クラウドワーカの能力とインクルージョン性を考慮したタスク割当て手法

橋本　大空† 　松原　正樹†† 　白石　優旗††† 　張　建偉†††† 　若月　大輔†††

森嶋　厚行††

† 筑波大学 図書館情報メディア研究科　〒 305–8550 茨城県つくば市春日 1-2
†† 筑波大学 図書館情報メディア系　〒 305–8550 茨城県つくば市春日 1-2
††† 筑波技術大学 産業技術学部　〒 305–8520 茨城県つくば市天久保 4-3-15
†††† 岩手大学 理工学部　〒 020–8551 岩手県盛岡市上田 4-3-5
E-mail: †hirotaka.hashimoto.2016b@mlab.info, ††{masaki,mori}@slis.tsukuba.ac.jp,
†††{yuhkis,waka}@a.tsukuba-tech.ac.jp, ††††zhang@iwate-u.ac.jp

あらまし　本論文はワーカの能力とインクルージョン性を考慮したタスクの割当て手法について提案する．ここでのインクルージョン性とはできるだけ多くのワーカに同じように作業分担してもらう性質のことである．ワーカの能力とインクルージョン性を考慮したタスクの割当ては，ボランティアベースの作業やソーシャルインクルージョンが重視されるシナリオにおいて有効である．インクルージョン性を高めるためには各ワーカに割り当てられるタスク数の分散を小さくすることが求められるが，その場合は従来のタスク割当てで重視されていた生産性やスループットが下がる．このようにインクルージョン性と生産性やスループットの間にはトレードオフの関係が生じる．そこで本論文では生産性やスループットを著しく下げることなくワーカのタスク量をできるだけ均等に割り当てる手法を提案する．提案手法を含め 4 つの手法によるタスク割当てのシミュレーションを行い，性能の比較評価を行った．

キーワード　クラウドソーシング, タスク割当て, ワーカ特性, ワーカ能力, ワークフロー

1. はじめに

クラウドソーシングにおけるタスク割当ては，重要なトピックであり，近年多くの研究が行なわれている．しかし，これらの研究における良いタスク割当てとは，結果のデータ品質や，タスク処理コストが安いもの，タスク処理時間が短くて済むもの，等の観点から定義されている．

本研究では，これらの研究とは異なった視点でのタスク割当て手法について提案する．具体的には，様々な能力を持つワーカが存在するときに，できるだけ多くのワーカに同じように作業分担してもらうためのタスク割当て手法である．我々は，この性質を「インクルージョン性」と定義し，インクルージョン性が高いタスク割当てを目標とする．

クラウドワーカの能力とインクルージョン性を考慮したタスク割当ては，いくつかのシナリオにおいて有効である．第一に，ボランティアクラウドソーシングシナリオがある．あるプロジェクトに貢献したいワーカがいるときには，その意思を無視せず，そのワーカにできる仕事をしてもらうことが重要である．また，特定のワーカへの負荷集中は避けなければならない．たとえば，講演中に講演を聞いている聴衆の人にボランティアでワーカとなってもらう時には，広く浅く負担することが理想である．

第二に，ソーシャルインクルージョンの考えが重要なシナリオである．ソーシャルインクルージョンとは社会的排除を無くそうという概念であり，これをクラウドソーシングの文脈で読むと，ワーカの能力によってタスクに参加できないという場面を無くし，ワーカそれぞれにできるタスクを行ってもらうという意味になる．今までのタスク割当てでは，限られた能力しか持たないワーカはタスクに参加できないということがあった．ソーシャルインクルージョンが重視されるシナリオでは，限られた能力しか持たないワーカにもそのワーカが行えるタスクを割り当て，今まで参加できていたワーカと共にタスクを行ってもらう仕組みが必要となる．

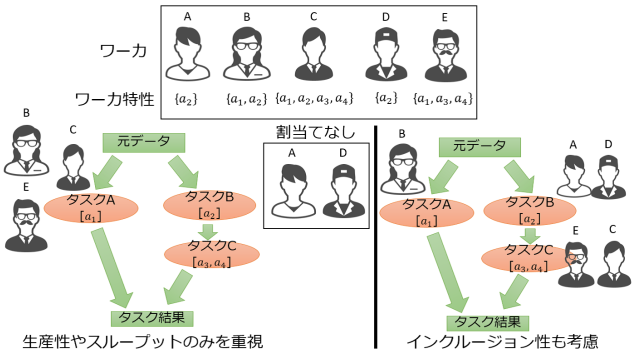

図 1　提案手法の全体図

本論文では，具体的には図 1 のモデルを対象として，タスク割り当てを行う．本モデルでは，タスクを複数組み合わせたワークフローを扱う．図において，楕円がタスクを表し，四角

はデータを表す．各ワーカの能力は，それぞれが持つ能力の集合 $\{a_1, a_2, \ldots\}$ として表現される．能力の例としては，文字を読める，手話がわかる，耳が聞こえる，等がある．また，ワークフロー中の各タスクには，その処理に必要な能力の集合が書かれている．このとき，そのワークフローを処理可能なように，ワーカにタスク割当てを行う．ここでは，タスク結果の品質については，単純に全てのワーカが一定以上の品質をもつとする．このとき，単に生産性や処理のスループットを最大化する割当てと，インクルージョン性を考慮した割当ては異なるものとなる．

もし，インクルージョン性を重視しない割当てを行うと，いくつかの問題が生じる場合がある．講演中に講演を聞いている聴衆の人にボランティアでワーカとなってもらう前述のケースで，タスク処理時間が短くて済むように割り当てた場合を想定する．ここで，速く結果が出力できるタスクが存在した場合，そのタスクができるワーカに負荷が集中してしまうことになる．また，生産性やスループットが最も良いタスクを行えないワーカは 図 1 のワーカ A やワーカ D のように，タスクに参加することができなくなってしまう．

一方，インクルージョン性だけを重視してしまうと，できるだけ多くのワーカに作業を分担するために，あるワーカであれば 1 人でできるタスクを複数人で行うことになる．その結果，インクルージョン性は高まるものの生産性やスループットが下がってしまう．

上記のような問題を解決するために，できるタスクが少ないワーカから割り当てるアプローチを用いたタスク割当て手法を提案する．このアプローチによって生産性やスループットを著しく下げずに，できるだけ均等にタスクを割り当てることでインクルージョン性を高める．

本論文では，このようなインクルージョン性を考慮したタスク割当てを行うための手法を提案し，シミュレーションによる評価を行う．

本論文の構成は次の通りである．まず第 2 節では本研究の関連研究について述べる．次に第 3 節では本論文で扱うモデルと問題について述べ，第 4 節ではタスク割当て手法について説明する．第 5 節でシミュレーションの方法と結果について示し，第 6 節でその結果を考察する．第 7 節はまとめと今後の課題である．

2. 関 連 研 究

これまでに，タスク割当てに関する研究は様々なものが行われている [1] [2] [3] [4] [5] [6]．例えば，タスク結果の質を向上させるタスク割当ての研究 [1] [2] [3] がある．これらの研究ではワーカの能力に応じてタスクを分割することや，タスクを行う順序の変更，評価尺度の導入などでタスク結果の品質向上を目指している．また，リクエスタ側の予算，タスク処理コストに注目してタスクを割り当てる手法を提案している研究 [4] [5] [6] もある．一方，この研究ではタスク結果の質やリクエスタ側の予算だけではなくインクルージョン性を考慮したタスク割当て手法を提案する．

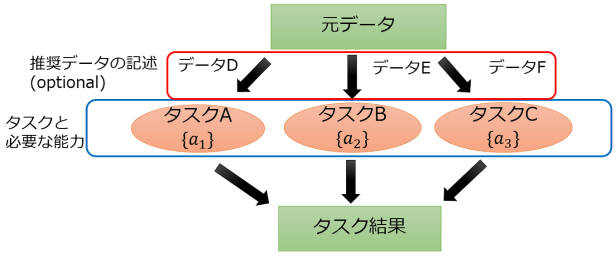

図 2　本研究で使用するワークフロー

また，クラウドソーシングのワークフローについて研究 [7] [8] [9] も存在しているが，いずれも最適化の提案や品質向上に役立つものであり，インクルージョン性を考慮した研究ではない．

他にも，タスク割当ての公平性について論じている研究 [10] もあるが，この公平性とは同じ能力を持つワーカが同じタスクにアクセスできるという意味の公平性であり，本研究が注目している多くの人々が同様に参加することを実現することとは異なる．

3. モデルと問題

本節では，本論文で扱うモデルと問題を定義する．

3.1 ワーカの設定

本論文では簡単化のため，初期状態でワーカの集合が決まっているものとする．図 1 で 5 人のワーカが既に存在するように，割当て前にワーカの集合が決まっている場合を考える．したがって，本論文では徐々にワーカが増えていったり，減っていくということは想定していない．これらのケースは今後の課題とする．

3.2 タスクの定義

本論文では，問題解決のためにタスクを細かく分割して複数のワーカで分担することを想定しているが，その分割されたタスクを**タスク**と呼び，そのタスクの 1 つ 1 つを**タスクインスタンス**と呼ぶ．具体的な例を挙げると，文字起こしをする際に分割された「聞こえた音声をタイピングで入力」や「入力された文の修正」がタスク,「1 文目の音声をタイピングで入力」や「1 文目の音声をテキスト化した文の修正」がタスクインスタンスである．

3.3 本手法で用いるワークフロー

本節では，本手法で使用するワークフローの例について説明する．このワークフローは図 2 のように，元データから最終的なタスク結果に至るまでに行うタスクに複数のパターンがあるワークフローで，それらのタスクが並列に並べられており，そのタスクごとにタスクを行うために必要な能力が示されているものである．

図 3 はワークフローを用いて記述された文字起こしシナリオの一例である．このように，文字起こしをするのにもタスクの分割方法によってはいろいろなパターンがある．そして，各タスクが要求する能力は異なっている．この特徴を活かして，ワーカの能力とインクルージョン性を考慮したタスク割当てを

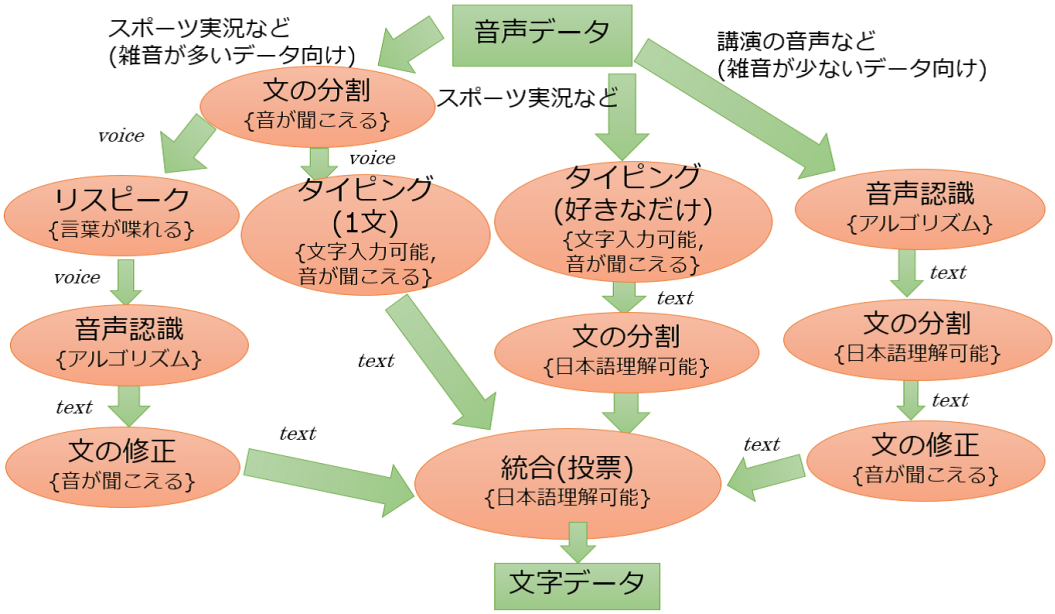

図 3　ワークフローを用いた文字起こしシナリオの例

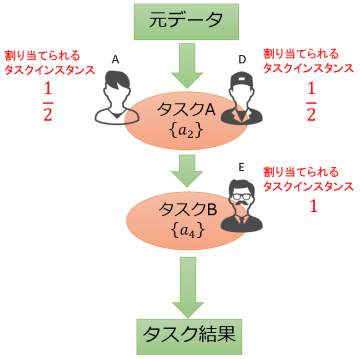

図 4　ワークシェアの例

行う．

3.4　ワークシェア

ここで，ワークフローを用いたタスク割当てで用いるワークシェアについて説明する．ここからの説明は図 4 を用いて行う．ワークシェアとは 1 人分のタスクを複数人で分担するという仕組みである．例えば，図 4 のように，パス中の前段のタスク (タスク A) には 2 人のワーカ (ワーカ A とワーカ D)，後段のタスク (タスク B) には 1 人のワーカ (ワーカ E) が割り当てられているシナリオを想定する．また，タスク A の処理時間とタスク B の処理時間は等しく，通常の 1 人分のタスクインスタンスの数を基準のために 1 とする．このとき，タスク A に割り当てられている 2 人が 1 人分のタスクをそれぞれ行ってしまうと，タスク B に割り当てられているワーカに 2 人分のタスクが与えられてしまい，処理しきれなくなってしまう．このとき，ワークシェアを行うとタスク A に割り当てられているワーカ A とワーカ D が 1 人分のタスクを分担するため，ワーカ A とワーカ D に割り当てられているタスクインスタンスは 1/2 となる．以降説明するタスク割当て手法では全ての手法でこのワークシェアを用いる．

3.5　記　　号

本論文で扱う問題の入力を次のようにモデル化する．

- タスクが要求する能力の集合 $Ability = \{a_1, a_2, \cdots, a_n\}$
- ワーカの集合 $W = \{w_1, w_2, \cdots, w_m\}$
- ワークフロー $Workflow(V, E)$

ワークフローの最初のノード v_1 は元データ，最後のノード v_{max} はタスク結果を示す．

- タスクの優先度 $Task\_Priority = [v_i, \cdots, v_l]$

タスクの優先度は列で表され，先頭にあるタスクの優先度が高いことを示している．

上記の入力が与えられた時，タスク割当ての問題とは次を出力することである．

- ワーカの割当て状況：$Assignment = \{(w_j, v_i)\}$

3.6　割当て手法の評価指標

本節では，割当て手法の評価指標について説明する．

3.6.1　インクルージョン性

1. 節で述べたが，インクルージョン性とは，様々な能力を持つワーカが存在するときに，できるだけ多くのワーカに同じように作業を分担してもらうということである．具体的には，ワーカに割当てられたタスクインスタンス数の分散が小さいことである．これを考えるために，次を定義する．$Variance$ はタスク割当て数の分散である．$Variance$ の値が小さければ小さいほど，ワーカに広く浅く作業を分担するような割当てができたということであり，インクルージョン性が高い割当てと言える．

$$Variance = \frac{\sum_{w \in W} (\overline{Tasks} - Tasks(w))^2}{|W|}$$

ただし，$Tasks(w)$ はワーカ w に割当てられたタスクインスタンス数，$\overline{Tasks}$ はその平均値とする．

3.6.2　生　産　性

ここでいう生産性とは，最終的なタスク結果が出力されるまでにかかるコストの効率を測るものである．生産性を評価する

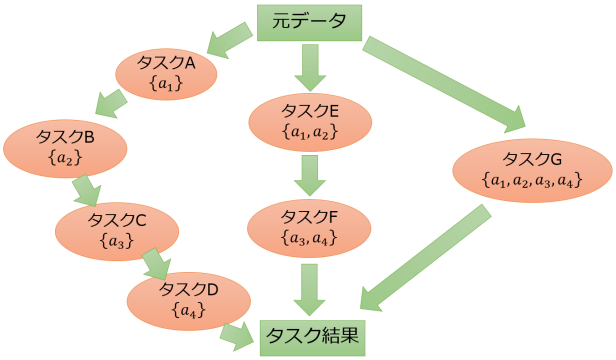

図 5　説明ワークフロー

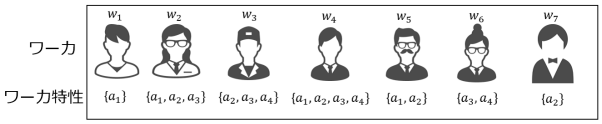

図 6　説明用ワーカ集合

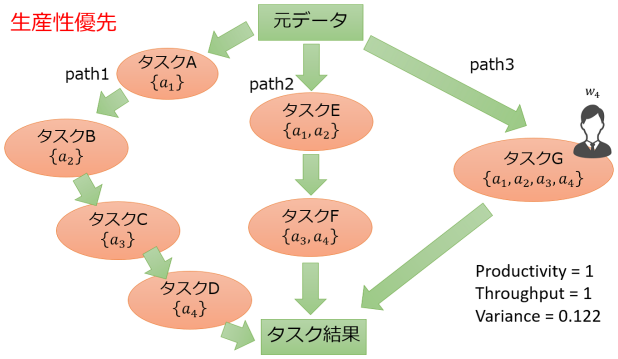

図 7　生産性優先割当て手法

ために次を定義する．$Productivity$ が生産性を表し，この値が大きいほど生産性が高い手法である．

$$Productivity = \frac{Final\_output}{Cost}$$

ただし，$Final\_Output$ は最終的なタスク結果数，$Cost$ はがコストを表すものとする．

本論文では簡単化のために，全てのタスクのコストをそれぞれ 1 とし，行われたタスク数とコストが同等とみなす．

3.6.3　スループット

ここでいうスループットとは，単位時間あたりの最終的なタスク結果数である．スループットを評価するために次を定義する．$Throughput$ がスループットを表し，この値が大きいほどスループットが高い手法である．

$$Throughput = \frac{Final\_output}{Unit\_time}$$

ただし，$Unit\_time$ が単位時間を表すものとする．

本論文では簡単化のために，単位時間を元データが最終的なタスク結果になるまでの時間とし，その時間は各パスで変わらないものとする．

4.　タスク割当て手法

本節では，シミュレーションで用いる 4 つのタスク割当て手法について説明する．ここで，説明のためのワークフローとワーカの集合をそれぞれ図 5 と図 6 に示す．

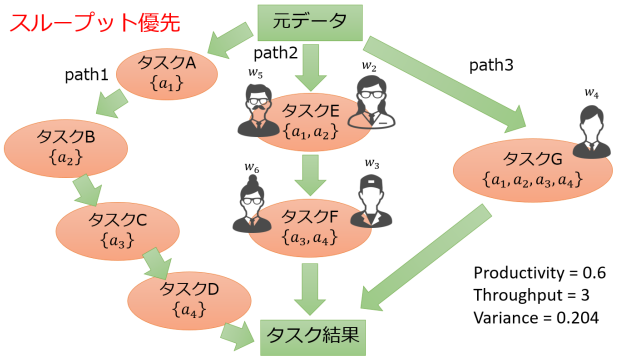

図 8　スループット優先割当て手法

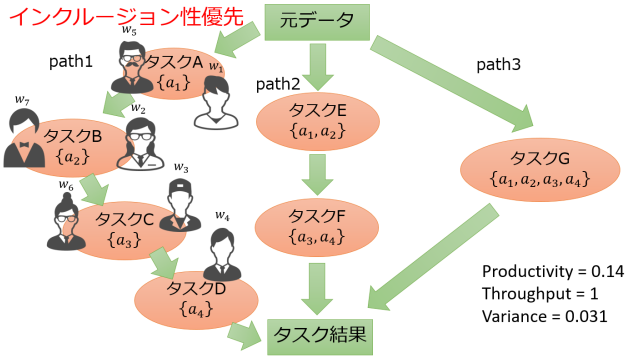

図 9　インクルージョン性優先割当て手法

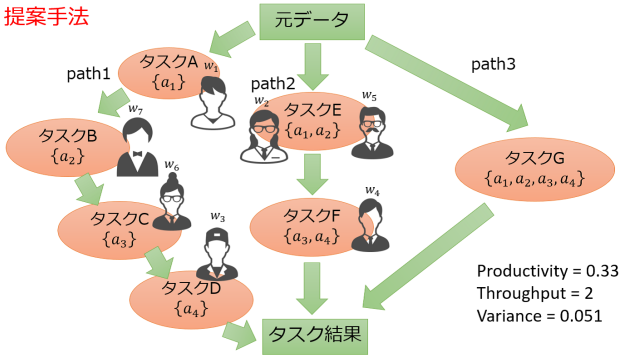

図 10　提 案 手 法

4.1　生産性優先割当て手法

生産性優先割当て手法とは図 7 のように，生産性を高めるためにコストが最も安く済むパスにしかワーカを割り当てない手法である．本論文の文脈では，タスク結果まで最短のパス中のタスクにしか割り当てないということである．そのため，最短パス中のタスクができないワーカはどのタスクにも割り当てられない．

4.2　スループット優先割当て手法

スループット優先割当て手法とは図 8 のように，スループットを高めるためにできるタスクが多いワーカを優先的に割り当てる手法である．スループットの最大化は NP 困難なため，ヒューリスティックを用いてできるタスクが多いワーカをできるだけ短いパスのタスクに割り当てることでスループット優先を実現する．

4.3 インクルージョン性優先割当て手法

インクルージョン性優先割当て手法とは図 9 のように，インクルージョン性を高めるためにワークフローの最長パスのタスクにしかワーカを割り当てない手法である．最長パスのタスクはタスク結果を出力するまでのタスクが多い，つまり大きなタスクが細かいタスクにより多く分割されているということになり，能力が少ないワーカにも参加できる可能性が高い．そのため，タスク割当て数の分散が下がり，インクルージョン性が高まる．

4.4 提 案 手 法

提案手法は図 10 のように，インクルージョン性を考慮しつつ生産性やスループットを著しく下げないようにタスクを割り当てる手法である．具体的には，できるタスクが少ないワーカを優先的にタスクに割り当て，そのタスク結果が無駄にならないようにできるタスクが多いワーカを割当てていく手法である．例えば，図 5 と図 6 では w_1 ができるタスクはタスク A のみ，w_7 ができるタスクはタスク B のみである．このとき，w_1，w_7 を先に割当て，そのタスクの結果が無駄になることなく最終的なタスク結果を出力できるようにタスクを w_2〜w_6 のワーカに割り当てていく．この時の割当て順もできるタスクが少ないワーカから割り当てる．もし，無駄になりそうなタスクがない場合はできるだけ短いパス中のタスクにワーカを割り当てる．これにより，生産性やスループットを著しく下げないことを実現する．次に，4.4.1 節で提案手法のタスク割当てアルゴリズムについて詳しく説明する．

4.4.1 提案手法のタスク割当てアルゴリズム

本節では，提案手法のタスク割当てアルゴリズムを示す．まず，アルゴリズム内で使用する関数を定義する．

- $Search\_Ability(w) \rightarrow \mathfrak{P}(Ability)$：ワーカ w が持っている能力を求める関数
- $Require\_Ability(v) \subseteqq Ability$：タスクが要求する能力を求める関数
- $Task\_PrioritySort(v, Workflow, Task\_Priority)$：あるノードに繋がっているノードを求め，タスクの優先度順にソートする関数

ここで，ワーカの能力とインクルージョン性を考慮したタスクの割当て手法のアルゴリズムを Algorithm 1 に示す．本アルゴリズムでは 7 行目〜10 行目であるワーカ w_j の持っている能力でできるタスクの数を数える．11 行目でその結果を $Task\_Numbers$ に格納し，これをワーカ全員分行う．14 行目で $Task\_Numbers$ で紐付いているワーカとできるタスク数を，できるタスクの少ない順に並び替える．15 行目で初期状態で割り当てるべきタスクをタスク優先度順に取り出し，$Task\_Available$ に格納する．このとき割り当てるべきタスクは元データ v_1 を入力とするタスクである．17 行目で $Task\_Numbers$ の先頭から 1 人のワーカを取り出し，19 行目〜26 行目で取り出したワーカに割当て可能なタスクを $Task\_Available$ の先頭から探す．見つかった時点で出力である $Assignment$ にワーカと割り当てるタスクの組を格納する．もし，取り出したワーカに割当て可能なタスクが無かった場合，$Task\_Available$ 以外のタスクに割り当てることを考えるため，$Task\_Priority$ 順に割当て可能なタスクを探す．その場合も見つかった時点で $Assignment$ にワーカと割り当てるタスクの組を格納する．格納が終わったら，割り当てたタスクの結果を入力とするタスクを $Task\_Available$ の先頭に追加する．もし，$Task\_Priority$ から割り当てるタスクを探した場合は割り当てたタスクより前のタスク，つまり本来であれば先に実行すべきタスクと割り当てたタスクの結果を入力とするタスクを $Task\_Available$ に追加する．これをワーカ全員にタスクが割り当てられるまで行う．

Algorithm 1 Assignment Algorithm

Input: $Ability, W, Workflow, Task\_Priority$
Output: $Assignment$

```
 1: Assignment = {}
 2: Task_Available = {}
 3: Task_Numbers = {}
 4: for j = 1 to |W| do
 5:    count = 0
 6:    T = V − v_1 − v_max
 7:    for i = 1 to |T| do
 8:       if Require_Ability(t_i) ⊆ Search_Ability(w_j) then
 9:          count = count + 1
10:       end if
11:    end for
12:    Task_Numbers << (w_j, count)
13: end for
14: Task_Numbers.sort_asc(count)
15: Task_Available =
      Task_PrioritySort(v_1, Workflow, Task_Priority)
16: for k = 0 to |Task_Numbers| do
17:    p = Task_Numbers(w).pop
18:    assign_flag = false
19:    for l = 0 to |Task_Available| do
20:       t = Task_Available.pop
21:       if Require_Ability(t) ⊆ Search_Ability(p) then
22:          Assignment << (p, t)
23:          assign_flag == true
24:          break
25:       end if
26:    end for
27:    if assign_flag == false then
28:       for l = 0 to |Task_Priority| do
29:          t = Task_Priority.pop
30:          if Require_Ability(t) ⊆ Search_Ability(p) then
31:             Assignment << (p, t)
32:             break
33:          end if
34:       end for
35:    end if
36:    Task_Available.unshift(Task_PrioritySort(t,
         Workflow, Task_Priority))
37: end for
```

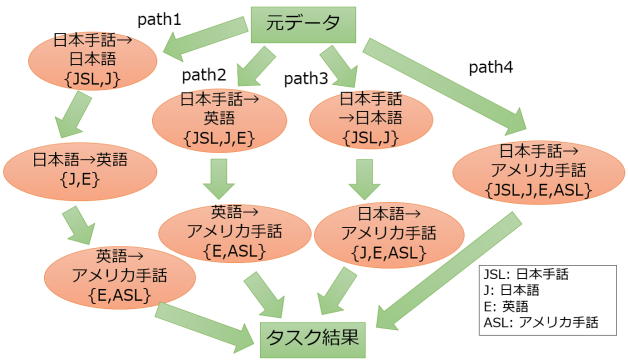

図 11　手話交流ワークフロー

5. シミュレーション

本節では，提案手法を評価するためのシミュレーションについて述べる．

5.1 シミュレーション概要

シミュレーションでは，提案手法が生産性やスループットを著しく下げずにインクルージョン性を高めているかどうかを評価する．比較するタスク割当て手法は提案手法を含む先述の 4 手法を用いる．また，タスクを行うためのワークフローは，2 種類用意する．そのワークフローの詳細を次に示す．2 つのワークフローはともに能力 4 つを用いるものである．

- 説明ワークフロー

4. 節で説明用として使ったワークフローである．(図 5)

- 手話交流ワークフロー

筑波技術大学で日本のろう学生とアメリカのろう学生が手話での交流をする際に実際に使用したワークフローである．(図 11) なお，手話交流ワークフローでは簡単化のために同様のタスク (path1 の 1 タスク目と path3 の 1 タスク目，path1 の 3 タスク目と path2 の 2 タスク目) が異なるパスに配置されているが，各指標を計算する際にスループットの最適化を行うため，同様のタスク内でワーカを移動する処理を行った．

本シミュレーションでは，ワーカを 100 人としたときに 100 人の能力の分布を変化させて，4 つのタスク割当て手法での割当てを行う．各タスクの割当て手法はワーカの割当て順序が非決定性を持っているため，複数回実験する必要がある．本シミュレーションでは，同じタスク割当て手法と同じワーカ分布で 5 回のシミュレーションを行った．評価手法には， 3.6 節で示した 3 つの評価指標を用いる．

変化させたワーカの分布を表 1 と表 2 に示す．表 1 は説明ワークフローでのシミュレーションに使用するワーカ分布，表 2 は手話交流ワークフローでのシミュレーションに使用する分布である．これらの表は所持能力の数でワーカの分布を示しているが，所持能力の数が等しいワーカの中で能力が異なることがあるため，詳しい能力の内訳も表中に記載した．また，分布 E〜分布 H で所持能力 1 つのワーカが 0 人なのは，図 11 の手話交流ワークフローでは所持能力 1 つのワーカを割り当てることができないためである．加えて，分布 E〜分布 H の所持能力 2 つのワーカの中で手話交流ワークフロー中のタスクに実行できるタスクがないワーカは存在しない．

5.2 シミュレーション結果

本節ではシミュレーションの結果を表とグラフを用いて示す．まず，各ワークフローと各分布での 5 回のシミュレーションの平均を表にまとめたものを表 3〜表 10 に示す．また，特筆すべき結果が出た分布 A，分布 F，分布 H のシミュレーション結果をグラフにしたものを図 12〜図 14 に示す．これらの図のグラフはそれぞれ，左のグラフが縦軸に Variance，横軸に Productivity，真ん中のグラフが縦軸に Variance，横軸に Throughput，右のグラフが縦軸に Throughput，横軸に Productivity をとったものである．なお，左下の原点に行けば行くほどよい手法と言えるように軸を配置した．

6. 考　　察

本節では，シミュレーション結果を踏まえた考察を述べる．

まず，表 3 〜表 10 より，インクルージョン性優先手法と提案手法を Throughput と Productivity で比較すると全てのパターンにおいて提案手法が優れている．さらに，生産性優先手法と提案手法を Variance で比較すると全てのパターンで提案手法が優れている．また，表 3，表 10 と図 12，図 14 を見ると，提案手法は Throughput と Productivity においてスループット優先手法に近い傾向を示しつつ，Variance の値がスループット優先手法を下回っている．したがって，提案手法が生産性やスループットを著しく下げることなく，インクルージョン性を高められる手法だといえる．しかし，必ずしもそうなるわけではなく，ワークフローや分布によっては全ての面でスループット優先手法より優れていないことがある．この要因については今後分析していく．

また，手話交流ワークフローの分布 F において，図 13 のように提案手法で 1 度だけ大きく Throughput と Productivity が下がることがあった．これは，割当て順に非決定性があり，特定のワークフローや分布によってはこのような事象が起こる場合がある．この点については今後手法を改善していく際に検討し，このような事象を避けるように手法を改善していく．

7. まとめと今後の課題

本研究では，ワーカの能力とインクルージョン性を考慮したタスク割当て手法を提案した．シミュレーションによって，ワークフローとワーカの分布によっては生産性やスループットを著しく下げることなくインクルージョン性を高めることができる割当てが存在し，提案手法がスループットや生産性を著しく下げずにインクルージョン性を高めることができる割当てを発見できる手法だということを示した．

今後の課題としては，ワーカの離脱や追加も考慮した割当てや，実際のコストやスループットも考慮した上での割当てなどが挙げられる．

謝　　辞

本研究の一部は JST CREST (#JPMJCR16E3) 及び JSPS 科研費 (JP16K16460, JP15K01056) の支援による。

表 1　ワーカの分布 (説明ワークフロー)

分布＼所持能力数	1 つ	2 つ	3 つ	4 つ
分布 A	25 人 $\{a_1\}$:6 人, $\{a_2\}$:8 人, $\{a_3\}$:7 人, $\{a_4\}$:4 人	25 人 $\{a_1,a_2\}$:6 人, $\{a_1,a_3\}$:5 人, $\{a_1,a_4\}$:4 人 $\{a_2,a_3\}$:2 人, $\{a_2,a_4\}$:4 人, $\{a_3,a_4\}$:4 人	25 人 $\{a_1,a_2,a_3\}$:10 人, $\{a_1,a_2,a_4\}$:6 人, $\{a_1,a_3,a_4\}$:2 人, $\{a_2,a_3,a_4\}$:7 人	25 人
分布 B	10 人 $\{a_1\}$:1 人, $\{a_2\}$:5 人, $\{a_3\}$:1 人, $\{a_4\}$:3 人	10 人 $\{a_1,a_2\}$:0 人, $\{a_1,a_3\}$:2 人, $\{a_1,a_4\}$:3 人 $\{a_2,a_3\}$:2 人, $\{a_2,a_4\}$:1 人, $\{a_3,a_4\}$:2 人	10 人 $\{a_1,a_2,a_3\}$:3 人, $\{a_1,a_2,a_4\}$:1 人, $\{a_1,a_3,a_4\}$:5 人, $\{a_2,a_3,a_4\}$:1 人	70 人
分布 C	70 人 $\{a_1\}$:15 人, $\{a_2\}$:17 人, $\{a_3\}$:22 人, $\{a_4\}$:16 人	10 人 $\{a_1,a_2\}$:2 人, $\{a_1,a_3\}$:3 人, $\{a_1,a_4\}$:1 人 $\{a_2,a_3\}$:1 人, $\{a_2,a_4\}$:1 人, $\{a_3,a_4\}$:2 人	10 人 $\{a_1,a_2,a_3\}$:4 人, $\{a_1,a_2,a_4\}$:2 人, $\{a_1,a_3,a_4\}$:1 人, $\{a_2,a_3,a_4\}$:3 人	10 人
分布 D	10 人 $\{a_1\}$:1 人, $\{a_2\}$:2 人, $\{a_3\}$:4 人, $\{a_4\}$:3 人	40 人 $\{a_1,a_2\}$:8 人, $\{a_1,a_3\}$:11 人, $\{a_1,a_4\}$:3 人 $\{a_2,a_3\}$:7 人, $\{a_2,a_4\}$:2 人, $\{a_3,a_4\}$:9 人	40 人 $\{a_1,a_2,a_3\}$:10 人, $\{a_1,a_2,a_4\}$:10 人, $\{a_1,a_3,a_4\}$:9 人, $\{a_2,a_3,a_4\}$:11 人	10 人

表 2　ワーカの分布 (手話交流ワークフロー)

分布＼所持能力数	1 つ	2 つ	3 つ	4 つ
分布 E	0 人	34 人 {JSL,J}:12 人, {J,E}:10 人, {E,ASL}:12 人	33 人 {JSL,J,E}:10 人, {JSL,J,ASL}:7 人, {JSL,E,ASL}:4 人, {J,E,ASL}:12 人	33 人
分布 F	0 人	10 人 {JSL,J}:5 人, {J,E}:1 人, {E,ASL}:4 人	10 人 {JSL,J,E}:4 人, {JSL,J,ASL}:3 人, {JSL,E,ASL}:0 人, {J,E,ASL}:3 人	80 人
分布 G	0 人	80 人 {JSL,J}:33 人, {J,E}:32 人, {E,ASL}:15 人	10 人 {JSL,J,E}:1 人, {JSL,J,ASL}:2 人, {JSL,E,ASL}:4 人, {J,E,ASL}:3 人	10 人
分布 H	0 人	10 人 {JSL,J}:2 人, {J,E}:6 人, {E,ASL}:2 人	80 人 {JSL,J,E}:24 人, {JSL,J,ASL}:21 人, {JSL,E,ASL}:20 人, {J,E,ASL}:15 人	10 人

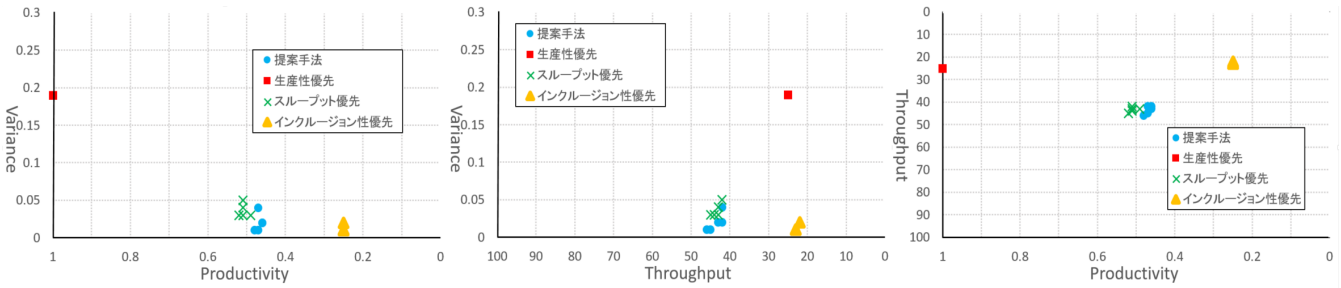

図 12　説明ワークフロー，分布 A でのシミュレーション

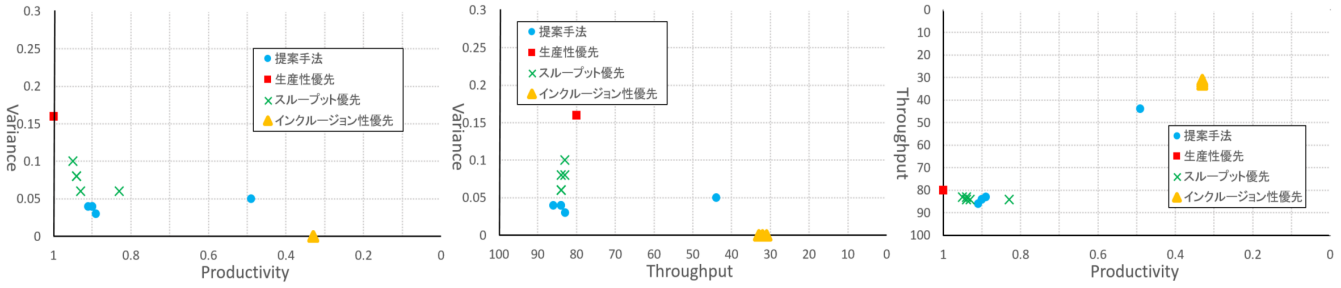

図 13　手話交流ワークフロー，分布 F でのシミュレーション

文　　献

[1] Dafna Shahaf and Eric Horvitz. Generalized task markets for human and machine computation. In *AAAI*, 2010.

[2] Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. Collaborative workflow for crowdsourcing translation. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pp. 1191–1194, New York, NY, USA, 2012. ACM.

[3] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. Qasca: A quality-aware task assignment system for crowdsourcing applications. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pp. 1031–1046, New York, NY, USA, 2015. ACM.

[4] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *Proceedings of the*

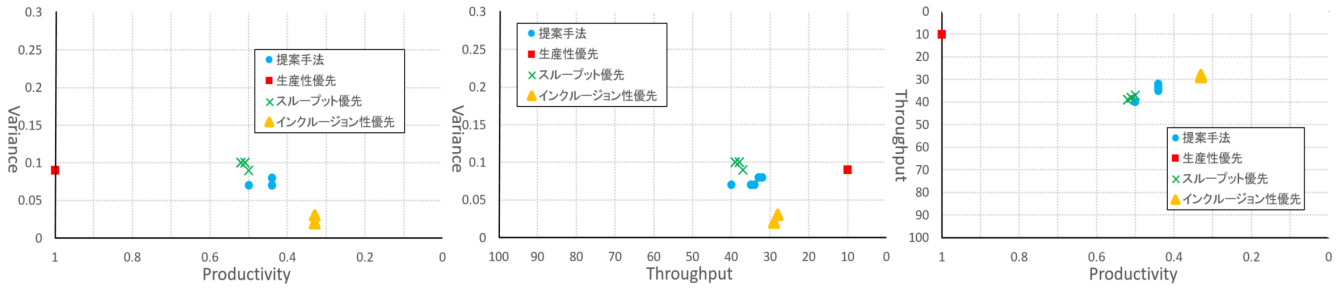

図 14　手話交流ワークフロー，分布 H でのシミュレーション

表 3　説明ワークフロー, 分布 A でのシミュレーション結果

手法＼指標	Productivity	Throughput	Variance
提案手法	0.47	43.6	0.020
生産性優先	1.0	25.0	0.190
スループット優先	0.51	43.4	0.036
インクルージョン性優先	0.25	22.8	0.012

表 4　説明ワークフロー, 分布 B でのシミュレーション結果

手法＼指標	Productivity	Throughput	Variance
提案手法	0.44	43.0	0.006
生産性優先	1.0	70.0	0.210
スループット優先	0.85	76.6	0.048
インクルージョン性優先	0.25	24.0	0.000

表 5　説明ワークフロー, 分布 C でのシミュレーション結果

手法＼指標	Productivity	Throughput	Variance
提案手法	0.28	27.0	0.002
生産性優先	1.0	10.0	0.090
スループット優先	0.36	32.4	0.020
インクルージョン性優先	0.25	22.6	0.008

表 6　説明ワークフロー, 分布 D でのシミュレーション結果

手法＼指標	Productivity	Throughput	Variance
提案手法	0.38	34.2	0.024
生産性優先	1.0	10.0	0.090
スループット優先	0.41	38.2	0.010
インクルージョン性優先	0.25	22.0	0.018

表 7　手話交流ワークフロー, 分布 E でのシミュレーション結果

手法＼指標	Productivity	Throughput	Variance
提案手法	0.54	42.4	0.060
生産性優先	1.0	33.0	0.220
スループット優先	0.63	52.2	0.044
インクルージョン性優先	0.33	31.0	0.002

表 8　手話交流ワークフロー, 分布 F でのシミュレーション結果

手法＼指標	Productivity	Throughput	Variance
提案手法	0.82	76.2	0.040
生産性優先	1.0	80.0	0.160
スループット優先	0.92	83.6	0.076
インクルージョン性優先	0.33	32.2	0.000

表 9　手話交流ワークフロー, 分布 G でのシミュレーション結果

手法＼指標	Productivity	Throughput	Variance
提案手法	0.36	27.0	0.030
生産性優先	1.0	10.0	0.090
スループット優先	0.43	32.0	0.030
インクルージョン性優先	0.33	26.2	0.018

表 10　手話交流ワークフロー, 分布 H でのシミュレーション結果

手法＼指標	Productivity	Throughput	Variance
提案手法	0.45	34.8	0.074
生産性優先	1.0	10.0	0.090
スループット優先	0.51	38.2	0.098
インクルージョン性優先	0.33	28.8	0.022

Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI'12, pp. 45–51. AAAI Press, 2012.

[5] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. I–534–I–542. JMLR.org, 2013.

[6] Jonathan Bragg, Andrey Kolobov, Mausam Mausam, and Daniel S Weld. Parallel task routing for crowdsourcing. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[7] Shinsuke Goto, Toru Ishida, and Donghui Lin. Understanding crowdsourcing workflow: Modeling and optimizing iterative and parallel processes. In *Fourth AAAI Conference on Human Computation and Crowdsourcing*, 2016.

[8] Bjorn Hartmann Anand Kulkarni, Matthew Can. Collaboratively crowdsourcing workflows with turkomaric. In *CSCW'12 Proceedings of the ACM 2012 conference on Computer Supported Cooperarive Work*, 2012.

[9] Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D. Ramchurn, and Nicholas R. Jennings. Crowdsourcing complex workflows under budget constraints. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 1298–1304. AAAI Press, 2015.

[10] Ria Mae Borromeo, Thomas Laurent, Motomichi Toyama, and Sihem Amer-Yahia. Fairness and transparency in crowdsourcing. In *EDBT*, pp. 466–469, 2017.