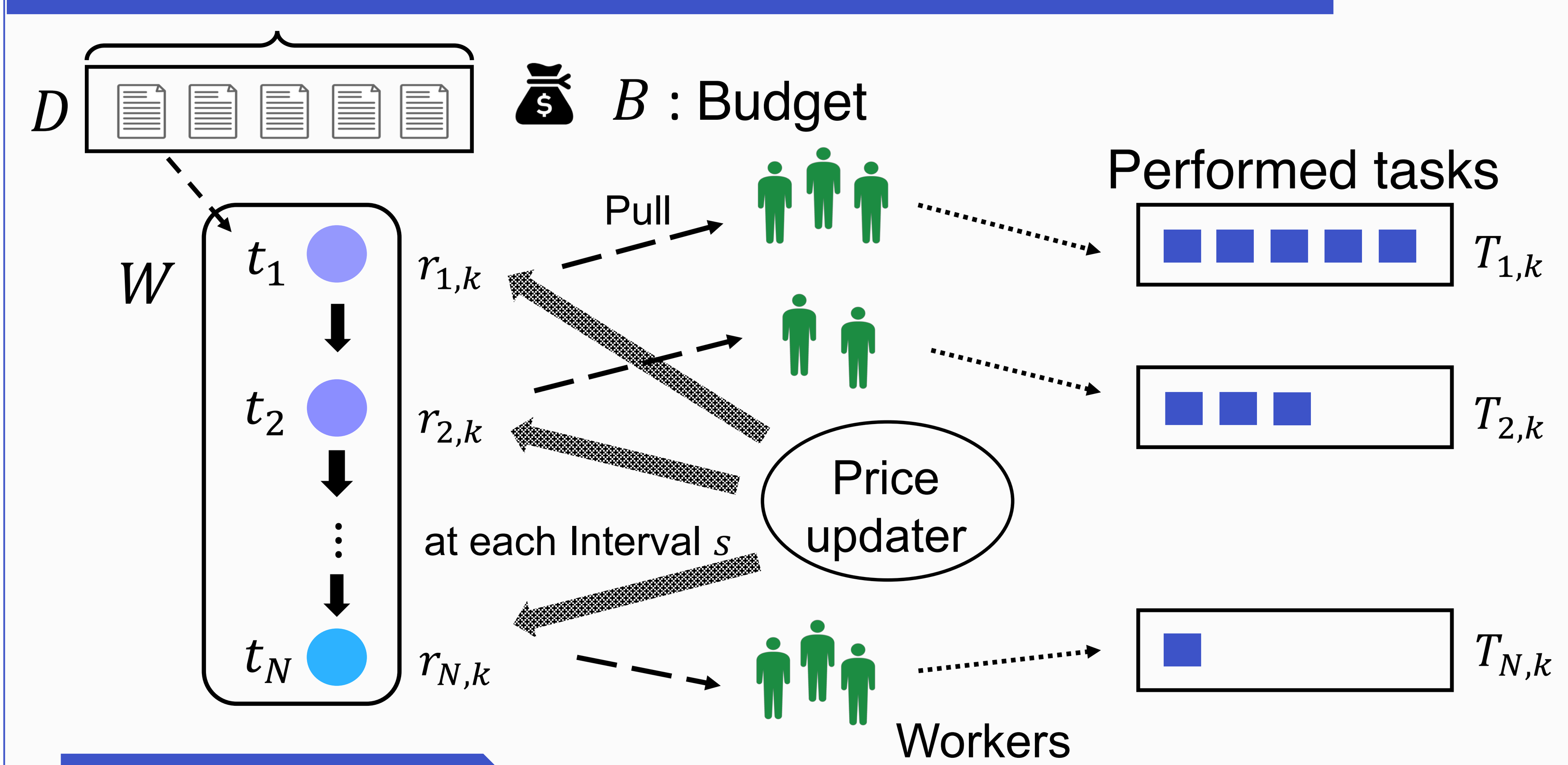
Pipeline processing of workflows with price control



Our approach is to control the budget distribution to subtasks in order to balance the execution speed of the subtasks and to improve throughput of overall sequential workflows.

Proposed Framework

Dynamically changing the price setting according to the progress of the workflow



Initial Setting

We assign the same amount of budget B/D to each of the D data items, and distribute them equally to the N tasks in W

$$r_{i,0} = \frac{B}{D \cdot N}$$

Updating Prices

Let $r_{i,k}$ be the price. Let $T_{i,k}$ be the number of task instances of t_i that workers have already completed at time $k \cdot s$.

$$r'_{i,k} = \left(\frac{D - T_{i,k}}{\sum_{j=1}^{N} D - T_{i,k}}\right)^p$$

we assign larger weights to tasks with more task instances that have not yet been performed.

Parameter p controls the degree of the influence over $r'_{i,k}$

We normalize $r'_{i,k}$ so that $\sum_1^N r'_{i,k} = 1$, and compute $r_{i,k}$ $r_{i,k} = r_{i,0} \cdot r'_{i,k}$

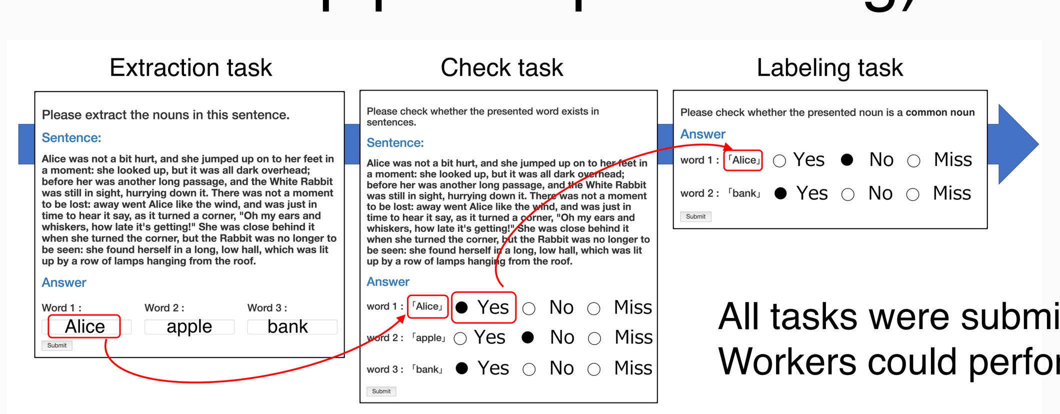
Processing Flow

Given the inputs, the framework generates tasks for t_i and updates the prices for the tasks according to the progress of execution. The framework monitors the progress status of the tasks at each interval s, and computes the price $r_{i,k}$ for each t_i for the k-th interval according to their status at that time.

Experiment

Dynamic pricing was up to 1.8 times faster on average than stepwise batch execution with fixed prices

Comparing four methods (one stepwise batch execution and three pipeline processing) in a sequential workflow



All tasks were submitted via Amazon Mechanical Turk. Workers could perform only one task instance for each task.

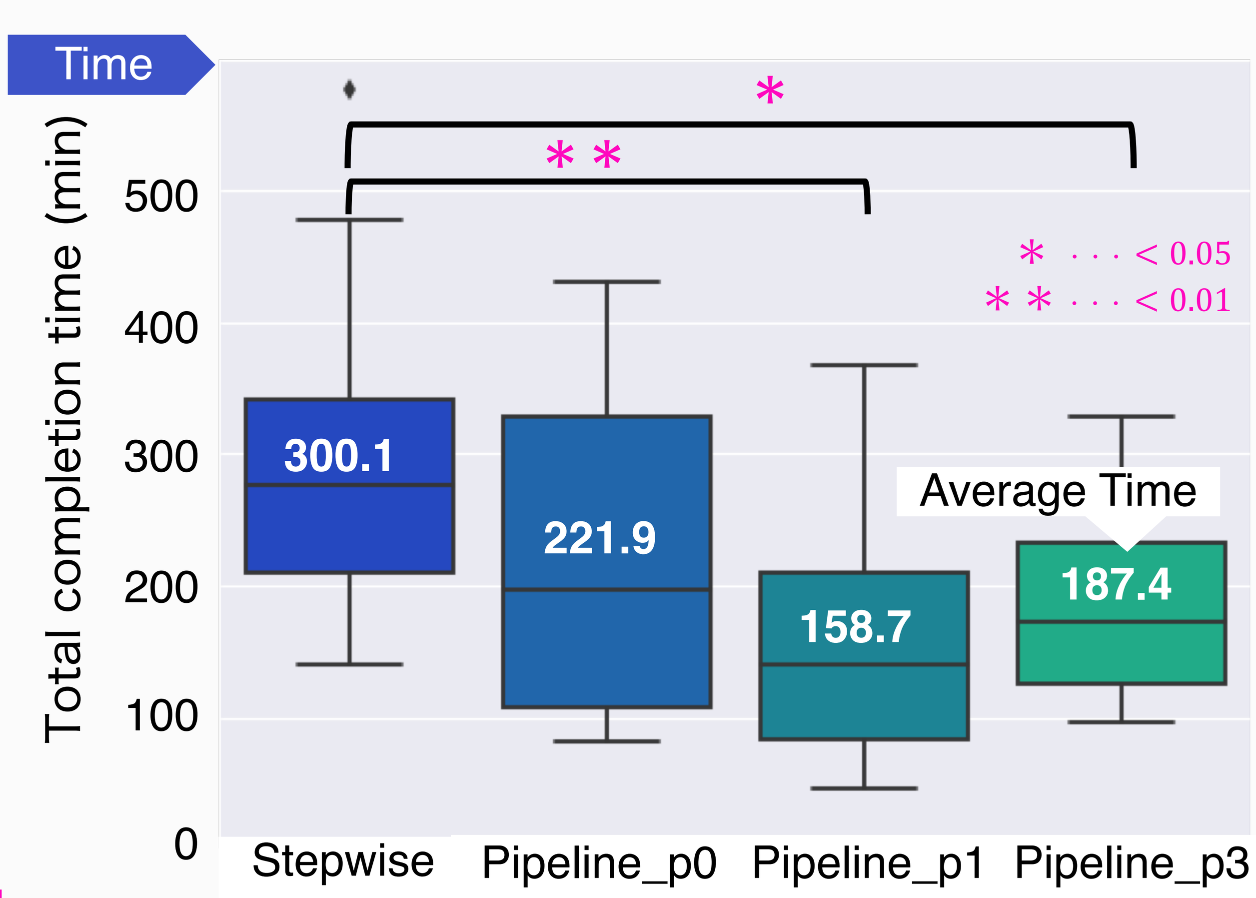
We ran the workflow 14 times with each scheme.

Method	Stepwise	Pipeline		
Name	**Stepwise**	**Change_p0**	**Change_p1**	**Change_p3**
Control	no	no	yes	yes
Parameter	-	p = 0	p = 1	p = 3

Proposed Method

#tasks in one process: 150 (=3×50) Budget: $ 6.00

Result Box plot of total completion time of each method



Multiple comparison tests with Bonferroni correction revealed significant differences between Stepwise and Pipeline_p1 (p<.01), and between Stepwise and Pipeline_p3 (p<.05) in the total processing time.

Cost

About Total cost
Average and maximum
p1 : $5.51 , $5.70
p3 : $5.62 , $5.85
(complete under budget)

Quality

Accuracy of the four methods was not statistically significant

81%, 82%, 86%, and 82% for Stepwise, Pipeline_p0, _p1, and _p3,respectively