Complex Tasks Workflow Design

Rui Jin, Quentin Wang, Lingxu Zhang

Introduction

- Opportunity
  - Mechanical Turk (MTurk) provides an on-demand source of human computation. This provides a tremendous opportunity to explore algorithms which incorporate human computation as a function call.
- Difficulty
  - Various systems challenges make this difficult in practice.
  - For now, MTurk is used almost exclusively for independent tasks.
- Solution?
  - Instead of independent, parallel tasks, MTurk could support iterative, sequential tasks.
  - We could write algorithms dictating the flow of human computation to achieve larger goals.

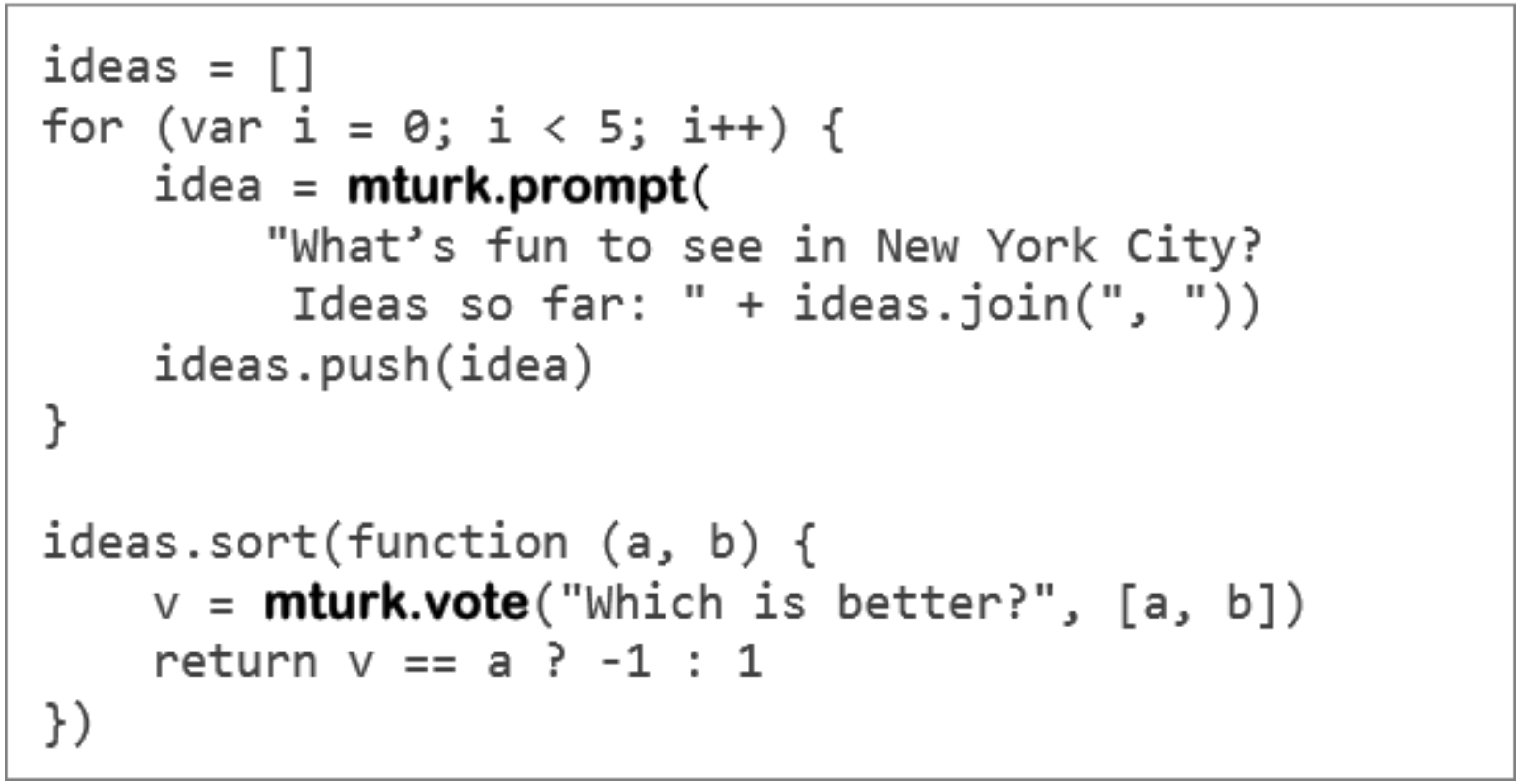
Challenges – in brief

- The tasks on MTurk
  - take time to complete
  - cost money to create

which makes programming complicated workflows more difficult, while at the same time making reliability more important.

TurKit

A toolkit for prototyping and exploring algorithmic human computation.



```
ideas = []
for (var i = 0; i < 5; i++) {
    idea = mturk.prompt(
        "What’s fun to see in New York City?
         Ideas so far: " + ideas.join(", "))
    ideas.push(idea)
}

ideas.sort(function (a, b) {
    v = mturk.vote("Which is better?", [a, b])
    return v == a ? -1 : 1
})
```

What’s special?

- TurKit incorporates a unique Crash-and-Rerun programming model to help overcome these systems challenges.
  - In this model, a program can be executed many times, without repeating costly work. Crash-and-Rerun allows the programmer to write imperative programs in which calls to MTurk appear as ordinary function calls, so that programmers can leverage their existing programming skills.

Crash-and-Rerun

- A method for allowing a script to be re-executed without re-running costly side-effecting functions.
  - TurKit uses TurKit Script, which is an extension of JavaScript that introduces functions for interacting with the MTurk platform.
  - An important functions is *waitForHIT*, which allows a script to wait until a HIT is completed.

- Web servers typically generate HTML for the user and then “crash” (forget their state) until the next request.

Let’s be concrete

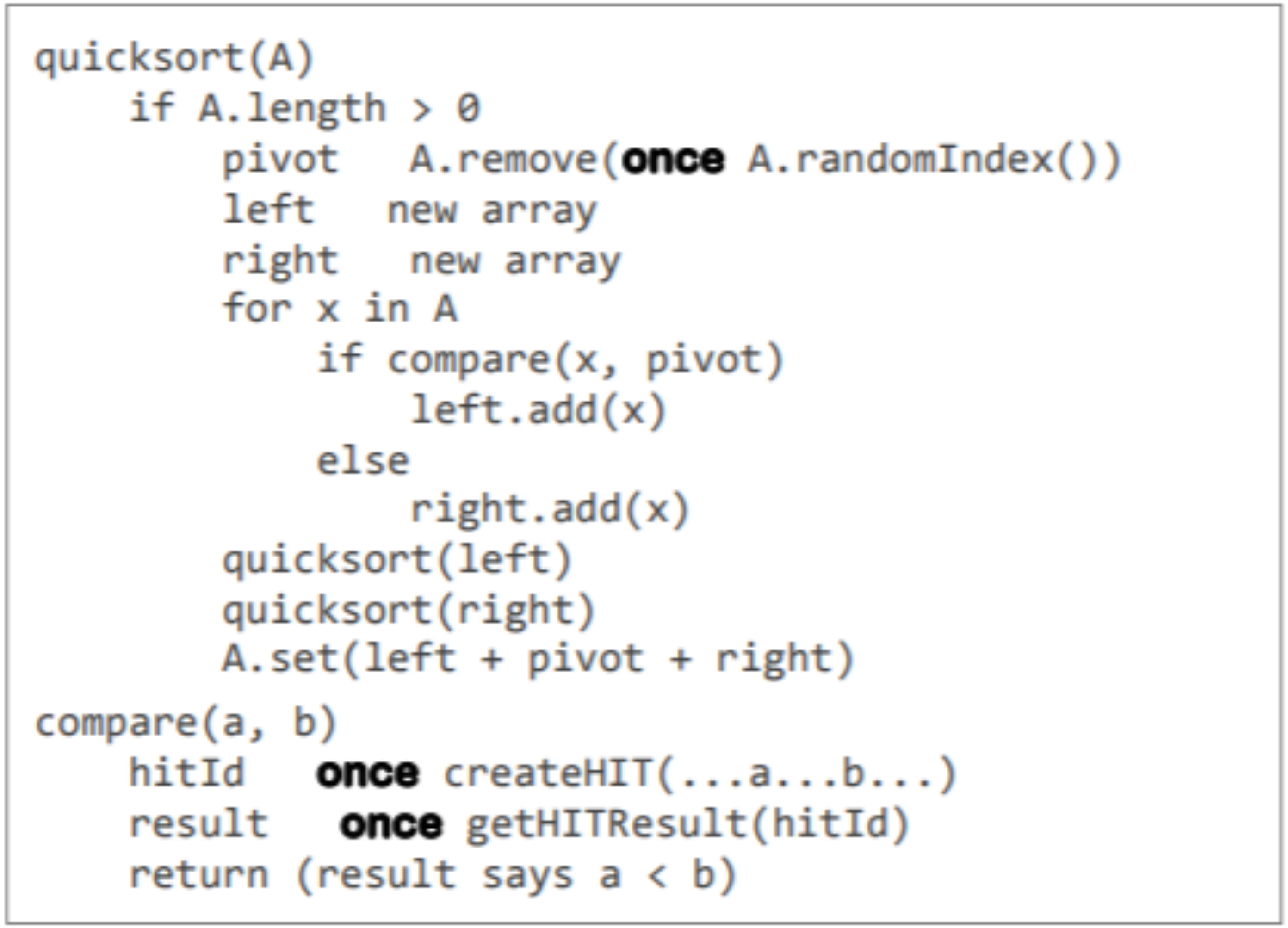
- Local computation is cheap
- External calls cost money and must wait for humans to complete work

- What can they do then?
  - If their program crashes, it is cheap to rerun the entire program up to the place it crashed, since local computation is cheap.
  - They do not re-perform all of the costly external operations from the previous run. To achieve this, they record information in a database every time a costly operation is executed.

once function

- Address these issues:
  - Non-determinism.
    - Wrapping non-deterministic calls in once ensures that their outcomes are the same in all subsequent runs of the program (e.g. *once Math.random()*).
  - High cost.
    - If a function is expensive (in terms of time or money), then it is important to wrap it in *once* so that the program only pays that cost the first time the program encounters the function call.
  - Side-effects.
    - For instance, approving results from a HIT multiple times causes an error from MTurk.

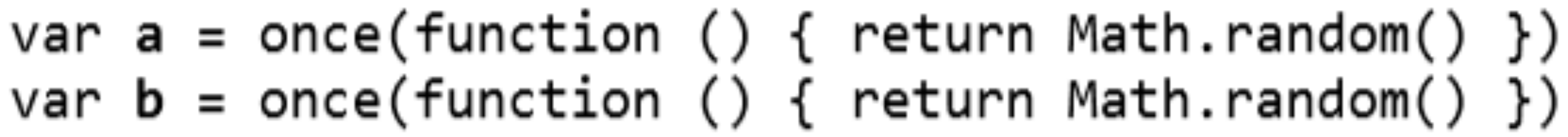
Adding *once* to the quicksort algorithm by marking the non-deterministic random pivot selection, as well as the expensive MTurk calls.



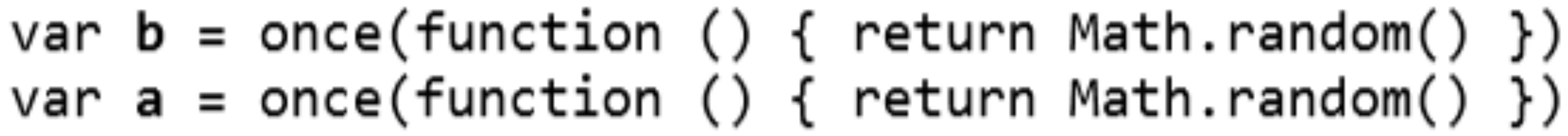
```
quicksort(A)
    if A.length > 0
        pivot   A.remove(once A.randomIndex())
        left   new array
        right   new array
        for x in A
            if compare(x, pivot)
                left.add(x)
            else
                right.add(x)
        quicksort(left)
        quicksort(right)
        A.set(left + pivot + right)
compare(a, b)
    hitId   once createHIT(...a...b...)
    result   once getHITResult(hitId)
    return (result says a < b)
```

Implementation - a limit

- The user is alerted if a change is detected in the sequence of once calls. Unfortunately, TurKit cannot detect all function re-orderings.
- If the first version assigns 5 to a and 7 to b, then re-running with the second version will assign 5 to b and 7 to a.



```
var a = once(function () { return Math.random() })
var b = once(function () { return Math.random() })
```



```
var b = once(function () { return Math.random() })
var a = once(function () { return Math.random() })
```

fork and *join*

- *fork*
  - creates a new branch in the recorded execution trace
  - useful in cases where a user wants to run several processes in parallel. For instance, they may want to post multiple HITs on MTurk at the same time, and have the script make progress on whichever path gets a result first.

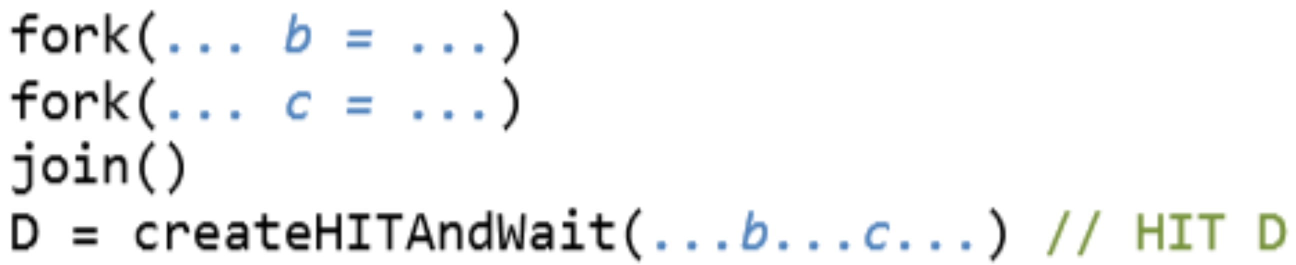
```
a = createHITAndWait()          // HIT A
b = createHITAndWait(...a...)  // HIT B
c = createHITAndWait()          // HIT C
```

$\Longrightarrow$

```
fork(function () {
    a = createHITAndWait()          // HIT A
    b = createHITAndWait(...a...)  // HIT B
})
fork(function () {
    c = createHITAndWait()          // HIT C
})
```

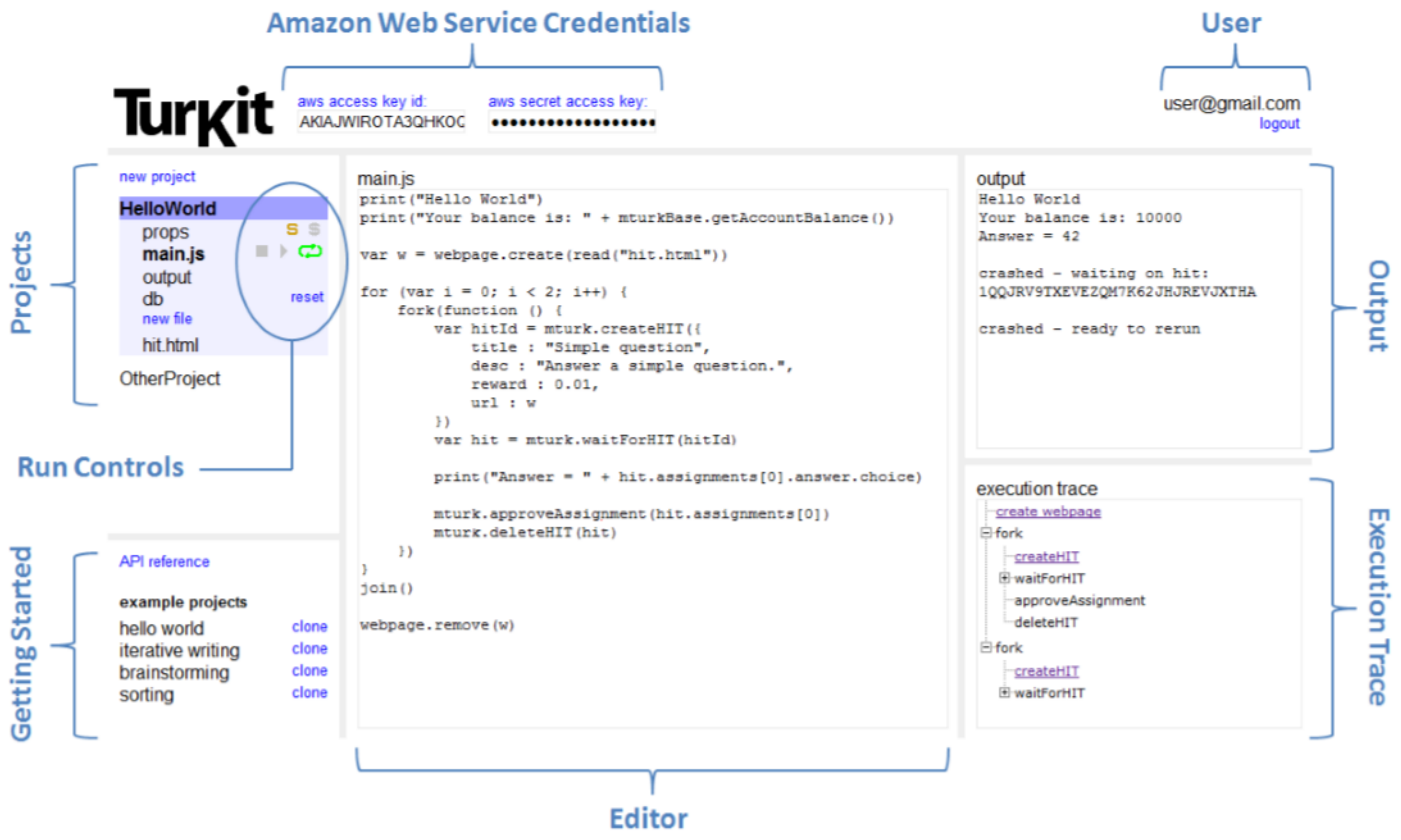
fork and *join*

- *join*
  - ensures that a series of forks have all finished.



```
fork(... b = ...)
fork(... c = ...)
join()
D = createHITAndWait(...b...c...) // HIT D
```

Web user interface



main.js

```
print("Hello World")
print("Your balance is: " + mturkBase.getAccountBalance())

var w = webpage.create(read("hit.html"))

for (var i = 0; i < 2; i++) {
    fork(function () {
        var hitId = mturk.createHIT({
            title : "Simple question",
            desc : "Answer a simple question.",
            reward : 0.01,
            url : w
        })
        var hit = mturk.waitForHIT(hitId)

        print("Answer = " + hit.assignments[0].answer.choice)

        mturk.approveAssignment(hit.assignments[0])
        mturk.deleteHIT(hit)
    })
}
join()

webpage.remove(w)
```

output

```
Hello World
Your balance is: 10000
Answer = 42

crashed - waiting on hit:
1QQJRV9TXEVEZQM7K62JHJREVJXTHA

crashed - ready to rerun
```

execution trace

```
create webpage
⊟ fork
    createHIT
  ⊞ waitForHIT
    approveAssignment
    deleteHIT
⊟ fork
    createHIT
  ⊞ waitForHIT
```

Basic functions

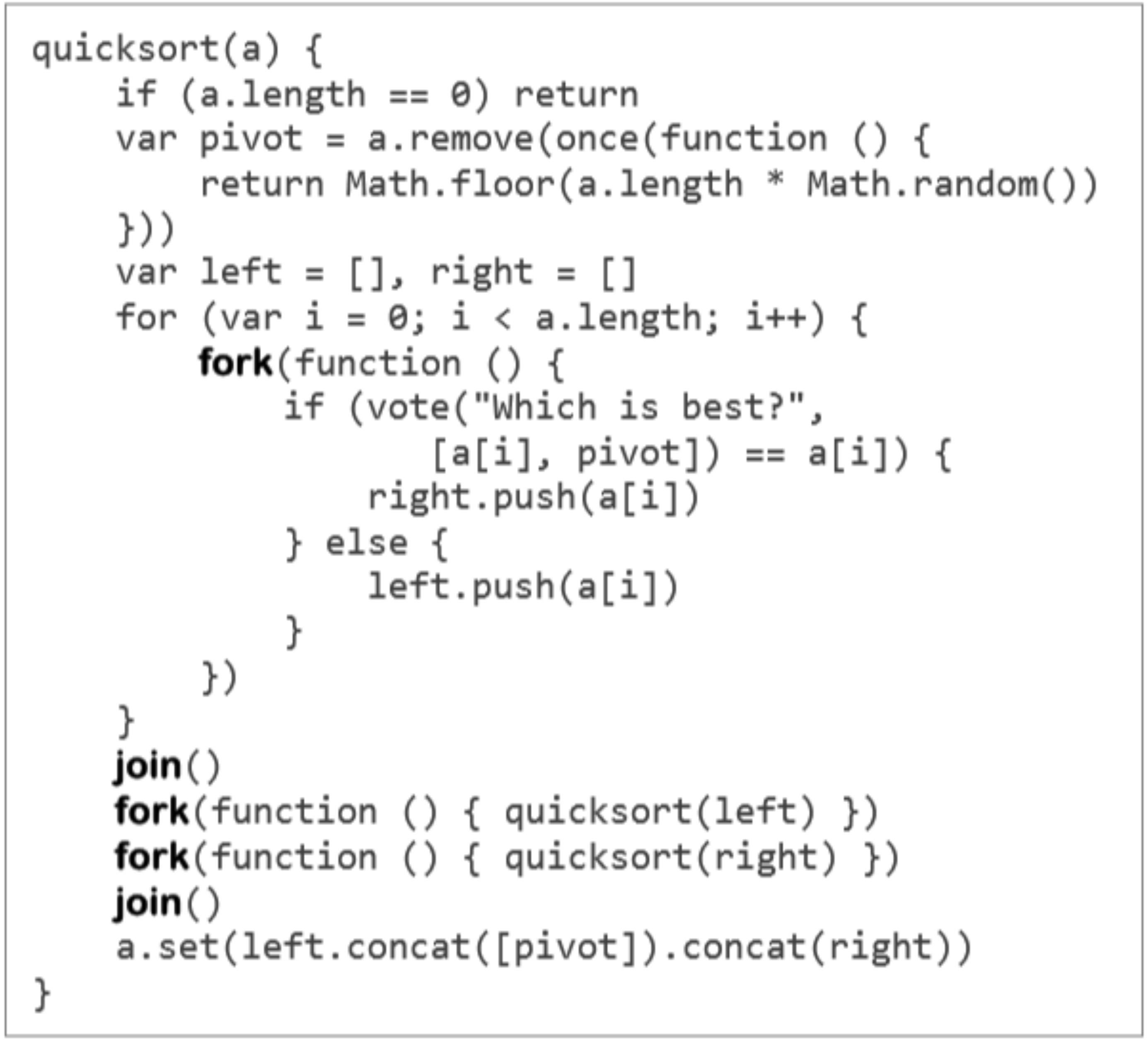
Voting

```
function vote(message, options) {
    // create comparison HIT
    var h = mturk.createHITAndWait({
        ...message...options...
        assignments : 3})

    // get enough votes
    while (...votes for best option < 3...) {
        mturk.extendHIT(...add assignment...)
        h = mturk.waitForHIT(h)
    }

    // cleanup and return
    mturk.deleteHIT(h)
    return ...best option...
}
```

Sorting

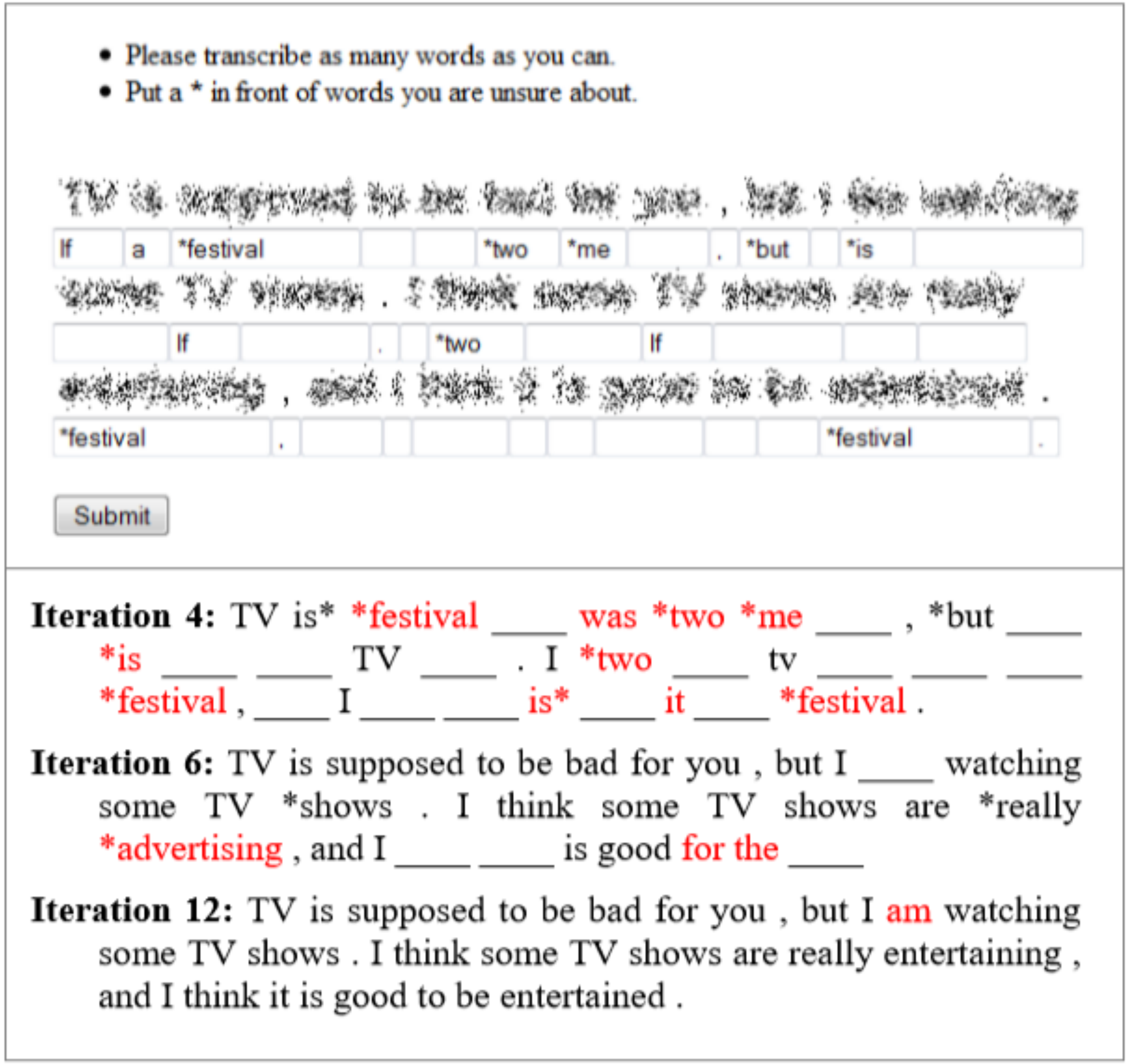
```
ideas.sort(function (a, b) {
    v = mturk.vote("Which is better?", [a, b])
    return v == a ? -1 : 1
})
```

Example – Quicksort



```
quicksort(a) {
    if (a.length == 0) return
    var pivot = a.remove(once(function () {
        return Math.floor(a.length * Math.random())
    }))
    var left = [], right = []
    for (var i = 0; i < a.length; i++) {
        fork(function () {
            if (vote("Which is best?",
                    [a[i], pivot]) == a[i]) {
                right.push(a[i])
            } else {
                left.push(a[i])
            }
        })
    }
    join()
    fork(function () { quicksort(left) })
    fork(function () { quicksort(right) })
    join()
    a.set(left.concat([pivot]).concat(right))
}
```

Example – Blurry Text Recognition



- Please transcribe as many words as you can.
- Put a * in front of words you are unsure about.

If	a	*festival			*two	*me		,	*but		*is	

	If		.		*two		If			

*festival	,									*festival	.

Submit

Iteration 4: TV is* *festival ____ was *two *me ____ , *but ____ *is ____ ____ TV ____ . I *two ____ tv ____ ____ ____ *festival , ____ I ____ ____ is* ____ it ____ *festival .

Iteration 6: TV is supposed to be bad for you , but I ____ watching some TV *shows . I think some TV shows are *really *advertising , and I ____ ____ is good for the ____

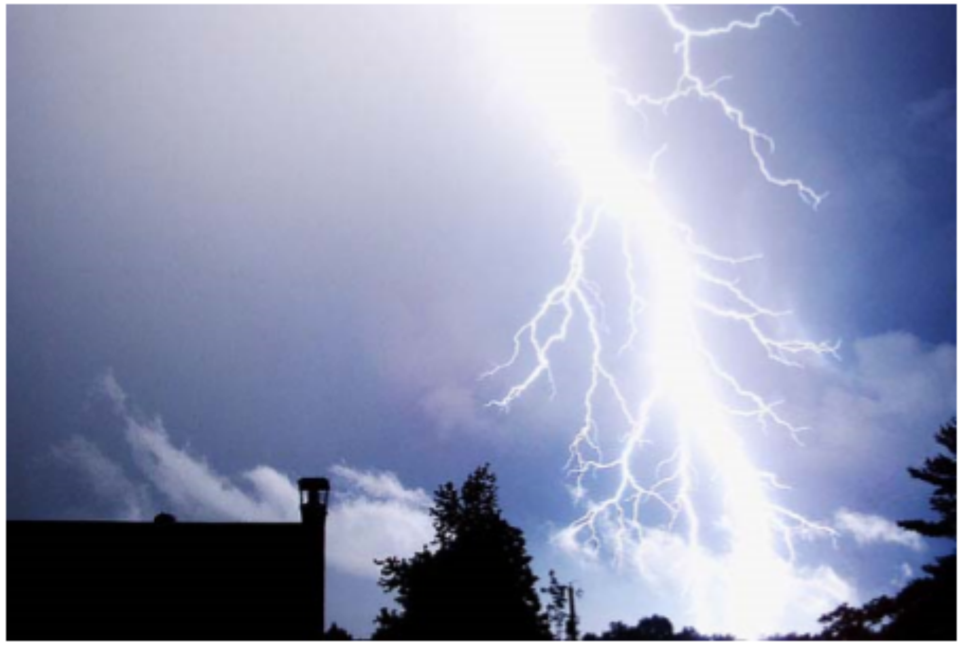
Iteration 12: TV is supposed to be bad for you , but I am watching some TV shows . I think some TV shows are really entertaining , and I think it is good to be entertained .

Discussion

- What are the challenges that crush-and-rerun and once function aim to conquer? Try to come up with a task that can be solved using TurKit.
  - You can add your own functions!
  - Examples include making a road trip plan, iteratively revising an article etc.

- For a complex job, what's the advantage of dividing the tasks through crowdsourcing tools, compared with handing the tasks over to a small group of experienced people?

Example – Iterative Writing



Iteration 1: Lightening strike in a blue sky near a tree and a building.

Iteration 2: The image depicts a strike of fork lightening, striking a blue sky over a silhoutted building and trees. (4/5 votes)

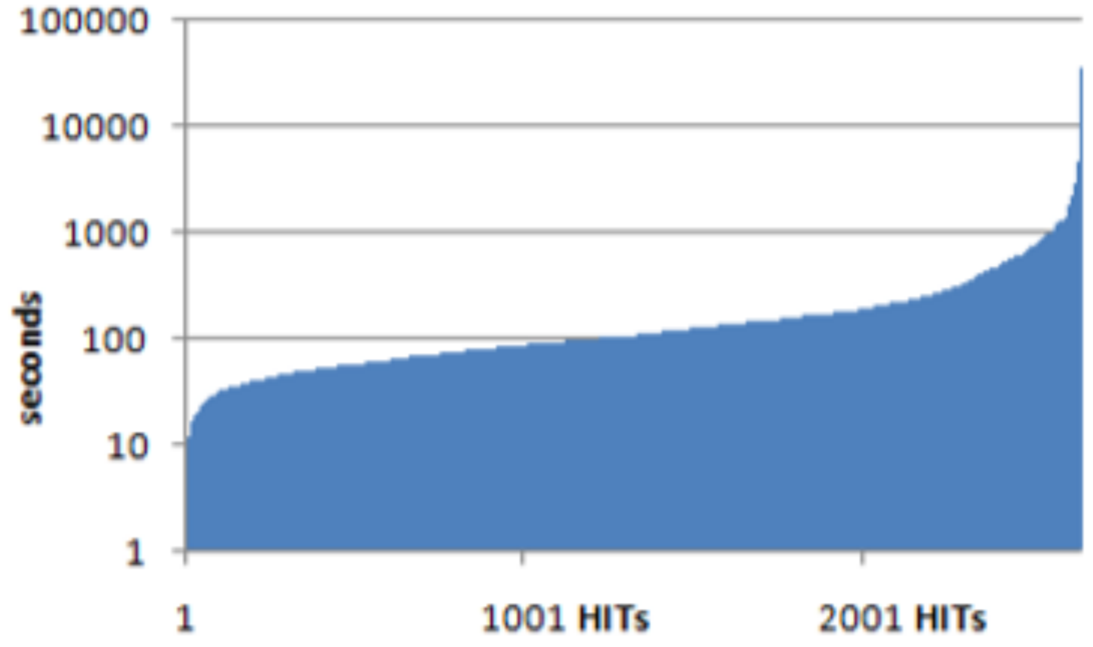
Iteration 3: The image depicts a strike of fork lightning, against a blue sky with a few white clouds over a silhouetted building and trees. (5/5 votes)

Iteration 4: ~~The image depicts a strike of fork lightning, against a blue sky wonderful capture of the nature.~~ (1/5 votes)

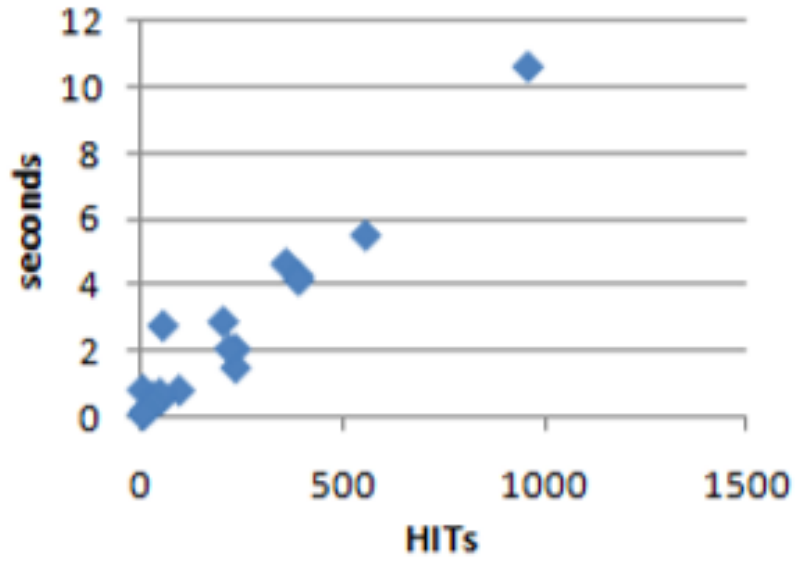
Iteration 5: This image shows a large white strike of lightning coming down from a blue sky with the tops of the trees and rooftop peaking from the bottom. (3/5 votes)

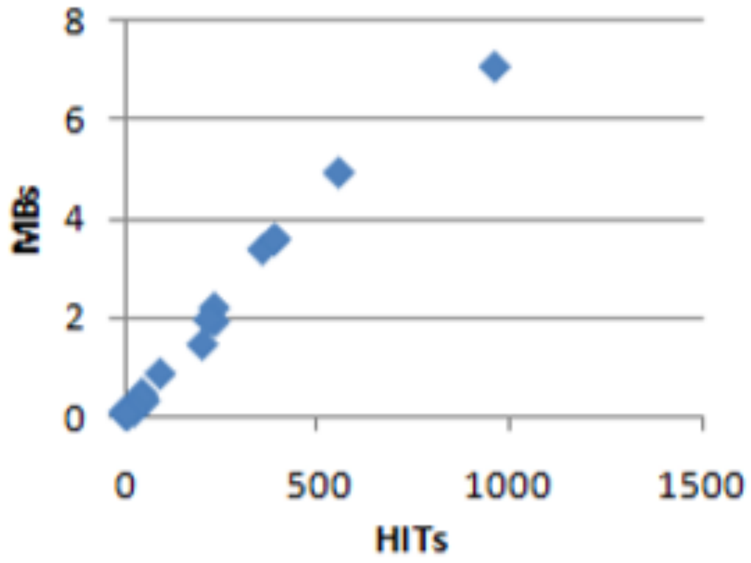
Iteration 6: This image shows a large white strike of lightning coming down from a blue sky with the silhouettes of tops of the trees and rooftop peeking from the bottom. The sky is a dark blue and the lightening is a contrasting bright white. The lightening has many arms of electricity coming off of it. (4/5 votes)

Evaluation



Time until the first assignment is completed.





Time and space requirements for 20 TurKit scripts.

Insights

- Crash-and-Rerun Benefits 😀
  - Incremental programming
  - Easy to implement
  - Retroactive print-line-debugging

😕

- Usability
  - When to be deterministic?
  - Can it be re-ordered?
  - Parallel feature

- Exploring New Algorithms

🙄

CrowdForge:

Crowdsourcing **Complex Work**

Typical work types of crowdsourcing

- Identify objects in images or video
- Upload recordings or images of certain topic
- Researching data details

- Low skills, short time, self-contained, repetitive

Can crowdsourcing do real-world jobs?

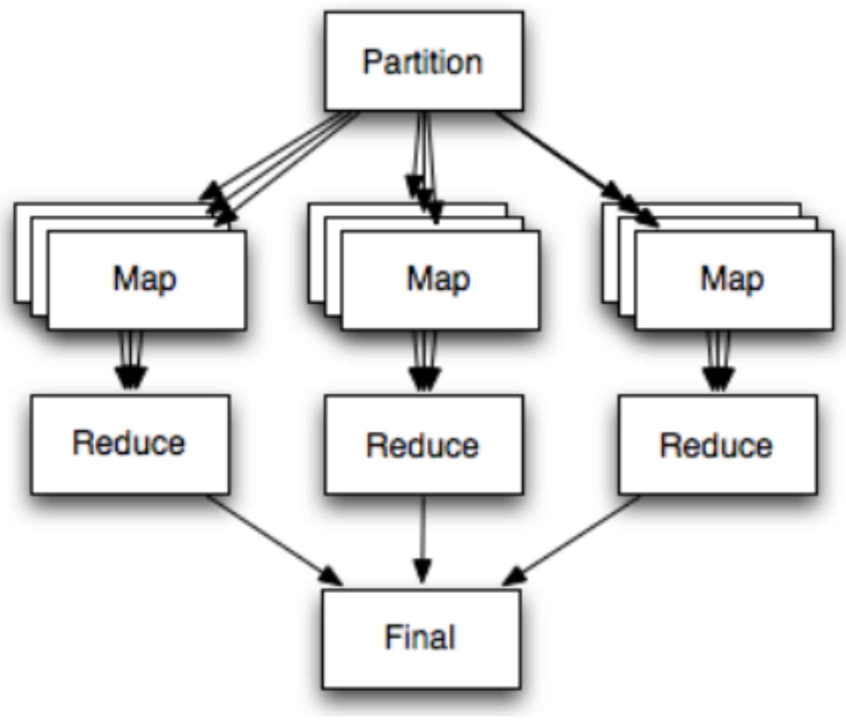
- Complex works

- Example: newspaper article
- Subtasks: Deciding structure, collect facts, writing narrative, take pictures, document layout, final edit
- Coordination between workers
- Quality: unknown skills, low commitment

CrowdForge framework and toolkit

- Partition, map, reduce

- Partition: break tasks into subtasks
- Map: Do the tasks
- Reduce: Merge the results of tasks to produce final result



Partition

- high level partitioning of the problem
- Example:
- Outline of article with section headings
- List of criteria for buying a new car

- Partition listed as a task
- Subtasks created with response of partition

Partition

- Advantage:

- Partitioner does not need to know all subtasks that will be created
- Defining the division of labor and subtask design are shifted to the market itself

Map

- Process subtasks
- Each task being as simple as “traditional” crowdsourcing ones

- Multiple workers might be asked to do the same subtask

- Example: collect facts for an article

Reduce

- Summarize subtasks and put them together

- Example: Summarize facts provided in tasks
- Choose best written line out of several identical tasks
- Write facts into paragraph

Iterative

- All these steps can be iterative

- Partition: partition a subtask into may subtasks

- Reduce: Reorder paragraphs

Example: news article with CrowdForge

- Tool: Self-developed tool utilizing MTurk indirectly
- Average cost: $3.26 per article
- Num subtasks: 36
- Partition: 5.3 topics per article

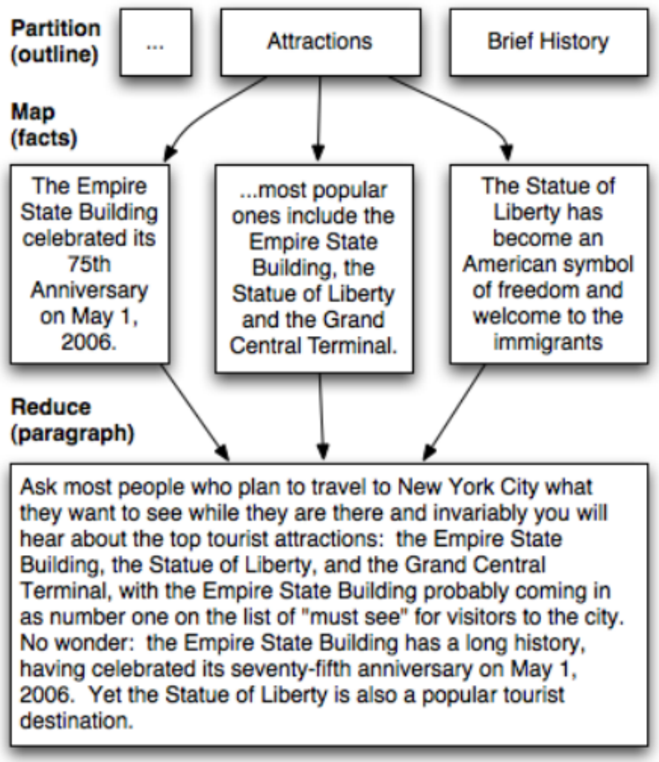

Partition (outline)
...
Attractions
Brief History
Map (facts)
The Empire State Building celebrated its 75th Anniversary on May 1, 2006.
...most popular ones include the Empire State Building, the Statue of Liberty and the Grand Central Terminal.
The Statue of Liberty has become an American symbol of freedom and welcome to the immigrants
Reduce (paragraph)
Ask most people who plan to travel to New York City what they want to see while they are there and invariably you will hear about the top tourist attractions: the Empire State Building, the Statue of Liberty, and the Grand Central Terminal, with the Empire State Building probably coming in as number one on the list of "must see" for visitors to the city. No wonder: the Empire State Building has a long history, having celebrated its seventy-fifth anniversary on May 1, 2006. Yet the Statue of Liberty is also a popular tourist destination.

Figure 2: Partial results of a collaborative writing task.

Quality control

- Partitioner greatly affects result quality
- E.g.: Bad outline cause bad article quality

- Methods:
- Map tasks to verify quality (minimize complexity)
- Reduce via voting on several outcomes of same task
- Reduce via merging several outcomes of same task (better quality)

Limitations: Research Purchase Decisions

- Partition: Given consumer condition, list aspects of car
- Partition: Give potential competitors of car

- Map: Evaluate a single cell of table

- Reduce: Consolidate with table given

- Faliure

Challenge: Turn science article to news

- Write newspaper article reporting contents of a paper
- Work with experts at partition level

Challenge and Solution

- Challenge:
- Summarize paper in one sentence
- Describe research procedures

- Solution:
- Present workers with consolidated info (abstract)
- Provide examples

Result

- Quality: Not the best, but quite acceptable
- Can be improved with editors

- No expertise
- Limited time and effort
- Limited context

Limitations

- Iterations or recursion in CrowdForge require stronger tools
  - Task designer need to specify each stage with current tools
  - Solution: e.g. vote to see if another stage is required
- Some works might be hard to be partitioned
  - Summarize experiment procedures require understanding
  - Partition and reduce may cost more than they are worth
- Subtasks may be dependent
  - Provide other workers' responses
  - Problem: Repeating bias

Discussion

- Using crowdsourcing tools to carry out a complex job may include task division, merging, verification and so on.
  What are the possible consequences if these tasks are all given to the workers without monitoring the workflow?

- Is the risk of task division in each stage the same?

- Is there any stage of task division has a greater impact on the overall work effect?

Turkomatic

- Turkomatic is a crowdsourcing tool that allows the crowd to collaboratively design and execute workflows in conjunction with a requester.

- The requester is able to monitor and edit the resulting workflows as they are produced.

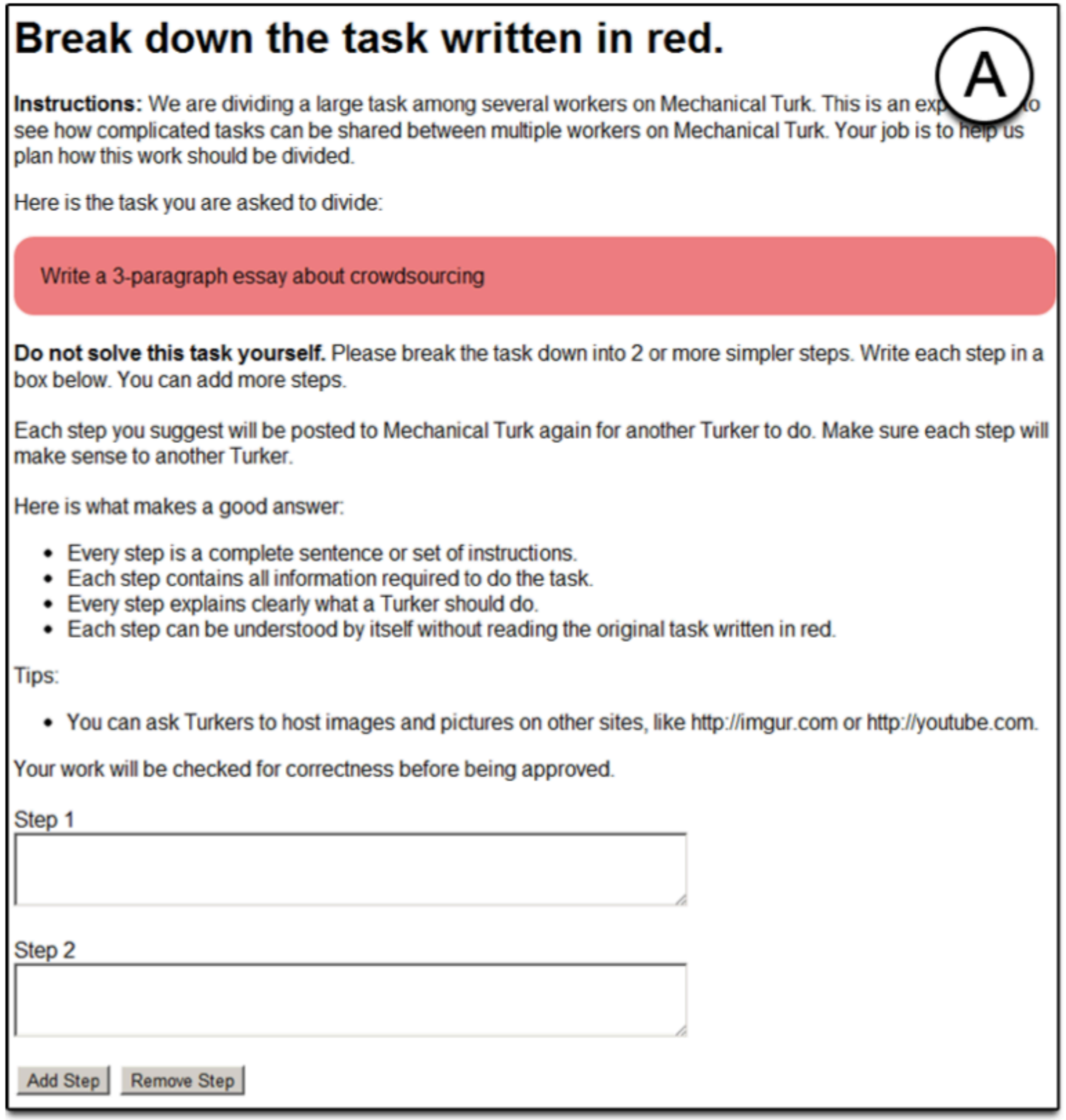
- Turkomatic shows the task context to workers to provide them with a birds-eye view of the overall decomposition .

- **Divide, Solve, Merge, Verify.** (The worker interfaces)

Example

- For the task : **Write a 3-paragraph essay about crowdsourcing**

(a) Divide



Break down the task written in red.

Instructions: We are dividing a large task among several workers on Mechanical Turk. This is an exp to see how complicated tasks can be shared between multiple workers on Mechanical Turk. Your job is to help us plan how this work should be divided.

Here is the task you are asked to divide:

Write a 3-paragraph essay about crowdsourcing

Do not solve this task yourself. Please break the task down into 2 or more simpler steps. Write each step in a box below. You can add more steps.

Each step you suggest will be posted to Mechanical Turk again for another Turker to do. Make sure each step will make sense to another Turker.

Here is what makes a good answer:

- Every step is a complete sentence or set of instructions.
- Each step contains all information required to do the task.
- Every step explains clearly what a Turker should do.
- Each step can be understood by itself without reading the original task written in red.

Tips:

- You can ask Turkers to host images and pictures on other sites, like http://imgur.com or http://youtube.com.

Your work will be checked for correctness before being approved.

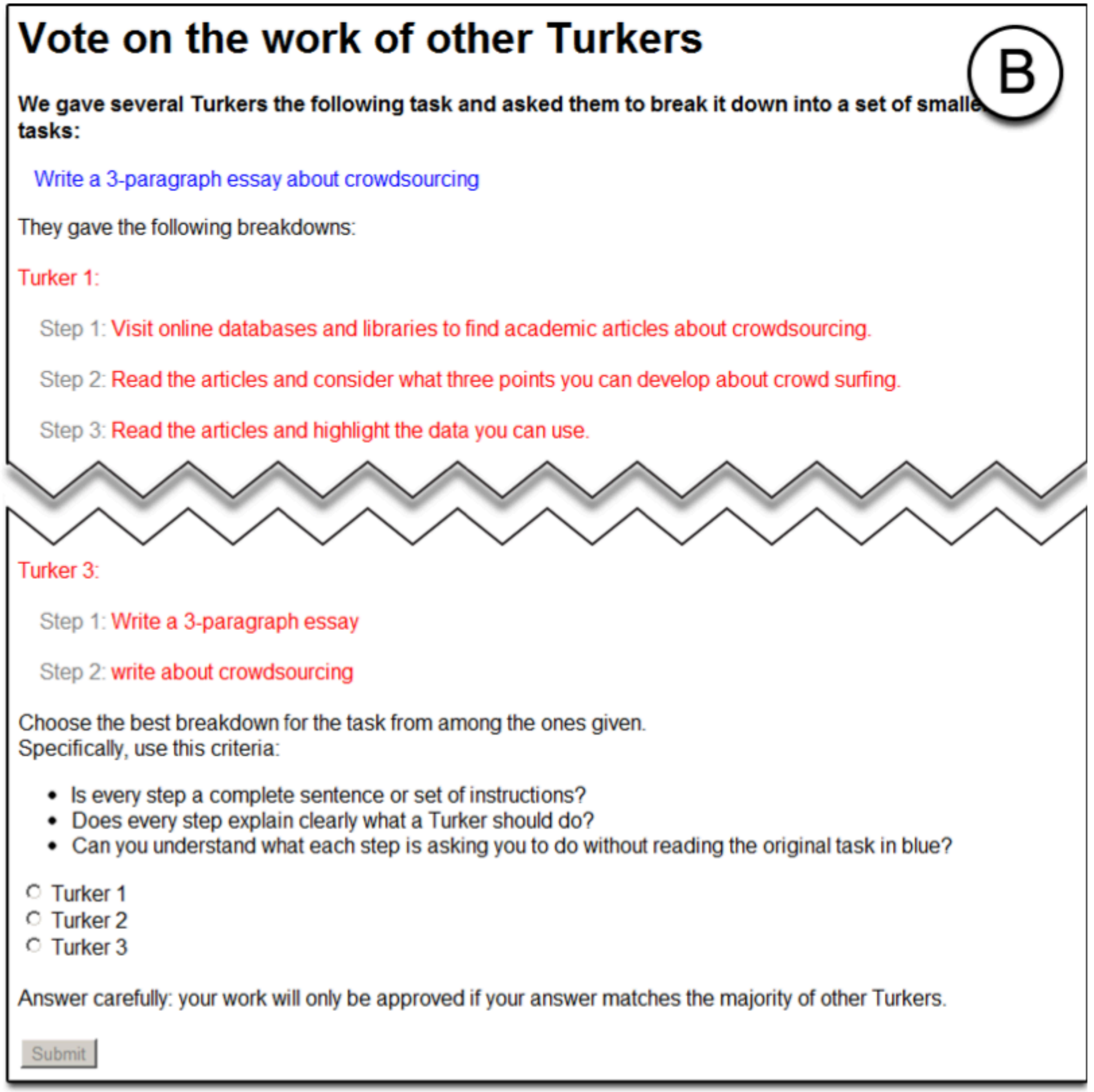
Step 1

Step 2

Example

- For the task : **Write a 3-paragraph essay about crowdsourcing**

(b) Verify



B

Vote on the work of other Turkers

We gave several Turkers the following task and asked them to break it down into a set of smalle tasks:

Write a 3-paragraph essay about crowdsourcing

They gave the following breakdowns:

Turker 1:

Step 1: Visit online databases and libraries to find academic articles about crowdsourcing.

Step 2: Read the articles and consider what three points you can develop about crowd surfing.

Step 3: Read the articles and highlight the data you can use.

Turker 3:

Step 1: Write a 3-paragraph essay

Step 2: write about crowdsourcing

Choose the best breakdown for the task from among the ones given.
Specifically, use this criteria:

- Is every step a complete sentence or set of instructions?
- Does every step explain clearly what a Turker should do?
- Can you understand what each step is asking you to do without reading the original task in blue?

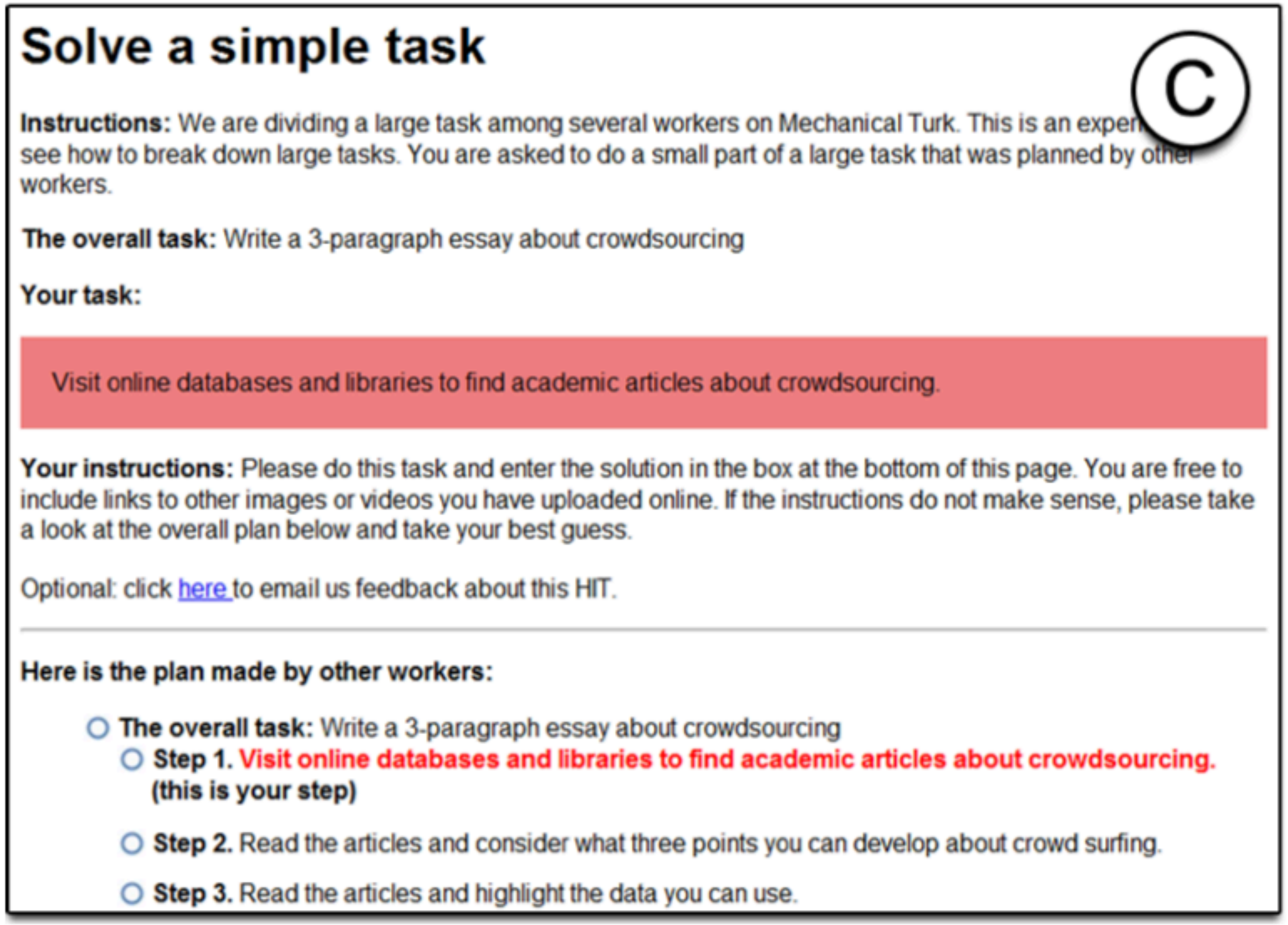
○ Turker 1
○ Turker 2
○ Turker 3

Answer carefully: your work will only be approved if your answer matches the majority of other Turkers.

Example

- For the task : **Write a 3-paragraph essay about crowdsourcing**

(c) Solve



Solve a simple task

C

Instructions: We are dividing a large task among several workers on Mechanical Turk. This is an exper see how to break down large tasks. You are asked to do a small part of a large task that was planned by other workers.

The overall task: Write a 3-paragraph essay about crowdsourcing

Your task:

Visit online databases and libraries to find academic articles about crowdsourcing.

Your instructions: Please do this task and enter the solution in the box at the bottom of this page. You are free to include links to other images or videos you have uploaded online. If the instructions do not make sense, please take a look at the overall plan below and take your best guess.

Optional: click here to email us feedback about this HIT.

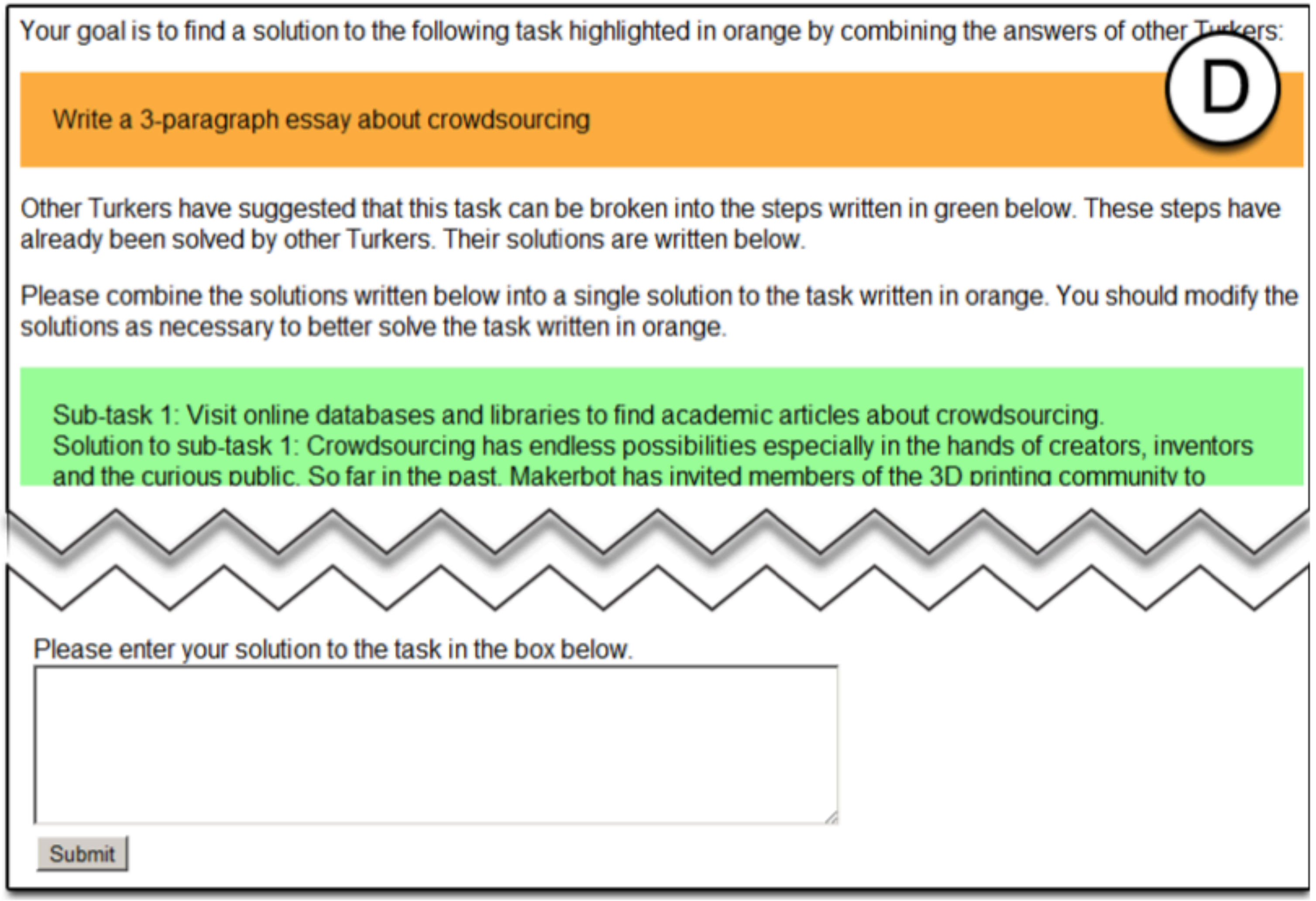
Here is the plan made by other workers:

- **The overall task:** Write a 3-paragraph essay about crowdsourcing
  - **Step 1. Visit online databases and libraries to find academic articles about crowdsourcing. (this is your step)**
  - **Step 2.** Read the articles and consider what three points you can develop about crowd surfing.
  - **Step 3.** Read the articles and highlight the data you can use.

Example

- For the task : **Write a 3-paragraph essay about crowdsourcing**

(d) Merge


Your goal is to find a solution to the following task highlighted in orange by combining the answers of other Turkers:

Write a 3-paragraph essay about crowdsourcing

Other Turkers have suggested that this task can be broken into the steps written in green below. These steps have already been solved by other Turkers. Their solutions are written below.

Please combine the solutions written below into a single solution to the task written in orange. You should modify the solutions as necessary to better solve the task written in orange.

Sub-task 1: Visit online databases and libraries to find academic articles about crowdsourcing.
Solution to sub-task 1: Crowdsourcing has endless possibilities especially in the hands of creators, inventors and the curious public. So far in the past, Makerbot has invited members of the 3D printing community to

Please enter your solution to the task in the box below.

Challenges

For instance, for the task of writing a paragraph as part of an essay-writing, one worker determined the first subtask for another worker is to “acquire a writing utensil or a computer”.

Another example. For the task of listing department chairs of top 20 Computer Science programs in the US, the final solution we got was a list of IKEA chairs.

……

Challenges

- **Task Derailment, Emergent Complexity and Cycling**
  Losing track of the original objective.

- **Task Starvation**
  After a while, no new workers attempted the available tasks, and the time limit for the experiment expired.

- **Standard Quality Assurance Techniques are Insufficient**
  Adding redundancy by asking more workers to contribute.

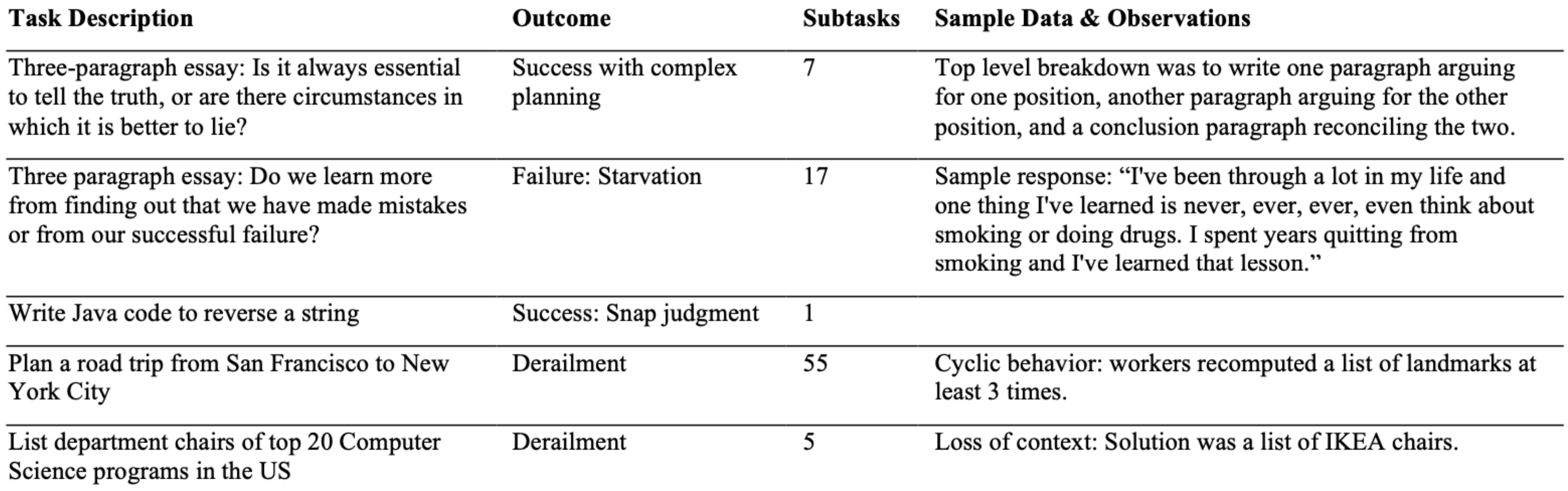
Two improved collaborative strategies

- The first strategy is to **recruit more expert workers**

- The second strategy is to **allow requesters to monitor and selectively intervene in the workflow design and execution process.**

Experiments

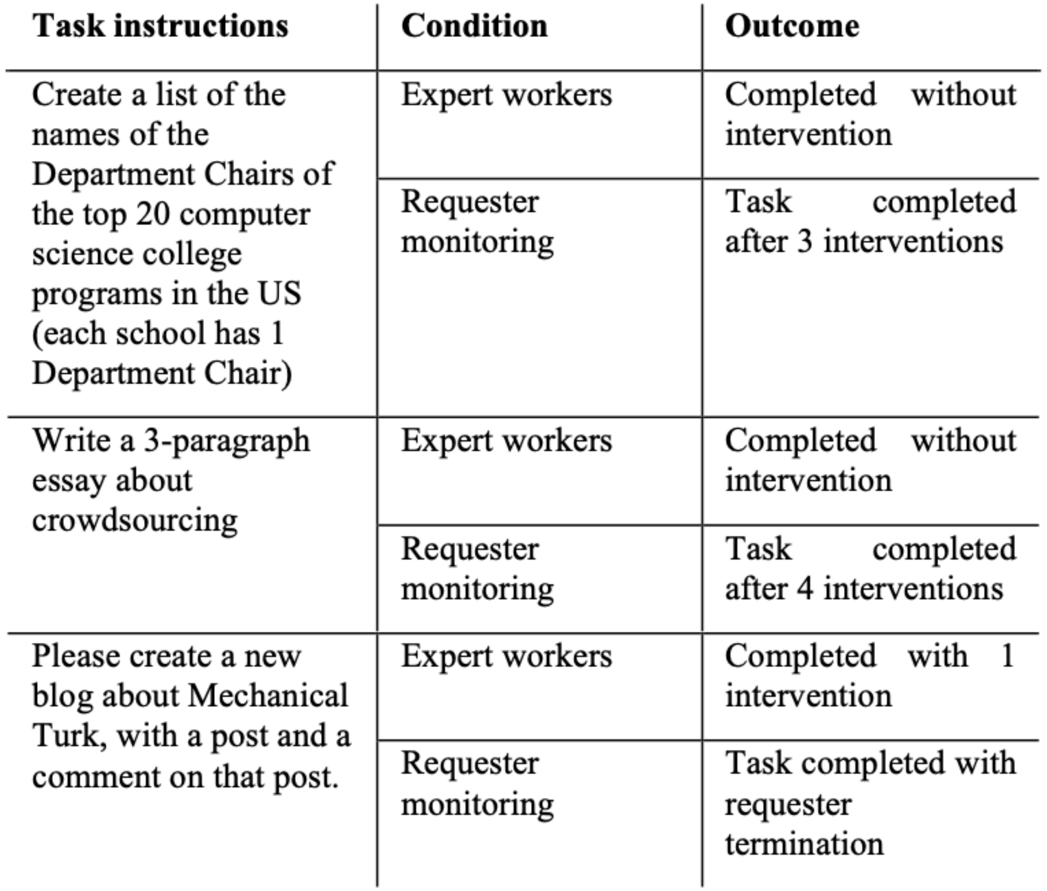
- Experiment for several unsupervised tasks (i.e. **without using the improved collaborative strategies**) :



Task Description	Outcome	Subtasks	Sample Data & Observations
Three-paragraph essay: Is it always essential to tell the truth, or are there circumstances in which it is better to lie?	Success with complex planning	7	Top level breakdown was to write one paragraph arguing for one position, another paragraph arguing for the other position, and a conclusion paragraph reconciling the two.
Three paragraph essay: Do we learn more from finding out that we have made mistakes or from our successful failure?	Failure: Starvation	17	Sample response: “I've been through a lot in my life and one thing I've learned is never, ever, ever, even think about smoking or doing drugs. I spent years quitting from smoking and I've learned that lesson.”
Write Java code to reverse a string	Success: Snap judgment	1	
Plan a road trip from San Francisco to New York City	Derailment	55	Cyclic behavior: workers recomputed a list of landmarks at least 3 times.
List department chairs of top 20 Computer Science programs in the US	Derailment	5	Loss of context: Solution was a list of IKEA chairs.

Experiments

- Experiment for the same tasks by using **Expert workers** and **Requester monitoring** :



Task instructions	Condition	Outcome
Create a list of the names of the Department Chairs of the top 20 computer science college programs in the US (each school has 1 Department Chair)	Expert workers	Completed without intervention
	Requester monitoring	Task completed after 3 interventions
Write a 3-paragraph essay about crowdsourcing	Expert workers	Completed without intervention
	Requester monitoring	Task completed after 4 interventions
Please create a new blog about Mechanical Turk, with a post and a comment on that post.	Expert workers	Completed with 1 intervention
	Requester monitoring	Task completed with requester termination

Outcome Analysis

- Both the two strategies avoided starvation and derailment and resulted in usable solutions.

- When expert workers carried out decomposition and solution of the subtasks, solutions were reached rapidly.

- When requesters intervened using the workflow editor, they also reached correct solutions without derailment or starvation. Actually, what requesters did is only rejection of inconsistent work.

References

[1] Little, Greg, et al. "Turkit: human computation algorithms on mechanical turk." *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. 2010.

[2] Kittur, Aniket, et al. "Crowdforge: Crowdsourcing complex work." *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011.

[3] Kulkarni, Anand, Matthew Can, and Björn Hartmann. "Collaboratively crowdsourcing workflows with turkomatic." *Proceedings of the acm 2012 conference on computer supported cooperative work*. 2012.