Vol.2016-ICS-182 No.6
2016/3/2

タスク優先度に基づくマイクロタスクの逐次割り当て手法の提案と評価

The Proposal and Evaluation of Sequential Assignment Method for Micro Task Based on Task Priority

江川知樹[1]　小野良太[1]　山下倫央[2]　川村秀憲[1]

Tomoki EGAWA[1], Ryota ONO[1], Tomohisa YAMASHITA[2] and Hidenori KAWAMURA[1]

[1] 北海道大学大学院情報科学研究科

[1]Graduate School of Information Science and Techology Hokkaido University

[2] 産業技術総合研究所　人工知能研究センター

[2] Artificial Intelligence Research Center,National Institute of Advanced Industrial Science and Technology

Abstract: Micro tasks have been a large amount processed in short time and easily.It can be preformed reduce workload and increase of the working efficiency for micro tasks.In this paper,We propoesd five assignment methods for minimizing the procesing time based on remaining tasks and the average time to process tasks.We evaluate methods using simulation that data was the actual task data of the processed task.The best method in the simulation was performed was introduced into the actual work, and evaluated.As a result,We reduced 9.3% of workload.

1 はじめに

近年，発注者から処理依頼のあったマイクロタスクをオンラインでワーカと呼ばれる受注者に委託するサービスが普及している．本稿では，多数のマイクロタスクの処理時間を短縮するため，ワーカの能力に応じたマイクロタスクの割り当て手法を提案し，その効果を検証する．

マイクロタスクとは，作業を細分化して短時間で処理できる難易度の低い仕事で，専門的なスキルが不要なデータの入力作業，画像の分類，アンケートの回答などが挙げられる．以降，本稿ではマイクロタスクをタスクと呼ぶ．通常，タスクは複雑な作業を細分化したものであるため，一度の発注で複数の種類のタスクが大量に生じる．そこで，ワーカごとにタスクの処理能力が異なる場合，ワーカのタスク処理時間を考慮したタスク割り当てを行うことで，全タスクの処理完了時間を短縮することが期待できる．

オンラインタスクの従来研究においては，ワーカのタスクの正答率を向上させるための手法が数多く提案されている．オンラインタスクのタスク割り当て問題としては、依頼者側からタスクの精度を上げるための研究が多くなされている．（本稿において，精度とは正答率のことを指す）従来研究では，ワーカに対して精度を向上させる手法として，タスクの作業結果を用いて，低精度のワーカに対してタスクを割り当てない手法が行われている[1][2]．Ho[3]らはタスク処理の期間を分割し，前半部分をワーカのスキルの獲得に当て，後半部分で獲得したスキルを基に，全タスクの精度が最大になるようにタスクの割り当てを行った．

タスクに対して精度を向上させる手法として，while[4]らは同じタスクを複数のワーカに割り当て，その結果からタスクの精度を向上させる手法を行っている．また松原ら[5]はタスク集合を複数の分割し，ワーカの能力に応じて割り当てることにより精度を向上させる手法を行った．

本稿では，ワーカの能力に応じてタスクの割り当てを行うことで，全タスク処理時間を短縮させる手法を提案し，シミュレーションを用いて提案手法の効果を検証した．それらの手法の中で平均タスク処理時間を最小にする手法を，実際に運用されてるイベント情報配信サービスにおけるイベント情報の編集作業の割り当てに導入し，実環境における効果測定を行った．

本稿ではタスクの割り当てにおいて，オンラインタスクではワーカの出現条件はあらかじめ与えられていない場合や一部のタスク処理を開始した後に新たにタスクが追加される場合がある状況を対象とする．そのような状況では，依頼者からタスクを与えられた時点で静的に割り当てを終了してしまうと処理されないタスクが発生する．本稿では，タスクを逐次的に割り当てていくことで処理されないタスクをなくし，かつ全タスク処理時間を最小化する割り当て手法を提案する．

本稿は以下の構成になっている．第２章ではタスク割り当て問題のモデルについて述べる．第３章では提案するタスク割り当て手法について述べる．第４章では提案手法を評価するためのシミュレーションのデータセット及びシミュレーションの結果について述べる．第５章では提案手法の実際に運用されているサービスへの導入と，その結果について述べて．第６章では本稿のまとめについて述べる．

2 タスク割り当て問題

本章では，タスク割り当ての対象となる継続的に発生するタスクが追加される状況を概説し，その定式化を行う．

2.1 タスク割り当て問題の対象

本稿では，タスク割り当て問題の対象として，発注者から継続的にタスクが追加される状況を対象とする．タスクの処理プロセスとして　発注者が複数種類のタスクの処理をシステムに依頼する．また，全タスクの処理が終了する前に，発注者からの新たなタスクが継続的に追加される．ワーカは複数人が並行して処理するタスクを要求し，タスクマネージャは未処理のタスクを割り当てる．ワーカは割り当てられたタスクの処理を終了すると，新しいタスクを要求する．次節ではタスクマネージャがワーカにタスクを割り当てるプロセスを対象として定式化を行う．

2.2　タスク割り当て問題の定式化

本章では Ho らによるタスクの精度を上げるためのタスク割り当て問題を参考にし，タスク処理時間の最小化問題のためのタスク割り当て問題として定式化する．

タスクマネージャからワーカに割り当てられるタスク集合を$S = \bigcup_{i=1}^{N} s_i$とする．ここで i はタスクの種類を表し，タスクの種類 i のタスク数はc_iで表す．タスクを処理するワーカを$j \in \{1,2,\dots,M\}$と定義する．

ワーカ全体でタスクを処理する順番を$t \in \{1,2,\dots,T\}$とし，t 番目にタスクを処理するワーカを$a(t) \in \{1,2,\dots,M\}$とする．t 番目のタスクが処理される際にタスクs_iに残っているタスク数を$c_{i,t}$とする．t 番目の割り当ては$c_{i,t} \geq 1$のタスクから割り当てる．

また，継続的にタスクが増加するため，t 番目のタスク処理前に加えられて増加するタスク数を $g_{i,t}$ とする．ワーカ j がタスクs_iを処理するためにかかったタスク処理時間を$w_{i,j}$とする．

ワーカ j にタスクs_i が割り当てられている状況変数を$x_{i,j}$として設定し，ワーカ j がタスクs_iが割り当てられている場合には 1，それ以外の場合には 0 とする．

上記のモデルから，全てのタスクを処理するという制約の元，全タスク処理時間の和を最小化させることを目的とし，最小化問題として定義する．

目的関数

$$\sum_{t=1}^{T}\sum_{i=1}^{N} w_{i,a(t)}\, x_{i,a(t)} \tag{1}$$

制約条件

$$\sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{x_{i,a(t)}} = 1 \tag{2}$$

$$c_{i,t} \geq 0 \ \ \forall(i,t) \tag{3}$$

$$c_{i,0} + \sum_{t=1}^{T} g_{i,a(t)} \geq \sum_{t=1}^{T} x_{i,a(t)} \quad \forall(i) \tag{4}$$

制約条件の式 2 は 1 回の割り当てにおいて各ワーカには１つのタスクしか割り当らないことを表している．式３は残りタスク数が存在するタスクから割り当てが行われることを表している．式 4 は割り当てられるタスク数は，初期タスク数と増加タスク数の合計値しか割り当てられないことを表している．

以上の式から全タスク処理時間の最小化問題として考えることができる．全タスクが処理される期間において，あらかじめワーカの出現する順番やタスクの追加される順番が既知であるなら，目的関数を線形計画法でタスク割り当て最小化することができる．しかし，ワーカの出現順やタスクの追加順は未知であるので，線形計画法を適用することができない．よって次章でタスクの割り当てを逐次的に行う手法を提案する．

3 タスク割り当て手法

タスクマネージャからワーカへのタスク割り当て$x_{i,a(t)}$を決定するために，本稿では各ワーカのタスク

の種類ごとに設定されたタスク優先度$p_{i,j}$を導入し，ワーカ j に対してはタスク優先度$p_{i,j}$が大きいタスクが割り当てられるようにする．
タスクマネージャはワーカがタスクの要求時に逐次的にタスクの割り当てを行う．あるワーカの複数のタスク優先度$p_{i,j}$が同じ場合には，以下で述べる期待タスク処理時間$w'_{i,j}$が小さいタスクを優先して割り当てる．

タスクマネージャがワーカのタスク優先度を算出するには，各ワーカのそれぞれのタスクの種類に対する期待タスク処理時間$w'_{i,j}$を用いる．各ワーカが同じ種類のタスクを行う場合，作業内容が類似しているため，作業時間もこれまでの作業時間と同じになる傾向がある．そのため，期待タスク処理時間として，過去の同じ種類のタスクの平均処理時間を用いる．以下に$p_{i,j}$の計算手法を述べていく．

3.1 処理時時間優先

タスク優先度$p_{i,j}$に期待タスク処理時間の逆数$\frac{1}{w'_{i,j}}$を代入し，期待タスク処理時間の短いタスクが優先される．

$$p_{i,j} = \frac{1}{w'_{i,j}}$$

この手法はタスクマネージャがワーカに対して，期待タスク処理時間が短いと予測されるタスクを割り当てることによって，全タスク処理時間を短くする割り当て手法である．

3.2 処理順位優先

各タスクに対して，全てのワーカのタスク処理時間の長さで順位付けを行う関数$ranking(i,j)$を設定する．この関数はワーカ j のタスクs_iにおける期待タスク処理時間の順位を出力する．タスク優先度$p_{i,j}$に順位関数$ranking(i,j)$を代入し，各タスクの順位が高いタスクが優先される．

$$p_{i,j} = ranking(i,j)$$

この手法は，タスクマネージャが各ワーカに対して，タスクの処理時間の順位が高い順番に割り当てることによって，期待タスク処理時間の短いワーカに割り当て，全タスク処理時間を短くする割り当て手法である．

3.3 処理時間偏差優先

各タスクに対して，タスク処理時間の平均$\overline{w_i}$を求める．タスク優先度$p_{i,j}$にタスク処理時間の平均とワーカの期待タスク処理時間の偏差 $\overline{w_i} - w'_{i,j}$を代入し，平均との偏差が大きいタスクが優先される．

$$p_{i,j} = \overline{w_i} - w'_{i,j}$$

この手法はタスクマネージャが各ワーカに対して，タスク処理時間の平均と期待タスク処理時間の偏差の大きいタスクを割り当てることによって，ワーカに平均よりも短い時間で処理可能なタスクを割り当てることにより，全タスク処理時間を短くする割り当て手法である．

3.4 正規化処理時間偏差優先

前節で説明したタスク処理時間偏差優先を正規化したもので，タスク優先度$p_{i,j}$に正規化した平均との偏差$\frac{\overline{w_i} - w'_{i,j}}{\overline{w_i}}$ を代入し，正規化した偏差の大きいタスクが優先される．

$$p_{i,j} = \frac{\overline{w_i} - w'_{i,j}}{\overline{w_i}}$$

この手法はタスクマネージャが各ワーカに対して，処理時間の正規化した平均との偏差が大きいタスクを割り当てることによって，ワーカが他のワーカに比べて効率的に処理できるタスクを割り当てることによって，全タスク処理時間を短くする割り当て手法である．

3.5 残りタスク数優先

タスク優先度$p_{i,j}$に平均タスク処理時間と期待タスク処理時間の偏差が正なら残りタスク数$c_{i,t}$に偏差をかけた値を代入，偏差が負なら偏差を代入することで，タスク処理時間の短いワーカは，残りタスク数の多いタスクを，タスク処理時間が長いワーカは，早く処理可能なタスクが優先される．

$$p_{i,j} = \begin{cases} (\overline{w_i} - w'_{i,j})C_{i,t} & (if\ \overline{w_i} - w'_{i,j} > 0) \\ \overline{w_i} - w'_{i,j} & (otherwise) \end{cases}$$

他の手法では，タスク処理順番 t が遅いワーカに対して，処理時間の長いタスクが割り当てられることが多い．そこで，この手法ではタスクマネージャが期待タスク処理時間が長く，かつ残りタスク数が多いタスクをタスク処理時間の短いワーカに優先して割り当てる．タスク処理時間の長いタスクをタスク処理順番 t が遅いワーカに割り当らないようにすることで，全タスク処理時間を短くする手法である．

3.6 タスク処理アルゴリズム

上記５つのタスク優先度の計算手法を提案した．これらの割り当て手法の評価を行うために，タスクマネージャとワーカをエージェントとするシミュレーションを構築し，割り当て手法の評価をう．以下にシミュレーションのアルゴリズムを述べる

Algorithm 1:タスク処理アルゴリズム

$c_{i,0} \leftarrow 0 \ \forall(i)$
for $t \leftarrow 1 \ to \ T$
　タスク数の更新を行う
　$c_{i,t} \leftarrow c_{i,t-1} + g_{i,t}$
　タスク優先度から割り当てるタスクを決定する
　$x_{i,a(t)} = \begin{cases} 1 & (if \ \arg\max_i(p_{i,a(t)} \mid c_{i,t} \geq 1)) \\ 0 & (otherwise) \end{cases}$
　タスク数の更新を行う
　$c_{i,t} \leftarrow c_{i,t} - 1 \quad (x_{i,a(t)} = 1)$

このアルゴリズムにより，$x_{i,a(t)}$を導出し目的関数(式 1)に代入することで，全タスク処理時間を求める．

4 シミュレーション

本章では，株式会社調和技研が運営しているイベント情報サイト「びもーる」[7]からイベント記事作成タスクのデータを利用して，前章で提案した５つのタスク優先度を用いたタスク割り当て手法の評価を行う．

4.1 イベント情報サイト

イベント情報サイト「びもーる」は札幌を中心とした 6 都市のお祭りやコンサート，カルチャースクールなどのイベント情報を収集，配信するサービスである．現在，392 個のイベント収集クローラーが月に平均 10,000 件以上のイベント情報を収集して，15 万人以上のユニークユーザーに配信している．以下，イベント情報サイトとは「びもーる」を指す．

イベント情報を配信するためには，広告主や web 上から収集した様々な形式の情報を，イベント情報サイトで指定されている形式への整形，記事タイトルの決定，イベントジャンルの分類を行う必要がある．本稿ではこのイベント情報の編集作業をワーカの処理するタスクとする．

イベント情報を配信するための作業にかかる時間は，編集作業を行うワーカの知識や能力に大きく影響を受ける．イベント情報サイトでは 392 種類の元記事ページからイベント情報の作成を行っており，本稿では元記事ページの違いをタスクの種類とする．

現在，このタスクは障がい者施設「札幌チャレンジド」の方々がワーカとして図 1 に示されるイベント情報作成ページにおいて行っている．タスク処理時間を短くすることによって，ワーカの負担軽減になることも期待される．また，イベント情報サイトで作成されたすべてのイベント情報に関して，管理者が品質のチェックを行っている．よって本稿においてはタスクの質については十分に保証されているものとする．また，品質のチェックにかかる時間はタスク処理時間に加算されている．品質の低い編集結果に対しては管理者が修正を加えるため，平均処理時間が長くなっている．

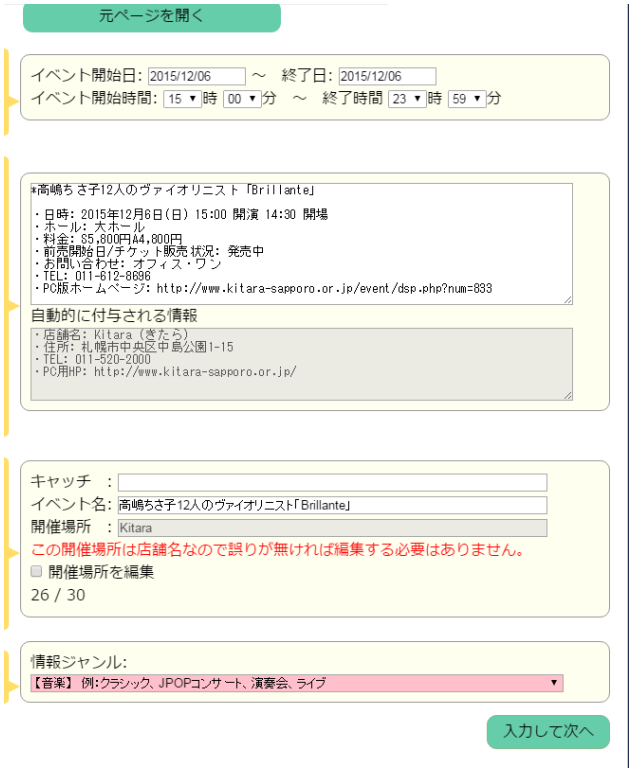

図 1:イベント情報作成ページ

4.2 タスク割り当ての効果

タスクの割り当てを行う際に，各ワーカのタスク種類ごとの処理時間の分散が小さければ，タスクの割り当てを行っても全タスク処理時間の変化は少ない．そこで，割り当ての効果を調べるために，イベント情報サイトの特徴として 2014 年 11 月 12 日から 2015 年 9 月 28 日の期間で処理された 257 種類のタスクと 36 名のワーカのタスク処理時間の平均と標準偏差を求めた．

	平均タスク処理時間(s)	標準偏差	変動係数
タスク種類別	203	59.8	0.292
ワーカ別	301	203	0.674

表 1:平均タスク処理時間と標準偏差

表 1 に示されているように，タスク別の変動係数は 0.292 であり，またワーカ別の変動係数は 0.674

であったことから，タスクとワーカに対してばらつきが大きいものと考えられ，タスク割り当ての効果が期待できる．

4.3 予備実験

割り当て手法の有用性を検証するために，イベント情報サイトのデータセットを用いて，シミュレーションによる予備実験を行う．

データセットは 2014 年 11 月 12 日から 2015 年 9 月 28 日の間に処理されたタスク，56,804 件のイベント情報を用いた．データセットを参考に，1 日毎にタスク割り当て問題を分割し，ワーカの期待タスク処理時間$w'_{i,j}$，タスクの処理順番$a(t)$のパラメータを代入，またタスク増加数$g_{i,t}$は 1 日に処理されたタスク数を代入した．また，期待タスク処理時間はデータセットの期間において，ワーカが処理を行ったタスクの平均処理時間であり，一定の値をとる．

これらのパラメータにより，実際のイベント情報サイトでのタスクと同じ状態で期間の日数分のシミュレーションを行い，平均タスク処理時間を求めた．

4.3.1 予備実験 1 タスク処理時間の比較

シミュレーションを行い，平均タスク処理時間を計算する．全てのエージェントの出現において，タスク処理順番$a(t)$が既知として線形計画法を適応した場合，導出された平均タスク処理時間は最小値となる．実際の作業時間，線形計画法の値を使い，提案手法による平均タスク処理時間の比較を行った．ここでの実際の作業時間とは，イベント情報サイト上で実際に割り当ったタスクを処理させた場合の平均処理時間のことを指す．

4.3.2 予備実験 1 結果

シミュレーションを行い，それぞれの手法の平均タスク処理時間と，実際の時間に対する減少率を表２に表す．

	平均処理時間(s)	削減率(%)
実際の作業時間	182	－
処理時間優先	147	19.1
処理順位優先	151	17.3
処理時間偏差優先	146	19.8
正規化処理時間偏差優先	146	19.7
残りタスク数優先	146	20.1
線形計画法	135	25.8

表 2:予備実験 1 結果

この結果から，実際の時間に対して，すべての提案手法において，17%以上の処理時間の減少が見られた．これらの結果からイベント情報サイトに導入した場合，最大で 20%のタスク処理時間の減少が可能である．

4.3.3 予備実験 2 タスク割り当て手法の比較

予備実験 1 ではワーカのタスク処理順番$a(t)$が決まっており，その順番に偏りがある場合に割り当て手法の評価を行うことは難しい．そこでタスク処理順番$a(t)$の偏りをなくすために，ランダムに順番を入れ替えたシミュレーションを 100 回行い，それらの平均と標準偏差の比較を行い，割り当て手法の評価を行う．

4.3.4 予備実験 2 結果

シミュレーションを 100 回行った平均の結果として，平均タスク処理時間，標準偏差を表３に表す．

	平均処理時間(s)	削減率(%)	標準偏差
実際の作業時間	182	－	－
処理時間優先	143	21.4	0.0826
処理順位優先	147	19.2	0.0901
処理時間偏差優先	142	21.9	0.0646
正規化処理時間偏差優先	143	21.4	0.0689
残りタスク数優先	141	22.5	0.0625
線形計画法	135	25.8	－

表 3:予備実験 2 結果

上記の結果から，残りタスク数優先手法がタスク処理時間の平均が 5 つの手法の中では最良であり，また最適解である線形計画法と比較しても，3%程度の差であった．また，標準偏差においても，他の手法よりもばらつきが小さく，タスク処理順番に左右されないことが分かった．これは残りタスク数優先は他の手法とは違い，残りのタスク数から将来のワーカに割り当てるタスクを考慮しているので，平均タスク処理時間が短くなったと考えられる．

5 実作業への導入

本章では，前章の 2 つの予備実験で最良の結果であった残りタスク数優先手法を実際のイベント情報の編集作業に導入することを考える．

実作業へ導入を行うために，実際のイベント情報の特徴である編集締め切り日を考慮したタスク割り

当て手法を検討し，予備実験によって実証実験で用いるタスク間隔のパラメータを決定する．また，「札幌チャレンジド」の作業環境にタスク割り当て手法を導入した実証実験で提案手法の効果を検証する．

5.1 予備実験

5.1.1 予備実験 3 タスクグループの生成間隔の評価

イベント情報の編集作業の特徴として，タスクである編集作業には異なる編集締め切りが存在することが挙げられる．また，イベント情報を公開する期間が存在し，早くタスクを処理すれば公開期間を長くすることができる．現在行われている割り当てでは，イベントの開始日が早いタスクの順番で割り当てているため，締め切りが過ぎるタスクは少ないが，公開期間やタスク処理時間は考慮されていない．そこで本稿において，これらの要素を考慮して，割り当てを行うために，締め切り日に応じてタスクグループの生成を行い，そのタスク群タスクの割り当てを行う．

タスクの割り当てにおいて，全てタスクを候補としてタスク割り当てを行うと，締め切りについては考慮されない．そこで，タスクの締め切りまでの時間に基づいて残りのタスクを抽出することにより，締め切りを超過するタスクを削減することができる．また，締め切り日までの時間の間隔を変化させてタスクグループとして生成することにより，タスクの処理時間，公開期間，締切超過タスク数に影響するため，タスクグループを生成する間隔を検討する必要がある．

タスクグループを生成する間隔を決めるためにシミュレーションを実行し，平均タスク処理時間，締切超過タスク数，平均公開期間の比較を行った．

データセットは 2016 年 9 月 1 日から 2015 年 12 月 24 日の間に収集されたタスク 32,571 件のイベント情報を用いた．ワーカのタスク処理時間$w_{i,j}$，タスクの処理順番$a(t)$，1 日に収集されたタスク数$g_{i,t}$をパラメータとして，シミュレーションを実行した．結果を表に示す．（$d = \{1,2,\dots,D\}$であり，またタスク間隔は少数点以下切り捨てで計算している）

タスク間隔	平均処理時間(s)	締切超過タスク数	平均公開期間(日)
1 日間隔	144	47	23.2
7 日間隔	139	79	23.2
30 日間隔	134	300	23.2
2^d日間隔	133	48	23.2
$\exp(d)$日間隔	132	48	23.2
$\exp\left(\frac{d}{0.9}\right)$日間隔	131	48	23.2
$\exp\left(\frac{d}{1.1}\right)$日間隔	133	48	23.2

表 4:予備実験 3 結果

5.1.2 予備実験 3 結果

タスクグループの生成間隔ごとの平均処理時間，締切超過タスク数，平均公開期間を表 4 に示す．

表 4 に示された予備実験 3 の結果からタスクグループの生成間隔を短くすると，締切超過タスク数は減少するが，平均タスク処理時間は増加する．タスクグループの生成間隔を長くすると，平均タスク処理時間は減少するが，締切超過タスク数は増加することが分かった．これらを同時に満たすように指数的に生成間隔を変化させることで平均処理タスク時間を短縮し，かつ締切超過タスクを減少させた．これらの結果から，平均タスク処理時間が短く，また締切超過タスク数が少ない$\exp\left(\frac{d}{0.9}\right)$ $d = \{1,2,\dots,D\}$のタスク間隔でタスクグループを生成する． また，ワーカのタスク処理時間は，時間経過による，タスク処理能力の向上や，タスクの質の変化により変動することが考えられる，よって一定の確率で探索を行う必要がある．本稿では， ε -greedy により行動を選択した．タスク間隔 $e^{\frac{d}{0.9}}$においては，締め切りが近い場合において，タスク間隔が短いため，ある程度タスクが分散するため， ε の値は十分に小さい 0.05 と設定した．

5.2 実証実験

5.2.1 実証実験 タスク割り当て手法の導入

2016 年 1 月 5 日から 2016 年 1 月 26 日の期間に「札幌チャレンジド」でタスク処理を行ったワーカ 12 名，タスクの種類 248 個の作業環境において，タスク割り当て手法を導入した．このとき，期待タスク処理時間の更新は簡略化のために，1 日に 1 度とした．

5.2.2 実証実験 結果

表 5 にタスク割り当て手法導入前の平均タスク処理時間との比較を以下の表に示す．この期間において，7,136 個のタスクが処理された．

	平均処理時間(s)	削減率(%)
導入前	161	-
導入後	146	9.3

表 5:実験結果

上記の結果より，割り当て手法導入前と比較し，9.3%のタスク処理時間の削減を行えることを確認した．平均公開日数は 18.1 日，締切超過タスク数は 0 であった．また，ワーカ毎，タスク毎の平均タスク処理時間を比較し， 50%のワーカと，60%のタスクにおいて割り当て手法導入前より平均タスク処理時間が減少した．これはワーカにおいては，タスク処理時間が大きいタスクが割り当てられるワーカにおいては割り当て手法導入後であっても平均タスク処理時間が長くなってしまう場合がある．しかし，タスクにおいては，平均よりもタスク処理時間が短いワーカに割り当てられるので，多くのタスクが割り当て手法導入後に平均タスク処理時間が短くなるという結果であったと考えられる．

5.3 考察

実作業環境にタスク割り当て手法を導入することにより，9.3%の平均タスク処理時間の削減を行えることを確認した．札幌チャレンジドの 1 ヶ月の作業の内，約 30 時間の作業時間の節約が見込める．

また，2016 年 1 月 22 日に 4 名のワーカに対して割り当て手法に関するヒヤリングを行った．割り当て手法の導入前後について，4 人全員が，手法導入後の方がタスクを処理しやすいとの意見を頂いた．また割り当て方法については，同じ種類のタスクばかりではモチベーションが低下するので，異なる種類のタスクも割り当てた方が良いとの意見を頂いた．本稿においては，ワーカのモチベーションについては考慮できていないので，今後のワーカのモチベーションに考慮した割り当て手法を考慮する必要がある．

6 おわりに

本稿ではマイクロタスク割り当て問題において，タスク処理時間の最小化のために，5 つの割り当て方法を提案した．割り当て方法の評価を行うために，実際に運営されているイベント情報サイトのデータを使用したシミュレーションを実行した．その結果，平均タスク処理時間が最小であった残りタスク数優先手法をイベント情報サイトの記事作成タスクに導入して，評価を行った．結果として，割り当て手法導入前と比較し，9.3%の処理時間の短縮を確認した．

謝辞

本研究の遂行に必要なデータを提供していただいた札幌チャレンジド,「びもーる」の運営元である株式会社調和技研に謝意を表します．

参考文献

[1] Kittur, Aniket. "Crowdsourcing, collaboration and creativity." *ACM Crossroads*17.2 (2010): 22-26.

[2] 芦川将之, 川村隆浩, and 大須賀昭彦. "マイクロタスク型クラウドソーシングプラットフォーム環境における精度向上手法の導入と評価." *人工知能学会論文誌* 29.6 (2014): 503-515.

[3] Ho, Chien-Ju, and Jennifer Wortman Vaughan. "Online Task Assignment in Crowdsourcing Markets." *AAAI*. Vol. 12. 2012

[4] Abdulkadiroğlu, Atila, and Tayfun Sönmez. "Random serial dictatorship and the core from random endowments in house allocation problems." *Econometrica*(1998): 689-701

[5] 松原繁夫，and 水島拓也．″クラウドソーシングにおける複数タスク割当て．″ 人工知能学会全国大会,(第 27 回) (2013)

[6] 「あなた情報マガジンびもーる」 http://bemall.jp/