クラウドソーシングにおける参加率と活躍度を考慮したタスク割当て手法

著者	橋本 大空
内容記述	筑波大学修士(情報学)学位論文・平成31年3月25日授与(41281号)
発行年	2019
URL	http://hdl.handle.net/2241/00159797

クラウドソーシングにおける
参加率と活躍度を考慮したタスク割当て手法

筑波大学
図書館情報メディア研究科
2019年03月
橋本　大空

目 次

図 目 次

第1章　序論

クラウドソーシングにおいて，タスク割当ては重要なトピックの 1 つであり，複雑なクラウドソーシングワークフローにおけるタスク割当ては近年注目を集めている [1] [2]．複雑なクラウドソーシングワークフローは，データフローを含む一連の様々なタスクで構成されている．例えば，Find-Fix-Verify は複雑なクラウドソーシングワークフローとしてよく知られている [3]．一般的に，タスク割当てはタスク結果の質や生産性，スループットの面で評価される．これらは全てタスク視点の指標 (task-centric measures) で，リクエスタにとっての有効性を評価している．

本研究では参加率と活躍度という 2 つのワーカ視点の指標 (worker-centric measures) を考慮したタスク割当てを提案する．参加率は参加可能なワーカの中でどれほどのワーカが参加できているかを表し，活躍度は参加しているワーカの活躍度を示す．具体的には，参加率はタスクに参加したワーカの参加率であり，活躍度は各ワーカによって実行されたタスク数の標準偏差を使用して計算される．

参加率や活躍度は 2 つの重要なシナリオでのタスク割当てで重要となる．1 つ目のシナリオはボランティアクラウドソーシング（あるワーカがプロジェクトに貢献したいとき）であり，この場合はワーカの意図を無視せずそのワーカが実行可能なタスクに割当てをすることが重要となる．また，特定のワーカにタスクの作業負荷が集中することは避けなければならない．例えば，何人かの人が講義に出席し，その講義を聞きながらボランティアでリアルタイムな文字起こしの作業を依頼されている例を考える．このとき，人々の主な作業は講義を聞くことでありボランティアでの作業なため，より多くの人が文字起こしに参加し，各々がより少ない仕事で済むことが望ましい．

2 つ目のシナリオは，異なる能力や属性を持つ人々をできるだけ参加させ，特定のグループの人々を除外しないことが重要と考えられる場合である．これは，社会から特定のグループの人々を排除することを避けるソーシャル・インクルージョンの概念と関連している．このようなシナリオでは，各々の能力に応じてできる限り多くの人にタスクに参加してもらうことが重要である．

具体的には，この研究では図 1.1 に示すモデルでタスクを割り当てる．このモデルでは楕円がタスクを表し，正方形がデータを表すような複数のタスクを組み合わせたワークフローで構成されている．各ワーカの能力は各能力の名前と実数値の組の集合 $\{a_1 : \mathbb{R}, a_2 : \mathbb{R}, \cdots\}$ として表され，各能力 $a_1, a_2, \cdots$ の能力値を実数値で表す．能力の例としては，文字を読む能力，手話の理解力，聴覚の鋭さ，TOEFL のスコア，タイピング速度などがある．ワークフロー内の各タスクにおいて，実行するために必要な能力の条件は集合 $\{c_1 : a_1 > \mathbb{R}, c_2 : a_2 > \mathbb{R}, \cdots\}$ として表される．その後，ワークフローが処理可能なようにワーカがタスクに割当てられる．このとき，生産性やスループットを最大にする割当てと，参加率と活躍度を考慮したタスク割当ては異なる割当てとなる．

参加率や活躍度に注目していないタスク割当てでは，いくつかの問題が発生する場合がある．例えば，講義を聴いている聴衆がボランティアかつできるだけ早くタスク結果を出せるようにタスクを依頼された場合を想定する．ここで，結果をより速く出力できるタスクが存在すると，そのタスクが実行できるワーカに負荷が集中してしまう．また，そのタ

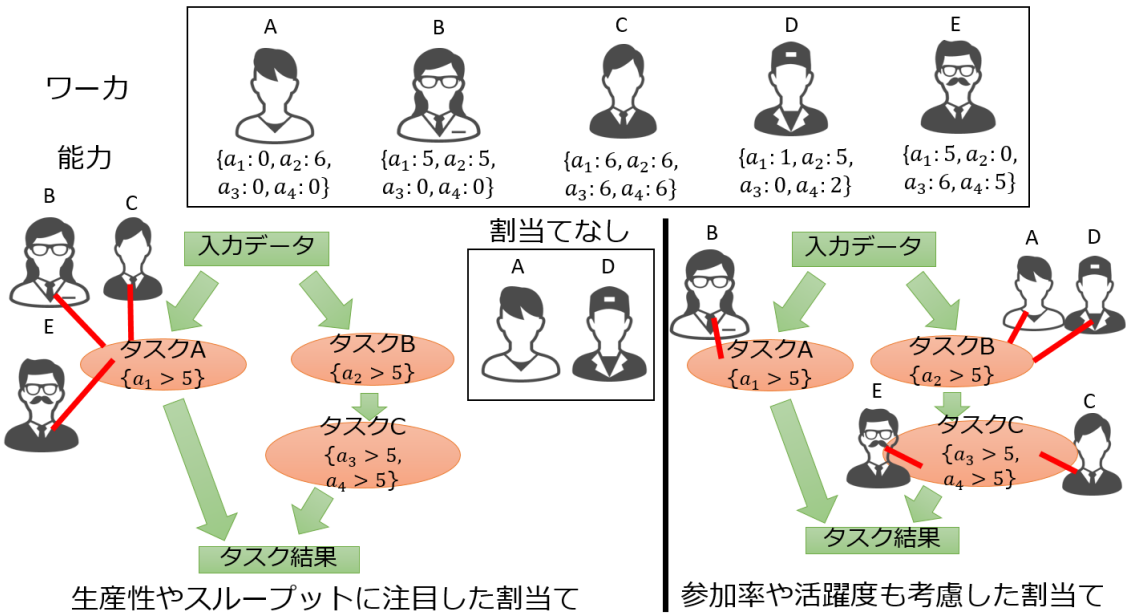

図 1.1: 全体図: (左) 生産性やスループットに注目したタスク割当て (右) 生産性やスループットだけでなく，参加率や活躍度も考慮したタスク割当て

スクが実行できないワーカは図 1.1 のワーカ A およびワーカ D のようにタスクに参加することができない．

対して，参加率と活躍度のみに注目しているタスク割当てでは，1 人のワーカでできるようなタスクもできるだけ多くのワーカで共有して実行することになる．その結果，参加率や活躍度は向上するものの，生産性やスループットが低下してしまう．

これらの問題を解決するために，5 つのタスク割当てを示し，その割当てをシミュレーションを用いて割当てを評価する．シミュレーションの結果，Participation Rate and Activity Degree Conscious Assignment が参加率と活躍度が高く，生産性やスループットを大幅に低下させることのない割当てであることを発見した．この研究の貢献は以下の通りである：

- 本論文は参加率と活躍度という新しいワーカ中心の指標を提案した．

  リクエスタが通常関心を持っているタスク中心の指標だけでなく，ワーカが関心を持っているワーカ中心の指標も考慮する．

- Participation Rate and Activity Degree Conscious Assignment が参加率と活躍度が高く，生産性やスループットを大幅に低下させることのない割当てであることを発見した．

  実際の世界で使用されている Cheap Task First Assignment と Versatile Worker First Assignment を含む 5 つの割当てを示す．先述の 2 つの割当てはそれぞれ生産性やスループットを考慮した割当てだが，想定するほどそれらの指標を向上させることができないことを発見した．対して，Participation Rate and Activity Degree Conscious Assignment は生産性とスループットを大幅に低下させずに，参加率と活躍度を向上させた．

本論文の構成は次の通りである．まず第 2 章では本研究の関連研究について述べる．次に第 3 章では本論文で扱うモデルや指標の定義について述べ，第 4 章ではタスク割当てについて説明する．第 5 章でシミュレーションの方法と結果について示し，第 6 章でその結果を考察する．第 7 章は結論および今後の課題である．

第2章　関連研究

2.1　ワークフローを用いたタスク割当てに関する研究

タスク割当ての問題は非常に重要なであり，様々な研究のトピックとなっている．特に，ワークフローを用いたタスク割当ては，複雑なタスクを処理する観点から多くの研究がされてきた [4] [5] [1] [3] [6] [7] [8]．例えば，ワークフローを用いたタスク割当ての研究ではコストの制約を維持しながら，タスク結果の質を確保することを目的とした研究 [5] や，作業者の割当て順序を制御することでタスク結果の質を向上させることを目的とした研究 [4] がある．これらの研究は，効率・コスト・品質などリクエスタにとって効果的な指標を考慮した割当てとなることを目的としている．しかし，本研究では参加率と活躍度という指標を用いて、ワーカにとって効果的なタスク割当てができるタスク割当て戦略を発見することを目的としている．

2.2　ワーカ視点での公平さを求めている研究

他にも，Fairness という概念を用いてワーカ視点で公平なタスク割当てを目的としている研究も存在する [9] [10] [11] [12] ．これらの研究ではタスクの割当てや報酬がワーカにとって公平であることを目指していたり [9] ，Spatial Crowdsourcing における作業の負荷や報酬を公平にすることを目指している [10] [11] が，本研究は参加率や活躍度といった面で平等に活躍できることを目指している．

第3章　定義

3.1　タスクとワーカ

$T = \{t_1, t_2, \ldots, t_{|T|}\}$ をタスクの集合とする． タスクとは同じ種類のタスクのテンプレートである． また，各タスクには，同じタスクテンプレートだが異なる識別子を持つタスクインスタンスがある． 同じタスクのタスクインスタンスは，関連付けられているデータが異なる可能性がある． 例えば，表示された画像に対してタグ付けするようなタスクがある場合，その画像は各タスクインスタンスで異なる画像であってもよいということになる．

次に，$W = \{w_1, w_2, \ldots, w_{|W|}\}$ をワーカの集合とする．また，$A = \{a_1, a_2, \ldots a_{|A|}\}$ を能力名の集合とする． 例えば,「英語のライティングスキル」は能力名である．能力名 a_i と労働者 w_j が与えられると、$a_i(w_j)$ は実数値で表した能力値を返す．

そして，関数 $W\text{-}Ability : W \to \mathcal{P}(A \times \mathbb{R})$ を定義し，ここで $W\text{-}ability(w)$ はベクトル $[a_1(w), \ldots, a_{|A|}(w)]$ を返す．

3.2　ワークフロー

図 3.1 にワークフローの例を示す． このワークフローにはデータフローを表す矢印で接続されたデータノード（四角形）とタスクノード（楕円）が含まれている． また，このワークフローは、入力データの集合を表す入力データノードから始まり、出力データの集合を表す出力データノードで終わるようになっている．

また，入力データノードから出力データノードへのパスが少なくとも 1 つ存在する． 複数のパスがある場合，データノードによって供給された各データは，パスのうちの 1 つに沿って移動していく． つまり，ノードから複数のパスにデータを供給するということはワークフローが「OR」構造であることを表している． データが供給されるパスは無作為に選択されるのではなく，タスクノードに関連付けられている設定値に従って選択される．

各タスクには，タスクを実行するワーカの能力値に関する条件が関連付けられている．例として，$TOEFL\_Score > 80$ を挙げる．これは，割り当てられたワーカの $TOEFL\_Score$ の能力値が 80 以上でなければならないことを示している．

具体例として，図 3.2 に文字起こしシナリオのワークフローを示す．このワークフローでは，様々なパスが存在している．各パスは，文字起こしされたテキストを作成するためのタスクの分割方法に対応している．各パス上のタスクは必要な能力が異なり，ワーカはある特定のパス上にあるタスクのみを実行するのではなく，異なるパス上で別のタスクを実行する場合がある．

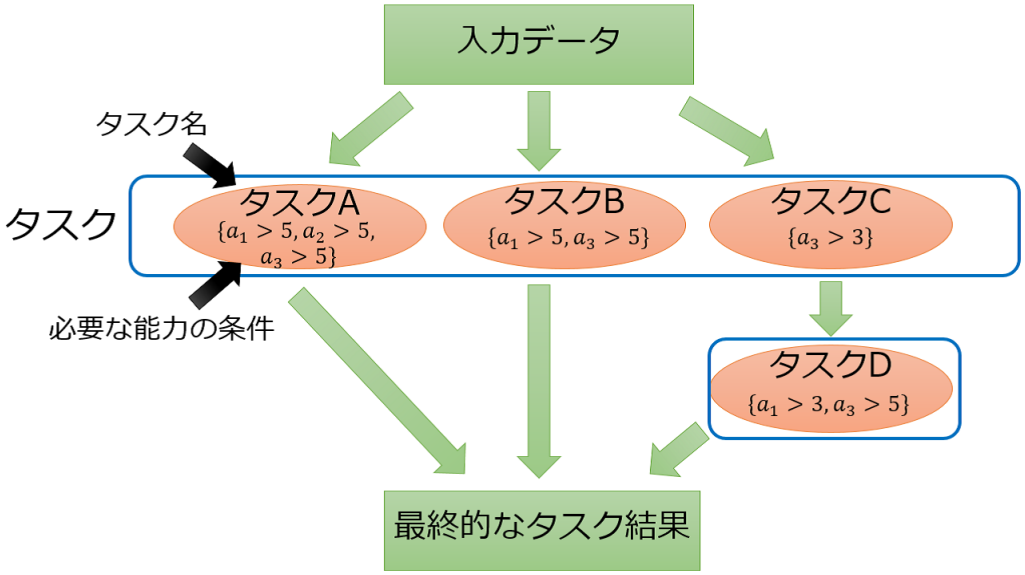

図 3.1: 本研究で使用するワークフロー

ここで，ワークフローを $G = (V, E, T\text{-}Ability, Cost, Time, \leq)$ として形式的に定義する．このときそれぞれは以下のように定義される:

- ノード集合 $V = T \oplus \{v_{start}, v_{end}\}$ はタスクの集合 T と入力データノードおよび出力データノードで構成されている．
- $T\text{-}Ability : T \to Cond$ はワーカがタスクの実行に必要とされている能力を持っているか判定するための条件（ブール関数）を返す．
- $Cost : T \to \mathbb{R}$ はあるタスクインスタンス t を実行するために必要なコスト $Cost(t)$ を返す．
- $Time : T \to \mathbb{R}$ はあるタスクインスタンス t を実行するために必要な時間 $Time(t)$ を返す．
- パス優先順序 $\leq : T \times T \to Boolean$ は兄弟ノード間で定義された半順序で，どのパスがワークフローのワーカ割当てに優先されるかを示す．

3.3 ワーカの割当てとタスクの実行

ワークフロー G とワーカ集合 W が与えられたとき，ワーカの割当ては $W \times T$ の部分集合として定義される．$W \times T$ は各 (w_j, t_i) について，$T\text{-}Ability(t_i)$ の条件を $W\text{-}Ability(w_j)$ が満たしているということである．例えば，$T\text{-}Ability(t_i)$ が $TOEFL\_Score > 80$ であり，$TOEFL\_Score(w_j) = 100$ であった場合，(w_j, t_i) は割当てに含めることができる．

ワークフロー G，ワーカ集合 W，およびそれらに対する割当て AS があるとき，パス上の全てのタスクにワーカが割り当てられている場合，入力データノードから出力データノードへの G のパス p は完全割当てパスであるという．

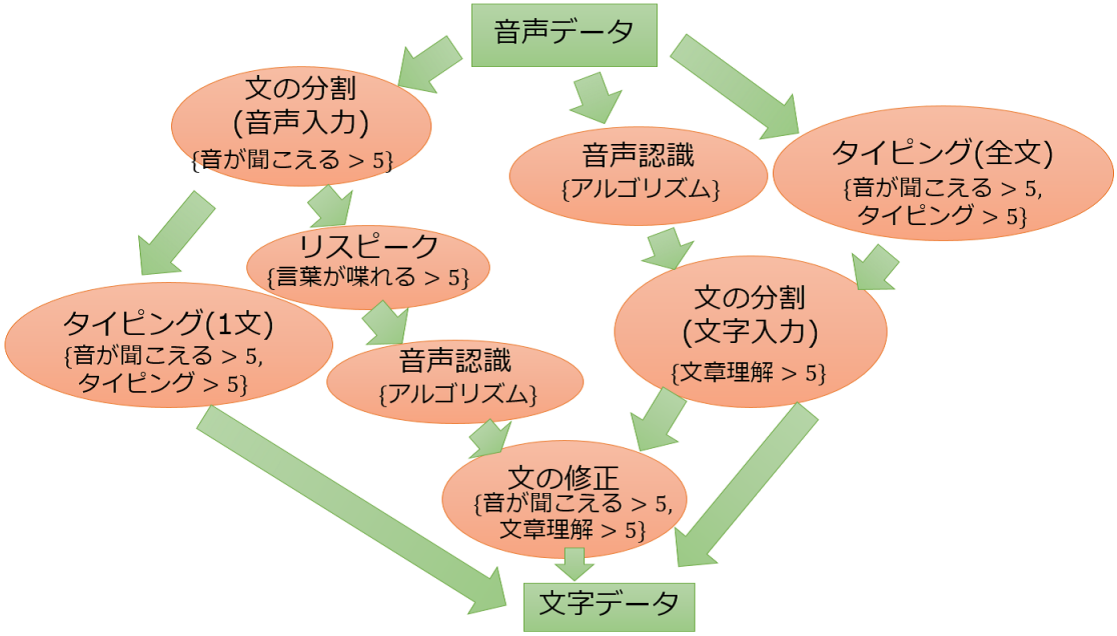

図 3.2: ワークフローを用いた文字起こしシナリオの例

完全割当てパスの集合として $completePaths(AS, G, W)$ を使用し，パス p 内のタスクの集合を示すために $Tasks(p)$ を使用する．

p を完全割当てパスとし，$capacity(p)$ を p 上の各タスクに割当てられているワーカ数の中で最も少ないワーカ数として定義する．これは，パス上の各タスクで処理できるデータの数である． ここでは，タスクに割当てられた全てのワーカーが各タスクのタスクインスタンスを均等に実行するワークシェアモデルを採用する．例えば，$capacity(p)$ が 1 であるような p 上のタスクに 2 人のワーカが割当てられている場合、各ワーカは図 3.3 のようにパスを通過しながら、0.5 タスクを実行するということである．

パスの選択では，入力データノードから来る各データに対してデータの数が $capacity(p)$ より少ないパスである p を選択する．[1]

3.4 タスク割当ての評価指標

本論文ではタスク割当てを評価するためにワークフロー全体での指標として 4 つの指標を定義する． 生産性を表す Productivity とスループットを表す Throughput はリクエスタが関心を持っている指標であるとともに，タスク視点の指標である．本研究では，ワーカ視点の指標として参加率を表す Participation Rate と活躍度を表す Activity Degree を提案する．Participation Rate はタスクにどれだけのワーカが割当てられたかの参加率を表し，Activity Degree はワーカの活躍度を表し，全てのワーカが同じように働くことができれば増加する値である．

[1] タスク実行中のパス選択は興味深い問題だが，その問題は将来の研究で扱うものとする．本研究は定常状態におけるワーカの割当てとタスクの実行を分析する．

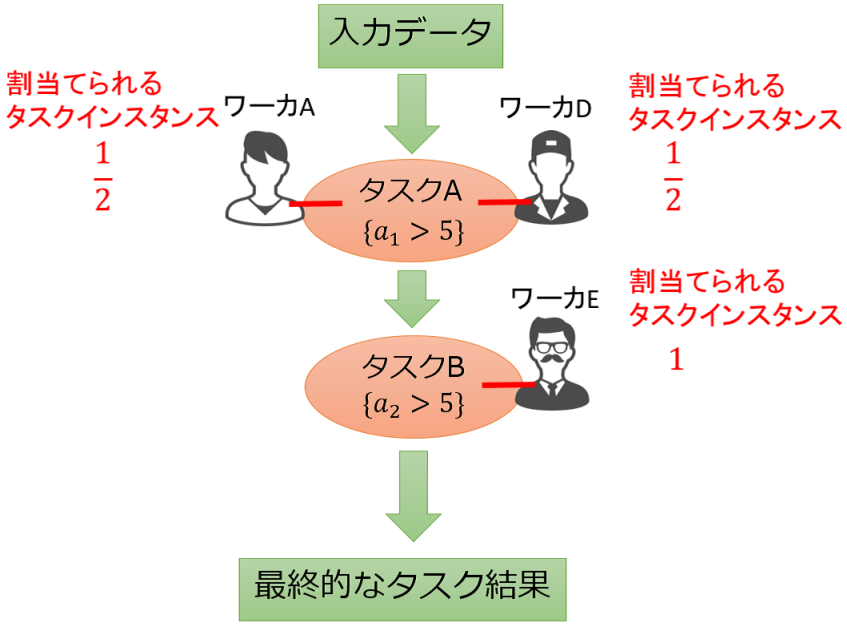

図 3.3: ワークシェアの例

3.4.1 Participation Rate

G と W と AS が与えられたとき，G と W における AS の Participation Rate である $PR(AS, G, W)$ は以下のように定義される:

$$PR(AS, G, W) = \frac{|\{w|(w, t) \in AS\}|}{|W|}.$$

3.4.2 Activity Degree

Activity Degree である $AD(AS, G, W)$ は以下のように定義される．

$$AD(AS, G, W) = \frac{1}{\sigma_{Num\_Task\_Instances(w)}},$$

ただし，σ は標準偏差を表し，$Num\_Task\_Instances(w)$ はワーカ w に割当てられたタスクインスタンスの数を表している．$Num\_Task\_Instances(w)$ は以下の式で表される:

$$Num\_Task\_Instances(w) = \frac{Min\_W_{AS}(p)}{W_{AS}(t)}$$

$$\{t, p|t \in Tasks(p) \wedge (w', t) \in AS \wedge w' = w)\}$$

ただし，$W_{AS}(t) = |\{w|(w, t) \in AS\}|$，$Min\_W_{AS}(p) = \min_{t \in Tasks(p)} W_{AS}(t)$ である．

3.4.3 Throughput

G におけるタスク割当て AS の Throughput である $TP(AS, G)$ は以下のように表される:

$$TP(AS,G) = \sum_{p \in completePaths(AS,G)} \frac{Min\_W_{AS}(p)}{max_{t \in Tasks(p)} Time(t)}.$$

直感的には，G における AS の Throughput は、AS の割当てにおける G 内の完全割当てパスの Throughput 値の合計である．完全割当てパスの Throughput はそのパス中の各タスクに割当てられているワーカ数の中で最も少ないワーカ数をそのパス中の最も実行に時間を必要とするタスクの実行時間で割ったものである．ここでは，ワークシェアモデルを採用しているため，パス上の各タスクに割り当てられるワーカの数が処理できるデータの数となることが保証される．

3.4.4 Productivity

G におけるタスク割当て AS の Throughput である $PROD(AS,G)$ は以下のように表される:

$$PROD(AS,G) = \sum_{p \in completePaths(AS,G)} \frac{PTR_{G,AS}(p)}{Price_G(p),}$$

ただし，$PTR_{AS,G}(p)$ は各パスにおける *proportion of throughputs*（Throughput の割合）であり，

$\sum_{p \in CompletePaths(G,AS)} PTR_{G,AS}(p) = 1$ となる．$Price_G(p)$ は $\sum_{t \in Tasks(p)} Cost(t)$ で表され，パス p を使用した時に最終的なタスク結果を出力するまでに必要なコストの合計である．したがって，$Price_G$ の逆数がパス p の生産性である．$PROD(AS,G)$ は、完全割当てパスの生産性の加重平均となる．

第4章　タスク割当て

本研究で扱う問題は与えられたワークフローとワーカにおいて 4 つの指標から見た良いタスク割当てを見つけることである．この章では、シミュレーションで比較する 5 つのタスク割当て戦略について説明する． それぞれのタスク割当てはバッチ割当てを行う（つまり，事前に全てのワーカを把握していると仮定する）．5 つのタスク割当て戦略はそれぞれ，いずれかの指標を向上させることが期待される戦略となっている．最初の 2 つの割当ては，それぞれ生産性とスループットを最大化するためにワーカをタスクに割当てる．また，活躍度を犠牲にすればどんなタスクでもとりあえずワーカを割り当てれば参加率を向上させることは簡単なため，残り 3 つのアルゴリズムのうちの 1 つは参加率を考慮しながら活躍度を増加させるようなアルゴリズムになっている．

4.1　Maximum-Productivity Assignment (MP)

MP は Productivity を最大化する戦略である．全ての割当て可能性を列挙し，Productivity が最大化される割当てを選ぶ．

4.2　Maximum-Throughput Assignment (MT)

MT は Throughput を最大化する戦略である．全ての割当て可能性を列挙し，Throughput が最大化される割当てを選ぶ．

しかし，図 3.1 のようなワークフローを用いたタスク割当てにおける Throughput の最大化の問題は計算量的に解くのが困難である．Throughput の最大化問題はナップサック問題に帰着することができる．Throughput の最大化問題を簡単化させた問題 T' を次のように定義する．

- 図 3.1 のような OR 構造を持ったワークフローにおいて，ワーカ集合に属している全てのワーカがワークフロー中のどのタスクも実行できるとする．

ここで，ナップサック問題は品物の集合 $I = \{1, 2, \cdots, N\}$，各品物 $i \in I$ の重みを w_i，価値を v_i，各品物の個数を x_i，ナップサックの容量を C としたとき，次のように表される:

$$max \sum_{i \in I} v_i x_i$$

$$s.t. \sum_{i \in I} w_i x_i \leq C$$

$$x_i \in \mathbb{N}$$

このとき，I がパスの集合，w_i をワークフロー中の各パスが最終的なタスク結果を出力するために必要なワーカの人数，v_i を各パスが最終的なタスク結果を 1 つ出力したときの Throughput の値，x_i を完全割当てパスの個数，C をワーカの人数と置き換えることがで

きる．このように置き換えると，Throughput の最大化問題を簡単化させた問題 T' はナップサック問題と等価であるといえるので，NP 完全な問題である．さらに，実際のタスク割当てではワーカがワークフロー中のあらゆるタスクを実行できる可能性は限りなく少なく，各ワーカが持つ能力によって割当てに制約がかかる．そのため，Throughput 最大化の問題はナップサック問題と同じくらい難しいか，それ以上に難しい．そのため，ワーカの分布によっては Throughput が最大化される割当てを発見できないことがある．

4.3 Cheap Task First Assignment (CTF)

CTF はタスクの総コストが低いパスを選択し，そのパスに対してタスクをランダムに実行できるワーカを選択し，選択したワーカをタスクに割当てて完全割当てパスを作るような戦略である．この戦略には Productivity の向上が期待されている．

4.4 Versatile Worker First Assignment (VWF)

VWF は現実の世界でよく用いられる手順の割当てである． 他のワーカよりも多くのタスクを実行できるワーカを最初に選び，$max_{t \in Tasks(p)} Time(t)$ が最も小さいパスのタスクを選択し，そのパス上のタスクに選んだワーカを割当てていく戦略である．この戦略には Throughput の向上が期待されている．

4.5 Participation Rate and Activity Degree Conscious Assignment (PAC)

PAC はできるだけ多くのワーカを参加させるとともに，他のワーカよりも実行できるタスク数が少ないワーカの活躍度を高めるために，そのワーカを先にタスクに割当てていく割当てである．基本戦略は，実行可能なタスクが少ないワーカーを優先的にタスクに割り当て、次にタスクの結果が無駄になるのを防ぐために（つまり，完全割当てパスを増やすために）多くのタスクが実行できるワーカをタスクに割当てるという流れである．無駄になる可能性がタスク結果がない場合，この戦略では，$max_{t \in Tasks(p)} Time(t)$ が最も小さいパスのタスクにワーカを割当てる． この戦略には参加率や活躍度の向上が期待されている．

Algorithm 1 PAC Algorithm

Input: $A, W, G, W\text{-}Ability$

Output: AS

1: $AS \leftarrow \{\}$
2: $Priority \leftarrow PriorityDepthFirst_G$
3: **for each** w in W' **do**
4: **for each** t in $Priority$ **do**
5: **if** $T\text{-}Ability(t)(w)$ **then**
6: AS *push* (t, w)
7: **break**
8: **end if**
9: **end for**
10: $N \leftarrow ImportantNodes_G(AS)$;
11: $Priority \leftarrow MoveNodes2Head(PriorityDepthFirst_G, N)$;
12: **end for**

Algorithm 1 に PAC のアルゴリズムを示す．Algorithm 1 で使用する関数と変数の定義を以下に示す:

- w が実行できるタスクの集合 $T_w \equiv \{t | t \in T$ ($T\text{-}Ability(t)$ は $W - Ability(w)$ の値を保持している)$\}$
- ワーカを実行できるタスク数が少ない順に並べた列 $W' = [w_{i_1}, \ldots, w_{i_{|W|}}]$, ただし，全ての $j < k$ において $|T_{w_{i_j}}| \leq |T_{w_{i_k}}|$ である．
- タスクの優先順 $Priority = [t_1, \cdots, t_{|T|}]$
  $Priority$ は割当て状況に応じて動的に変化する．
- $PriorityDepthFirst_G$: 与えられたグラフ G の設定値によって与えられた兄弟ノードの優先順位にしたがって，深さ優先探索の順序でノードの列を返す．
- $ImportantNodes_G(AS)$: G の各タスクノードについて，先祖および子孫の兄弟の合計の最大ワーカ数を計算し、タスクノードに割当てられているワーカ数が先祖もしくは子孫の兄弟の合計より小さい場合はそのタスクノードを結果に含めるようなノードの集合を返す．
- $MoveNodes2Head(PriorityDepthFirst_G, N)$:
  $PriorityDepthFirst_G$ によって返された列に基づいて，ノード集合 N に含まれるノードが先頭に移動した列を返す．N に含まれるノード間の順序は設定値によって与えられる兄弟の優先順位によって決まる．

このアルゴリズムでは，$PriorityDepthFirst_G$ の結果を $Priority$ の初期値として使用する．割当ての部分（5 行目から 8 行目）では，最も実行できるタスク数が少ないワーカの能力で実行できるタスクを $Priority$ の順で探し，割当て可能なタスクが見つかればワーカとタスクの組を AS に格納する．

その後，$Priority$ の更新をするために $ImportantNodes_G(AS)$ で無駄になりそうなタスク結果が存在しているパス中のタスクを抽出する．最後に，抽出したタスクを $Priority$ の先頭に移動するように $Priority$ を更新する．以上の一連の流れを全てのワーカに対して行う．

第5章　シミュレーション

この節では，5 つのタスク割当て戦略を評価するために行ったシミュレーションについて説明する．

5.1　シミュレーション概要

このシミュレーションでは，5 つのタスク割当て戦略に生産性やスループットを著しく下げずに，参加率や活躍度が高い割当てを生成できるものがあるかを評価する．比較するタスク割当て戦略は先述の 5 つを用いる．また，ワークフローは 3.2 節で説明した文字起こしシナリオのワークフローである図 3.2 を用いる．文字起こしの題材はニュースの音声で，時間は 78 秒間の音声である．

次に，ワーカの能力分布について説明する．本シミュレーションではワーカの能力分布を 3 つ用意した．1 つ目は,「リアル分布」と呼び，実際のクラウドソーシングプラットフォームである Yahoo! クラウドソーシングで，130 人のワーカから各々の能力を 0(全くできない)〜6(とてもよくできる) の 7 段階で自己申告するタスクを用いて収集した分布である．このシミュレーションでは，7 段階のうち 5 もしくは 6 と答えたワーカがその能力を所持していると定義した．2 つ目と 3 つ目はシミュレーションのために作成した能力分布で，2 つ目を「能力数フラット分布」，3 つ目を「能力タイプフラット分布」と呼ぶ．これらの分布は Yahoo!クラウドソーシングで収集した分布と条件を揃えるため，ワーカの人数を 130 人としている．能力数フラット分布では，所持能力数 1,2,3,4 におけるワーカの分布をフラットに割り振ったような分布である．また，所持能力数の中で存在する所持している能力のタイプ（例 : { タイピング，文章理解 }）それぞれにフラットにワーカを割り振った．能力タイプフラット分布では，このワークフロー中のタスクを 1 つでも実行できる所持能力のタイプ全 14 タイプにフラットに割り振ったものである．すべての場合で，フラットに割り切れない場合はランダムでワーカを割り振った．収集したワーカの 3 つの能力分布を，実行できるタスクの数によって分類したものを表 5.1 及び 5.2，5.3 に示す．

このシミュレーションでは，以上のワークフローとワーカの能力分布を用いて 5 つのタスク割当て戦略での割当てを行う．各割当て戦略は非決定性を持っているため，複数回同じ割当てを行う必要がある．そのため，今回は 5 回同じ割当てを行った．評価値には 3.4 節で示した 4 つの指標を用いる．この 4 つの指標は完全に割当てが完了した段階で計算される．

今回のシミュレーションでは結果をわかりやすくするために，正規化を行った．リアル分布での結果においては，Productivity と Throughput はこのワークフローとこのワーカの能力分布で最大の Productivity, Throughput を 1 とするように正規化を行った．能力数フラット分布と能力タイプフラット分布での結果において，Productivity に関してはリアル分布と同様だが，Throughput の最大を発見することは 4.2 節で述べたように困難なため，能力数フラット分布と能力タイプフラット分布では発見できなかった．そのため，このワークフローと理想のワーカ（全てのワーカがあらゆるタスクを実行できるワーカ）とした状況下の最大を 1 とするように正規化を行った．また，Activity Degree は最大が無限になるので，以下の式で正規化を行った．

表 5.1: リアル分布

能力数	ワーカ数
1	6 {文章理解}:6
2	4 {言葉が喋れる, 文章理解}:2, {言葉が喋れる, タイピング}:1, {音が聞こえる, 言葉が喋れる}:1
3	18 {言葉が喋れる, タイピング, 文章理解}:1, {音が聞こえる, 言葉が喋れる, 文章理解}:9, {音が聞こえる, 言葉が喋れる, タイピング}:8
4	102 {音が聞こえる, 言葉が喋れる, タイピング, 文章理解}:102

表 5.2: 能力数フラット分布

能力数	ワーカ数
1	32 {音が聞こえる}:10,{言葉が喋れる}:11,{文章理解}:11
2	33 {音が聞こえる, 言葉が喋れる}:5,{音が聞こえる, タイピング}:6, {音が聞こえる, 文章理解}:6,{言葉が喋れる, タイピング}:5, {言葉が喋れる, 文章理解}:6,{タイピング, 文章理解}:5
3	32 {音が聞こえる, 言葉が喋れる, タイピング}:8, {音が聞こえる, 言葉が喋れる, 文章理解}:8, {音が聞こえる, タイピング, 文章理解}:8, {言葉が喋れる, タイピング, 文章理解}:8
4	33 {音が聞こえる, 言葉が喋れる, タイピング, 文章理解}:33

$$AD(AS, G, W) = 1 - \frac{\sigma_{Num\_Task\_Instances(w)}}{0.5}$$

表 5.3: 能力タイプフラット分布

能力数	ワーカ数
1	28 {音が聞こえる}:9,{言葉が喋れる}:10,{文章理解}:9
2	54 {音が聞こえる, 言葉が喋れる}:9,{音が聞こえる, タイピング}:9, {音が聞こえる, 文章理解}:9, {言葉が喋れる, タイピング}:9, {言葉が喋れる, 文章理解}:9, {タイピング, 文章理解}:9
3	38 {音が聞こえる, 言葉が喋れる, タイピング}:10, {音が聞こえる, 言葉が喋れる, 文章理解}:10, {音が聞こえる, タイピング, 文章理解}:9, {言葉が喋れる, タイピング, 文章理解}:9
4	10 {音が聞こえる, 言葉が喋れる, タイピング, 文章理解}:10

表 5.4: 各タスクに必要な時間とコスト

タスク名	必要時間	コスト
文の分割（音声入力）	81.43	8.14
リスピーク	65.21	6.52
文の修正	16.98	1.70
タイピング（1 文）	54.38	5.44
タイピング（全文）	452.18	45.22
文の分割（文字入力）	64.73	6.47

タスクに必要な時間は実測値を用いることとした．4 人のワーカに全てのタスクを実行するよう依頼し，各タスクの必要時間を記録した．その時間の平均を各タスクに必要な時間として使用することとした．また，音声認識にワーカは必要なく，必要時間も 0 秒と定義した．また，タスクに必要なコストを 1 秒あたり 0.1 と定義した．各タスクに必要な時間とコストを整理した表を表 5.4 に示す．

5.2　シミュレーション結果

この節ではシミュレーションの結果を示す．表 5.5 はリアル分布を用いた時の各割当て戦略の評価指標を示している．同様に表 5.6 に能力数フラット分布，図 5.7 に能力タイプフラット分布を用いた時の評価指標を示す．図 5.1〜図 5.6 は、リアル分布を用いて 5 回行った試行の各試行における全ての評価指標を 2 次元のグラフにまとめたものである．同様に図 5.7〜図 5.12 に能力数フラット分布，図 5.13〜図 5.18 に能力タイプフラット分布を用いた時の全ての評価指標を 2 次元のグラフにまとめたものを示す．

表 5.5 や表 5.6 は PAC が CTF や VWF に比べて Activity Degree が高くなることを示している．Throughput に関しても PAC は CTF や VWF と比べて優れている．また，Productivity の面ではリアル分布では CTF や VWF と比較して著しく劣っていない．能力数フラット分布では VWF に多少劣っている．表 5.7 では，PAC が Throughput の面では CTF や VWF より優れており，Productivity の面では著しく劣っていない．しかし，Activity Degree で VWF に劣ってしまっている．

また，図 5.1〜図 5.6 と図 5.7〜図 5.12，図 5.13〜図 5.18 でも多くのパターンで表から読み取れることと同様の傾向が見て取れる．また，リアル分布を使った割当ての図 5.1〜図 5.6 は，PAC のある 1 つのパターンで MT と同様の割当てが生成できることを示している．

表 5.5: リアル分布での評価指標

タスク割当て戦略 \ 評価指標	Productivity	Throughput	Activity Degree	Participation Rate
Participation Rate and Activity Degree Conscious	0.972	0.999	0.895	1
Cheap Task First	0.970	0.965	0.808	1
Versatile Worker First	0.974	0.900	0.611	1
Maximum-Productivity	1	0.937	0.448	0.915
Maximum-Throughput	0.975	1	0.826	1

表 5.6: 能力数フラット分布での評価指標

タスク割当て戦略 \ 評価指標	Productivity	Throughput	Activity Degree	Participation Rate
Participation Rate and Activity Degree Conscious	0.852	0.781	0.860	1
Cheap Task First	0.868	0.725	0.652	1
Versatile Worker First	0.944	0.466	0.303	0.82
Maximum-Productivity	1	0.637	0.072	0.654
Maximum-Throughput	—	—	—	—

表 5.7: 能力タイプフラット分布での評価指標

タスク割当て戦略 \ 評価指標	Productivity	Throughput	Activity Degree	Participation Rate
Participation Rate and Activity Degree Conscious	0.938	0.670	0.211	1
Cheap Task First	0.954	0.613	0.149	0.969
Versatile Worker First	0.889	0.376	0.338	0.78
Maximum-Productivity	1	0.549	0.068	0.569
Maximum-Throughput	—	—	—	—

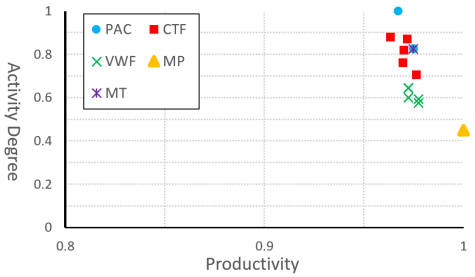

図 5.1: リアル分布: (縦軸)Activity Degree (横軸)Productivity

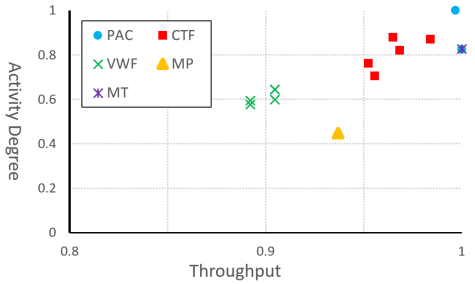

図 5.2: リアル分布: (縦軸)Activity Degree (横軸)Throughput

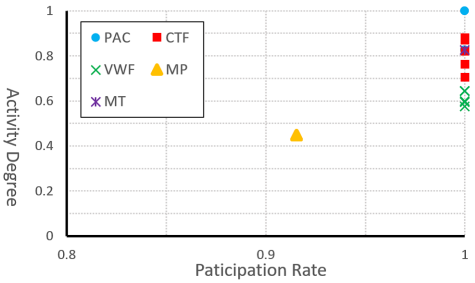

図 5.3: リアル分布: (縦軸)Activity Degree (横軸)Paticipation Rate

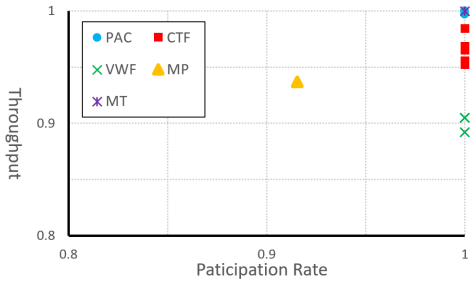

図 5.4: リアル分布: (縦軸)Throughput (横軸)Participation Rate

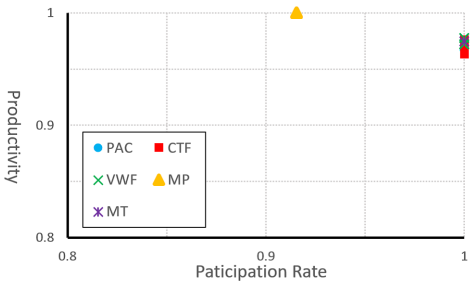

図 5.5: リアル分布: (縦軸)Productivity (横軸)Paricipation Rate

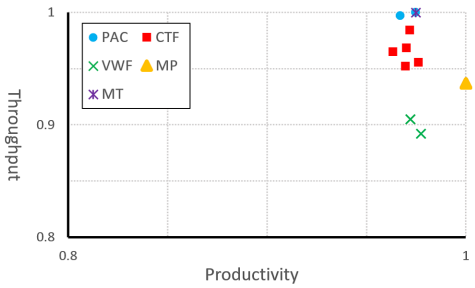

図 5.6: リアル分布: (縦軸)Throughput (横軸)Productivity

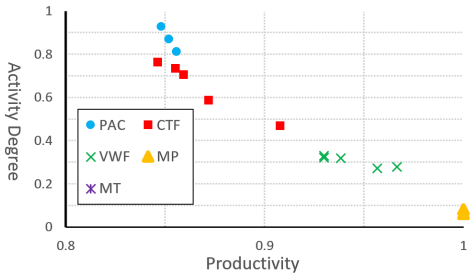

図 5.7: 能力数フラット分布: (縦軸)Activity Degree (横軸)Productivity

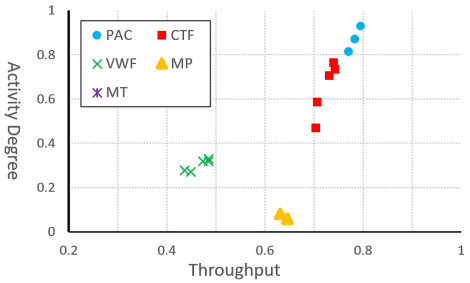

図 5.8: 能力数フラット分布: (縦軸)Activity Degree (横軸)Throughput

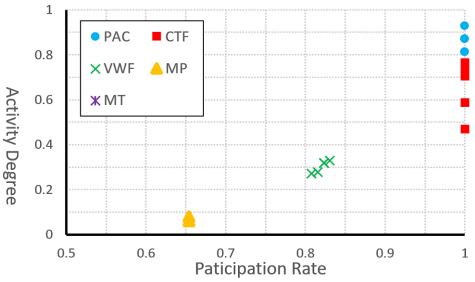

図 5.9: 能力数フラット分布: (縦軸)Activity Degree (横軸)Paticipation Rate

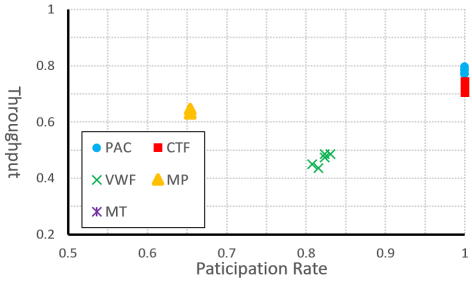

図 5.10: 能力数フラット分布: (縦軸)Throughput (横軸)Participation Rate

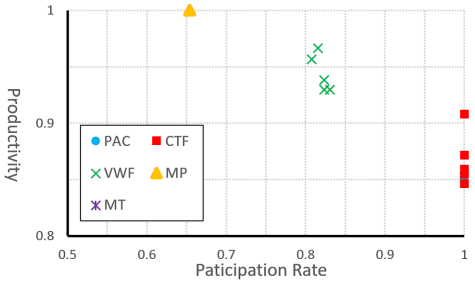

図 5.11: 能力数フラット分布: (縦軸)Productivity (横軸)Paricipation Rate

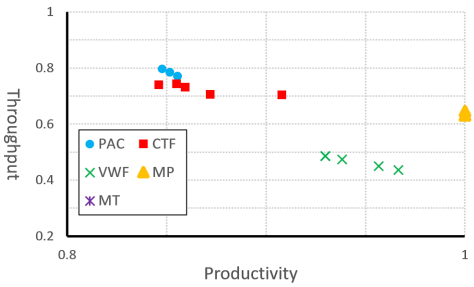

図 5.12: 能力数フラット分布: (縦軸)Throughput (横軸)Productivity

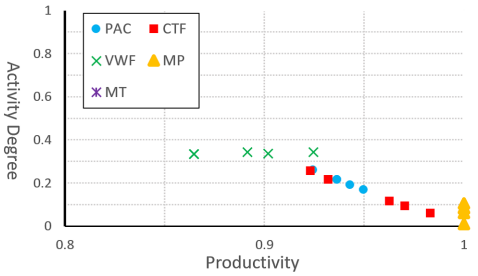

図 5.13: 能力タイプフラット分布: (縦軸)Activity Degree (横軸)Productivity

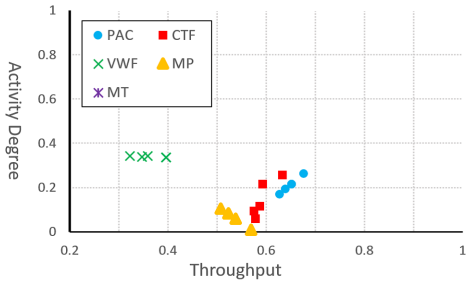

図 5.14: 能力タイプフラット分布: (縦軸)Activity Degree (横軸)Throughput

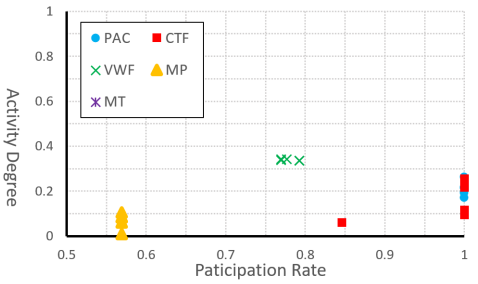

図 5.15: 能力タイプフラット分布: (縦軸)Activity Degree (横軸)Paticipation Rate

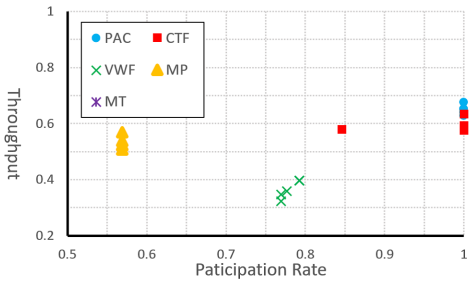

図 5.16: 能力タイプフラット分布: (縦軸)Throughput (横軸)Participation Rate

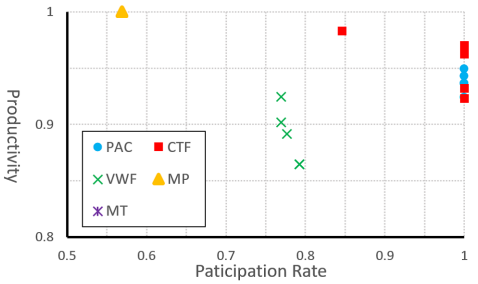

図 5.17: 能力タイプフラット分布: (縦軸)Productivity (横軸)Paricipation Rate

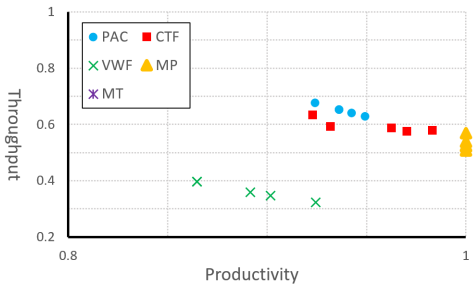

図 5.18: 能力タイプフラット分布: (縦軸)Throughput (横軸)Productivity

第6章　考察

リアル分布と能力数フラット分布において，PAC が高い活躍度の割当てを生成できるのは，実行できるタスク数が少ないワーカを最初に割当て，その後に実行できるタスク数が多いワーカを最初に割当てたワーカを有効に使うようにタスクに割り当てることで実現できていると考えられる．また，Throughput や Productivity の面から言えば，最初に割当てたワーカのタスク結果を無駄にしないように割当てているとも言えるため，他の割当てより優れているか遜色ない結果になっていると考えられる．ただし，能力数フラット分布の Productivity においては，VWF のほうが優れてしまっている．これは，VWF が実行できるタスク数が多いワーカを先に割当てていることと，タスクの必要時間によってコストを決めたため所要時間を短くする割当てが結果的に生産性が高くなる割当てとなることから，最初に多くのタスクができるワーカを使って生産性が高い割当てをした後，少ないタスクしかできないワーカを切り捨てることで生産性が高い割当てになっていると考えられる．これは表 5.6 を見ると Participation Rate が VWF で PAC や CTF と比べて低いことからも裏付けられる．

また，能力数フラット分布では Throughput で PAC が優れた結果，Productivity では遜色ない結果となっているが，Activity Degree で VWF より劣った結果となった．これは CTF や VWF がワーカ全員を参加させずに Participation Rate が 1 とならないことよりも，PAC が参加率を上げるためにワーカ全員を参加させようとしたことで 1 つのタスクに多くのワーカが割当てられたことにより割当て指標としての Activity Degree が低下してしまったものと考えられる．このように，分布によっては活躍度を向上させようという戦略の PAC でも Activity Degree が思ったように高くならない場合があることが明らかとなった．

また，CTF と VWF がそれぞれ生産性やスループットを思ったように向上させないことも明らかとなった．CTF については，CTF が最も生産性の高いパスのタスクだけでなく，可能な限り生産性が高いパスのタスクにワーカを割当てようとしており，ワーカの能力分布によって生産性が高くないパスを選択することもあるためであると考えられる．これは 5.6 などで CTF を MP と比較すると，最も生産性が高いパスのタスクにしか割当てない MP と比べて CTF では Productivity が下がっている代わりに Participation rate が上がっていることからも言えると考えられる．VWF については，この割当て戦略が参加率を考慮したものであるがゆえに，後半に生産性もしくはスループットがよくなるような割当てをできないワーカが残ってしまうことでタスク結果を有効に使えず，最終的なタスク結果数が PAC と比べて少なくなってしまうことで起こっていると考えられる．このように，ある指標を向上させようとした割当てがワーカの能力による割当て制約によって結果的にうまくいかないことがあることも示された．

第7章　結論

本論文では，今まで考慮されてこなかったワークフロー全体の割当てとしてのワーカ視点の指標である参加率や活躍度を提案した．次に，タスク視点の指標である生産性やスループットとワーカ視点の指標である参加率や活躍度について定義し，それらいずれかの指標を向上させると期待される 5 つのタスク割当て戦略を示した．その 5 つのタスク割当て戦略を比較し評価するシミュレーションでは，実際のクラウドソーシングプラットフォームを用いて集めたワーカ分布とシミュレーションのために作成したワーカ分布 2 つを用いて，文字起こしシナリオによるタスク割当てを行った．その結果，Participation Rate and Activity Degree Conscious Assignment (PAC) が参加率と活躍度が高く，生産性やスループットを著しく下げない割当てを実現できる戦略であることを示した．しかし，ワーカの分布によっては PAC が Activity Degree を向上させることができない場合もあることも示され，これに関しては今後分析や検討が必要であると考えられる．

今後の課題としては，さらに多くのワーカ分布によるシミュレーションによってワーカ分布による傾向のさらなる分析や，割当てを操作できるようなパラメータを考案・モデル化して，今回挙げた割当て戦略のみならず，リクエスタの意向やタスク実行の環境に合わせた割当てを生成できるようにすることが挙げられる．

謝辞

本論文を提出するにあたり，研究指導教員である森嶋厚行教授と副指導教員である鈴木伸崇准教授に深く感謝致します．特に森嶋教授にはご多忙の中でも研究の話を熱心にしてくださり，行き詰まった際には一緒に長時間相談に乗ってくだるなど多くのことをご指導いただきました．

また，共著者でもあった筑波大学の松原正樹助教，筑波技術大学の白石優旗准教授，若月大輔准教授，岩手大学の張建偉准教授にも感謝致します．松原助教には約 2 年にわたり研究室でささいな質問にも嫌な顔一つせずに答えてくださり，研究の障壁を取り除いてくださいました．筑波大学外の先生方には週 1 回の打ち合わせを学部時代から行っていただき約 3 年もの間研究の相談に乗っていただきました．他にも私が専門外の領域に関しても多くの知見を与えてくださいました．

最後に，融合知能デザイン研究室の先輩・同期・後輩には，研究で行き詰まった時に相談に乗っていただいたり，論文の添削を夜遅くまでしていただきました．この研究室の先輩・同期・後輩の皆様に恵まれて有意義な研究生活を送ることができました．

その他にも研究についてアドバイスをいただいた多くの皆様にこの場を借りて深く感謝致します．ありがとうございました．

参考文献

[1] Bjorn Hartmann Anand Kulkarni, Matthew Can. Collaboratively crowdsourcing workflows with turkomaric. In *Proceedings of CSCW'12*, 2012.

[2] Ken Mizusawa, Keishi Tajima, Masaki Matsubara, Toshiyuki Amagasa, and Atsuyuki Morishima. Efficient pipeline processing of crowdsourcing workflows. In *Proceedings of CIKM '18*, 2018 (upcoming).

[3] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: A word processor with a crowd inside. In *Proceedings of UIST '10*, pp. 313–322, New York, NY, USA, 2010. ACM.

[4] Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. Collaborative workflow for crowdsourcing translation. In *Proceedings of CSCW '12*, pp. 1191–1194, New York, NY, USA, 2012. ACM.

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

[6] Aniket Kittur, Susheel Khamkar, Paul André, and Robert Kraut. Crowdweaver: Visually managing complex crowd work. In *Proceedings of CSCW '12*, pp. 1033–1036, New York, NY, USA, 2012. ACM.

[7] Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D. Ramchurn, and Nicholas R. Jennings. Budgetfix: Budget limited crowdsourcing for interdependent task allocation with quality guarantees. In *Proceedings of AAMAS '14*, pp. 477–484, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[8] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. Crowdforge: Crowdsourcing complex work. In *Proceedings of UIST '11*, pp. 43–52, New York, NY, USA, 2011. ACM.

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

[10] Qing Liu, Talel Abdessalem, Huayu Wu, Zihong Yuan, and Stéphane Bressan. Cost minimization and social fairness for spatial crowdsourcing tasks. In Shamkant B. Navathe, Weili Wu, Shashi Shekhar, Xiaoyong Du, X. Sean Wang, and Hui Xiong, editors, *Database Systems for Advanced Applications*, pp. 3–17, Cham, 2016. Springer International Publishing.

[11] Fuat Basık, Bugra Gedik, Hakan Ferhatosmanoglu, and Kun-Lung Wu. Fair task allocation in crowdsourced delivery. *IEEE Transactions on Services Computing*, 2018.

[12] Paul Hyman. Software aims to ensure fairness in crowdsourcing projects. *Commun. ACM*, Vol. 56, No. 8, pp. 19–21, August 2013.