DBSJ Japanese Journal
Vol. 14-J, Article No. 2, March 2016

クラウドソーシングの時空間拡張におけるワーカの時間制約を考慮したタスク割当て手法

Task Assignment by Referring to Worker's Temporal Constraint in Mobile Crowdsourcing

羽田野 真由美♡ 中辻 真♢
戸田 浩之♠ 小池 義昌♣

Mayumi HADANO Makoto NAKATSUJI
Hiroyuki TODA Yoshimasa KOIKE

本研究は，動的に変化する時空間情報を出来るだけ早く収集することで，よりきめ細やかな行動支援情報をユーザに提供することを目的とする．そこで，低コストで不特定多数の群衆に仕事を委託する枠組みであるクラウドソーシングに着目し，これを用いて動的な時空間情報を収集するシステムを提案する．提案システム実現のため，データ収集の短納期と高い網羅性の 2 つを目指し，タスクをワーカに割当てる手法に取り組む．既存手法では，ワーカは常にタスクが実行出来るという想定での最適化を行っているが，現実ではワーカの働く時間は限られており，ワーカの位置も頻繁に変化する．そこで本研究ではワーカの将来行動を把握した上で将来にわたってのタスク割当てを行うことで，移動範囲の広いワーカに優先的にタスクを割当てることを可能にする．評価実験の結果，提案手法は従来手法と比較して割当てタスク数を約 7%増加させることを確認した．

This paper focuses on task assignments to workers in mobile crowdsoucing systems. The current method of assigning tasks to workers creates a directed graph that represents the relationships between workers and tasks, and it finds the best set of worker-task pairs by solving a min-cost max flow problem on it. It, however, does not assign tasks so well since it considers only workers who are ready to work at the time of optimization, whereas in reality workers tend to change their activities over time. To tackle with this problem, we propose a new method that incorporates two new ideas: (1) it creates a 'time-extended' worker-task graph that expresses the relationships between workers and tasks over a time period by referring to workers' schedules; (2) it handles information on workers that changes over time and finds the best set of worker-task-time triples by using the time-extended worker-task graph. We evaluated our method by using real-world visiting logs. The results show that our method increases the rate of assigned tasks by more than 6% compared with another state-of-the-art assignment method.

♡ 非会員 日本電信電話株式会社 NTT サービスエボリューション研究所
hadano.mayumi@lab.ntt.co.jp
♢ 正会員 NTT レゾナント株式会社
nakatuji@nttr.co.jp
♢ 正会員 日本電信電話株式会社 NTT サービスエボリューション研究所
toda.hiroyuki@lab.ntt.co.jp
♠ 非会員 日本電信電話株式会社 NTT サービスエボリューション研究所
koike.y@lab.ntt.co.jp

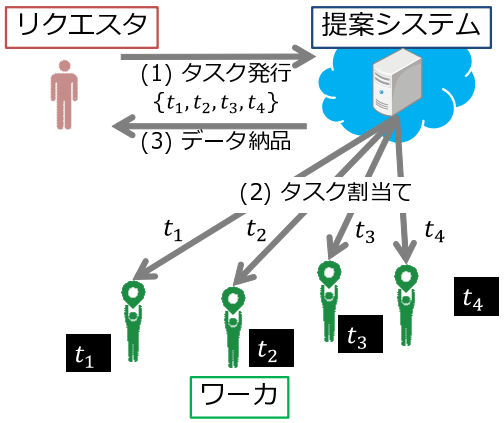

図 1: クラウドソーシングを利用した地域情報収集イメージ

1. はじめに

1.1 社会的な背景

地図のデジタル化に伴い，地図情報が手軽に閲覧・加工出来るようになったことを背景にして，カーナビゲーションサービスや経路検索サービスなどの地理空間情報サービスの国内市場が急速に拡大している [1]．国内における 2013 年の地理空間情報サービス市場は 10 兆円と試算され，その 8 割近くが観光情報提供や地域情報検索，交通情報提供など，ユーザの街中での行動支援を行うサービスである．一般に，このようなサービスでは，基盤地図の情報と独自データベースを利用してユーザが望む場所の情報を提供することでユーザの行動支援を行っている．例えば，飲食店検索サービスでは飲食店データベースを基盤地図上に表示することで食事行動の支援を行っており，カーナビゲーションサービスでは交通規制データベースなどを基盤地図上に表示することで目的地までの移動の支援を行っている．

一方，モバイル通信端末の普及に伴って，ユーザは自身の直後の行動を決定するために必要な情報にアクセスする機会が増えた．しかしながら，サイバー空間には存在しない情報が多く存在するため，ユーザの取りうる行動範囲には限界があると考えられる．我々が 500 人を対象に実施したアンケートによると，内 8 割の人が時間や場所に紐づいた情報が入手できずに困ったことがあると回答している．その情報には店舗混雑度や公共交通機関の運行情報，商品の在庫情報などが挙げられ，数十分から数時間単位で動的に変化する情報が多く含まれた．従来このような動的な情報にアクセスすることは，運営機関が独自に情報提供している場合を除き，極めて困難である．近年では Twitter をはじめとするソーシャルメディアでユーザが発信した情報を分析することで，災害の発生 [2] やイベントの発生 [3] を検知する取組みも存在する．しかしながらこのアプローチは発信ユーザの知りうる情報しか得られないために時間や場所に対しての網羅性に欠けるという問題がある．一方，情報取得の即時性を気にせずに場所の情報だけを収集する場合は，調査員を派遣した現地調査が行われることが多い．例えば，地図作成会社が更新地点の確認のために調査員を派遣したり，警備会社が災害発生時の被害状況の確認のために警備員を派遣したりする例が存在する．しかしながら，専門調査員の派遣は，移動時間が多くかかり経済的負担も大きいという問題がある．具体的には，1 地点当たり 1500 円から 3000 円で委託されている例が存在する．

1.2 本研究のアプローチ

本研究の目的は，動的に変化する実世界情報を出来るだけ早く収集することで，ユーザの行動決定に関わる利便性を向上させることを目的とする．例えば，ユーザが現在地周辺で食事をする店を

選ぶ際，混雑情報を併せて提供することで空いている店舗を選択出来るような，きめ細やかな行動支援の実現を目指す．そこで本研究は，不特定多数の群衆に仕事を委託する枠組みであるクラウドソーシング [4] に着目し，これを用いて動的な時空間情報を収集するシステムを提案する (図 1)．クラウドソーシングは安価に作業を外注出来る枠組みとして近年注目を集めており，現地にいる群衆に現地調査の仕事を委託することができれば，調査員を派遣する場合と比較して経済的で即時に情報収集を行えるメリットがある．また，ソーシャルメディアでユーザが誰も発信していない場所であっても所望の場所へ人を派遣することが出来るため網羅性が保証されると考えられる．

図 1 に我々が提案するシステムを示す．クラウドソーシングを用いた情報収集は，まず (1) 仕事を依頼する人 (リクエスタ) が時空間条件の指定された情報収集の仕事 (タスク) をシステムに依頼し，(2) 指定された場所・時間にいる人 (ワーカ) がスマートフォン経由でタスクを行い，(3) 調査結果をリクエスタに返すという 3 つのプロセスによって実現される．

1.3 本研究が取組む技術的課題

動的な時空間情報を収集するために特に重要な観点は，(a) 網羅性：締切時刻内に出来るだけ多くのタスクを完了させること，(b) 即時性：その中でも出来るだけ短い時間でタスクが完了されることの 2 つである．本研究では以上の要件を満たした効率的な情報収集のために，ワーカにタスクを割当てる手法の開発に取り組む．

タスク割当てを実現するための先行研究には，タスクとワーカの制約条件のもとで，ある評価指標を最適化する数理最適化手法を取るものが多い [5][6]．しかしながら，これらの手法は (1) 時刻によってワーカが働けるかどうかが変化する状況下での非効率な割当てや，(2) ワーカの現実的な振る舞いを反映しない設定でのシミュレーションが行われていることが課題として挙げられる．

(1) の課題と本研究の解決方法について説明する．既存研究ではワーカが常に利用可能であるという想定を置いている．そのため，ワーカの利用可能状態が時刻によって変化する状況下での効率的な割当てを行うことができないという課題が存在する．本研究では，(i) 効率的な仕事計画を立てるためにワーカはスケジュールをシステムに入力する，または (ii) システム利用履歴から日常行動の将来予測が可能，という前提が成立すると考えワーカの将来行動を把握した上で将来にわたってのタスク割当てを行う．結果，移動範囲が広いワーカの出現を待って優先的にタスクを割当てることが出来る．これにより従来手法よりも網羅性と即時性のあるタスク割当てが実現できる．

(2) の課題と本研究の解決方法について説明する．既存研究では，商用のチェックインサービスのユーザログデータを利用し，そのユーザを擬似的にワーカとして扱うものが多い．しかしながら，日常行動のログデータから能動的に仕事を行うワーカをシミュレーションするのは現実に即しておらず，評価データとして適していないおそれがある．そこで，本研究では事前にワーカ候補を対象にしたアンケート調査を行ってワーカが働く際の振る舞いを調べ，実験のパラメータに利用することで，より現実に即したシミュレーション実験を実現した．

本研究の貢献をまとめると以下のようになる．

- 出来るだけ多くのタスクを締切時刻までに割当てるとともに，タスク完了時刻を最短にする効率的な自動タスク割当て手法の提案．
- ワーカ候補へアンケートを実施し，得られたパラメータをシミュレーションに利用する実験法の提案．
- シミュレーション実験の結果，従来手法比で割当てタスク数を約 7%増加させ，平均タスク所要時間を約 8%削減させることを確認．

本稿の構成は以下の通りである．まずで関連研究について概観し，次にで問題設定について説明する．で提案手法について述べ，その手法の評価実験の方法と結果をで述べる．最後にで本稿をまとめる．

2. 関連研究

クラウドソーシングにおけるタスクの選択にはワーカによる自由選択とシステムによるタスクの自動割当てとの 2 つのアプローチが存在する．ワーカによる自由選択は一般のクラウドソーシングサービス [1] で多くみられる形式である．しかしながら，時空間の条件が指定されたクラウドソーシングでは，指定場所への移動という高コストな付帯業務が発生するため，ワーカによる自由選択では非効率な移動が数多く起こるおそれがある．その結果，ワーカに対する負担が大きくなるとともに，タスク完了時間の遅延や収集データの網羅率の低下により，リクエスタの満足度が低下するおそれがある．さらに，タスクが近距離に存在するのをワーカ自身が気付かない可能性も存在する．

一方でシステムによるタスクの自動割当ては，システムがワーカの情報を利用して効率的にタスクを割当てることが可能である．また，ワーカ自身が気づかない近距離のタスクをジオフェンシング [7] のようにプッシュで通知する機能としても利用出来るため，本研究ではシステムによるタスクの自動選択のアプローチを選択した．

クラウドソーシングにおけるタスクの自動割当ての研究は，タスクに対する地理的条件の有無で 2 つに大別される．地理的条件のない通常のクラウドソーシングにおけるタスク割当ての研究には，ソーシャルメディアのプロフィール情報などの外部情報を利用したタスク推薦アプローチ [8] や，ワーカの能力を推定しながらタスク割当てを行うオンラインアプローチ [9] [10] などが存在する．しかしながら，これらの手法は地理的条件を持つタスクを対象にしていないため，本研究の目的には一致しなかった．

地理的条件を持つクラウドソーシングにおけるタスク割当ての研究では，タスクとワーカの位置情報から計算した制約条件下で，ある評価指標を最適化する手法を取るものが多い．例えば，Reddy ら [11] は，タスクの時空間的なカバー率の最大化を予算付き最大被覆問題に定式化し，貪欲法によって解く手法を提案している．しかしながら，この手法は比較的狭い空間範囲でのタスク割当てを対象にしているため，ワーカの移動範囲に上限を設けていない．したがってタスクとワーカ間の距離が遠い場合でもタスクを割当てる事象が発生しうる．本研究はワーカの移動範囲にはある上限があるという想定のもとでのタスク割当て手法を提案するものである．

一方，Kazemi ら [5] はワーカの移動範囲の制約下での完了タスク数最大化を，有向グラフにおける最大フロー問題に定式化する手法を提案している．この問題設定をそのまま引き継ぎ，ワーカの信頼度を考慮した研究 [12] や複合タスクの完了数を最大にする研究 [6] も存在する．しかしながらこれらの研究は，ワーカが常に利用可能であるという想定を置いているため，ワーカの利用可能状態が時間によって変化する状況下で効率的に割当てることができない．本研究は，Kazemi ら [5] にならい，ワーカの移動範囲の上限を制約としたタスク割当て問題を扱うとともに，ワーカの利用可能状態が時間によって変化する状況下での効率的な割当て手法を提案する．

また，Chen ら [13] はワーカの 1 日での総移動時間に上限を設け，割当てタスク数の最大化を整数計画問題に定式化する手法を提案している．具体的には，既存研究 [14][15] などによって移動ログから将来のワーカ移動を予測出来るとした上で，ワーカの移動系列中のタスク割当てを行っている．本研究も Chen らのように，ワーカの将来行動はワーカによってシステムに入力されている，あるいは既存研究で予測出来るとした上で，タスク割当てを行う．

[1] 例えば Amazon Mechanical Turk(https://www.mturk.com/mturk/) やランサーズ (http://www.lancers.jp/) など

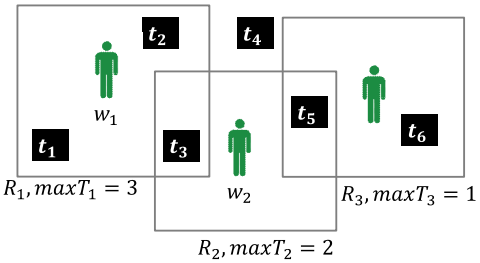

図 2: ワーカとタスクの例

3. 自動タスク割当ての問題設定

この章では，地理的条件を持つクラウドソーシングにおける自動タスク割当ての問題設定について説明する．

3.1 用語の定義

はじめに，用語の定義を述べる．

定義 1 (タスク) タスク $t_i \in T$ は，リクエスタがシステムに依頼した時空間上の仕事のことである．ここで，T はタスク集合のことである．タスク t_i は二次元平面における位置 (x_i, y_i) と有効期限の時刻を示す締切時刻 d_i を属性として持つ．また，タスク t_i は後述するワーカが締切時刻 d_i までにその場所 (x_i, y_i) に移動してから実行するような仕事を指す．タスクの例としては，指定された場所の建物の写真を撮影することや，ある店舗の混雑度を報告すること，商品の在庫数を報告することなどが挙げられる．図 2 の例では，t_1 から t_6 の合計 6 つのタスクが存在している．

定義 2 (ワーカ) ワーカ $w_j \in W$ は，クラウドソーシングサービスにてタスクを実行する人のことである．ここで，W はワーカ集合のことである．ワーカ w_j は二次元平面における位置 (x_j, y_j) と最大許容エリア R_j，最大許容タスク数 $maxT_j$ を属性として持つ．最大許容エリア R_j とは，ワーカがタスクを行うために移動出来る範囲のことで，ワーカはその範囲内であれば自由に移動することが出来るものとする．最大許容タスク数 $maxT_j$ とは，ワーカが許容出来るタスク数の上限のことである．図 2 の例では，w_1 から w_3 までの合計 3 人のワーカが存在しており，それぞれの最大許容エリア R_j を矩形で示し，最大許容タスク数 $maxT_j$ を矩形の下に表示している．

定義 3 (タスク割当てグラフ) タスク割当てグラフ $G(V, E)$ とは，タスクとワーカの情報が表現された有向グラフのことである．頂点集合 V と枝集合 E を属性としてもち，各枝 $e(\in E)$ は流量 $f(e)$ と流量の上限を示す容量 $\bar{f}(e)$，流量あたりのコスト $c(e)$ を属性として持つ．

以下にタスクとワーカの情報を有向グラフに表現する手順を説明する．まず，(1) 頂点集合 V は $|W| + |T| + 2$ 個の頂点を持ち，各ワーカ $w_j \in W$ を頂点 v_j に，各タスク $t_i \in T$ を頂点 $v_{|W|+i}$ に対応させる．また，湧き出し頂点である v_{src} と吸い込み頂点である v_{sink} を加える．次に，(2) 枝集合 E は $|W| + |T| + m$ 個の枝を持ち，頂点 v_{src} から W に対応した頂点をつなぐ枝を $|W|$ 本，T に対応した頂点から頂点 v_{sink} をつなぐ枝を $|T|$ 本つなぐ．さらにワーカ w_j が対応する頂点からは，ワーカ w_j の最大許容エリア R_j に含まれるタスクに対応する頂点をつなぐ枝を結び，これを全ワーカ分の合計 m 本つなぐ．(3) 最後に，頂点 v_{src} から w_j に対応した頂点をつなぐ枝の容量 $\bar{f}$ に，ワーカ w_j の最大許容タスク数である $maxT_j$ を設定し，その他の枝の容量に 1 を設定する．

図 2 に図 3 のタスクとワーカの情報を表現したタスク割当てグラフの例を示す．枝に示した数字はその枝の容量を示している．頂点 v_1 から v_3 がワーカに，頂点 v_4 から v_9 がそれぞれタスクに対応しており，ワーカとタスク間の枝はワーカの最大許容範囲内にタスクが存在する場合のみ結ばれている．

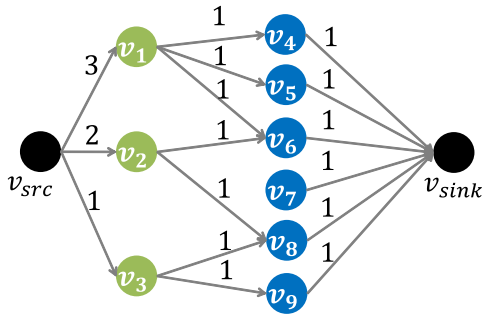

図 3: タスク割当てグラフの例

3.2 グラフを用いた従来の最適化計算

ここでは，タスク割当て問題を有向グラフを用いた最適化計算によって実現する方法について説明する．1 章において本研究の要件として挙げた網羅性と即時性を (1) 最大タスク割当て問題，(2) 最小コスト割当て問題の 2 段階の最適化問題を解くことよって実現させる．なおこれらはタスク割当てグラフにおける (1) 最大フロー問題，(2) 最小コストフロー問題の最適化問題に対応している．以下にそれぞれの最適化問題の定義とその計算方法について述べる．

定義 4 (最大タスク割当て問題) 最大タスク割当て問題は以下に述べる制約下でワーカに割当てたタスクの数を最大化する処理のことである．ここで制約とは，ワーカ w_j の最大許容エリア R_j 内にタスク t_i が存在し，ワーカ w_j の最大許容タスク数に余裕がある場合のみ，タスク t_i をワーカ w_j に割当てることが出来るというものである．

最大タスク割当て問題はタスク割当てグラフを用いることで，有向グラフの湧き出し頂点 v_{src} と吸い込み頂点 v_{sink} 間の流量を最大化する最大フロー問題 [16] に定式化出来る．一般に，最大フロー問題は線形計画法で解くことが出来るほか，効率的に解くアルゴリズムも多く存在する [16][17].

以下に最大フロー問題として定式化する際の目的関数と制約式を示す．$G = (V, E)$ をタスク割当てグラフとし，各枝 $e \in E$ の上限容量を $\bar{f}(e) > 0$，流量を $f(e) > 0$ とする．また，頂点 v を始点とする枝集合を $\delta^+ v$，頂点 v を終点とする枝集合を $\delta^- v$ とすると，最大フロー問題の目的関数は以下の通りに表すことが出来る．

$$\max \sum_{e \in \delta^+ v_{src}} f(e) \tag{1}$$

また，制約条件は以下の通りである．

$$0 \leq f(e) \leq \bar{f}(e) \quad (\forall e \in E) \tag{2}$$

$$\sum_{e \in \delta^+ v} f(e) - \sum_{e \in \delta^- v} f(e) = 0 \ (\forall v \in V \backslash \{v_{src}, v_{sink}\}) \tag{3}$$

式 (2) は各枝の容量制約を示している．また，式 (3) は湧きだし頂点と吸い込み頂点以外の頂点における流量保存則を示しており，流入量と流出量の大きさが等しくなることを表している．

定義 5 (最小コスト割当て問題) 最小コスト割当て問題は，以下に述べる制約のもとでタスクを実行する際の総コストを最小化する処理のことを示す．ここで制約とは，最大タスク割当て問題における制約式 (2)(3) に，割当てタスク数が節で得られる最大流量 f_{max} と一致するという制約を加えたものである．

先ほどと同様にタスク割当てグラフを用いることで，最小コスト割当て問題は最小コストフロー問題に定式化することが出来る．最小コストフロー問題とは，与えられた流量を有向グラフの湧き出し頂点 v_{src} と吸い込み頂点 v_{sink} 間に流した際の各枝の総コストを最小化する問題のことである．この際，流量が節で得られる最

表 1: 記号一覧

記号	定義
ϕ_k, φ	時刻，時刻集合
$t_i, T(\phi_k)$	タスク，ある時刻 ϕ_k のタスク集合
$w_j, W(\phi_k)$	ワーカ，ある時刻 ϕ_k のワーカ集合
$W(\varphi)$	全時刻のワーカ集合
$R_j(\phi_k)$	時刻 ϕ_k のワーカ w_j の最大許容エリア
$maxT_j$	全時刻を通したワーカの最大許容タスク数
$\mathcal{CP}$	割当て候補三組み集合
f_{max}	最大割当てタスク数
c_{min}	ワーカとタスク間の最小コスト
$\mathcal{MT}$	ワーカとタスクとコストの三組み集合
$\mathcal{P}$	最小コストペア集合
$\mathcal{T}$	最小コスト三組み集合

大流量 f_{max} と一致するような制約式を加えれば，最大流量を流した際のコストを最小にする組み合わせを計算出来る．一般に，最小コストフロー問題は，線形計画法で解くことが出来る．

以下に最小コストフロー問題を定式化する際の目的関数と制約式を示す．最小コスト最大タスク割当て問題の目的関数は以下の通りに表すことが出来る．

$$\min \sum_{e \in E} f(e)c(e) \tag{4}$$

また，制約条件は式 (2) と式 (3) に，以下の式 (5) を加えたものとなる．

$$\sum_{e \in \delta^+ v_{src}} f(e) = f_{max} \tag{5}$$

最大フロー問題を解き，そこで得られた最大流量を最小コストフロー問題を解く際の入力とした 2 段階の最適化計算を行うことで，最大タスク割当て数を保証しながら最小コストとなるワーカとタスク間の組み合わせを計算することできる．

4. 提案手法

本章では，時刻によってワーカ集合および最大許容範囲が変化する状況下でのタスク割当て方法を提案する．既存研究ではワーカが常に利用可能であるという想定を置いているため，ワーカの利用可能状態が時間によって変化する状況下では，効率的な割当てを行うことができていない．本研究では，(1) 効率的な仕事計画を立てるためにワーカはスケジュールをシステムに入力する，または (2) システム利用履歴から日常行動の将来予測が可能，という前提が成立すると考えワーカの将来行動を把握した上で将来にわたってのタスク割当てを行う手法を提案する．表 1 に用いる記号の一覧を示す．

提案手法におけるワーカとタスクの状態について以下に説明する．まず時刻集合 φ を導入し，時刻によってワーカの利用可能状態と最大許容エリア R_j が変化することを考える．ここで，ある時刻 ϕ_k におけるタスク集合を $T(\phi_k)$ とし，時々刻々と出現するものとする．また，全時刻におけるワーカ集合を $W(\varphi)$ とし，ワーカのスケジュールを参照することで把握可能とする．ある時刻 ϕ_k におけるワーカを $w_j(\phi_k)(\in W(\phi_k))$，最大許容エリアを $R_j(\phi_k)$ とする．なお，最大許容タスク数 $maxT_j$ は与えられた時刻の幅で設定されているものとする．

4.1 提案手法の概観

提案手法全体の構成図を図 4 に示す．提案手法は全時刻のワーカ集合 $W(\varphi)$ と各時刻のタスク集合 $T(\phi_k)$ を入力とし，タスク完了時刻を最小にするワーカとタスクと時刻の三組み集合 $\mathcal{T}$ を出力するものである．ワーカとタスクのペアを出力する従来手法と違い，本手法はワーカにタスクを割当てる時刻も合わせて出力する．提案手法の手順は大きく分けて (1) 最大タスク割当て計算，(2) 最小コスト割当て計算，(3) 割当て時刻計算の 3 つの処理に分けられる．

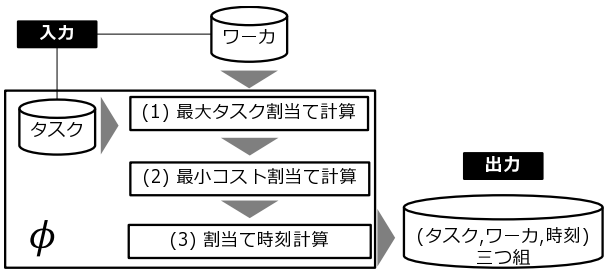

図 4: 提案手法の処理の流れ

(1) の最大タスク割当て計算では，全時刻のワーカ集合 $W(\varphi)$ とある時刻のタスク集合 $T(\phi_k)$ を入力とし，候補割当て三組み集合 $\mathcal{CT}$ を出力する．さらに，全時刻をまたいだ割当てタスク数の最大化を行い，その際の最大割当てタスク数 f_{max} を出力する．具体的には，あるタスクを実行可能なワーカを時刻をまたいで探索した結果を利用して 1 つのタスク割当てグラフを作成し，最大フロー問題を解く．ここでの計算方法については，節で詳しく述べる．

(2) の最小コスト割当て計算では，(1) で得られた割当て候補三組み集合 $\mathcal{CT}$ と最大割当てタスク数 f_{max} を入力とし，最小コストペア集合 $\mathcal{P}$ とワーカとタスクとコストの三つ組集合 $\mathcal{MT}$ を出力する．具体的には，ワーカとタスク間のコストとして最短タスク完了時刻を計算し，最小コストフロー問題を解く．ここでの計算方法については，節で詳しく述べる．

次に (3) の割当て時刻計算では，(2) で得られた最小コストペア集合 $\mathcal{P}$ を入力とし，最小コストのワーカとタスクと時刻の三組み集合 $\mathcal{T}$ を出力する．具体的には，割当てペア集合と一致する要素をもつ最適割当て三組み集合を抽出することで最適割当て三組み集合を出力する．ここでの計算方法については，節で詳しく述べる．

以上 (1) から (3) の提案手法の計算結果として，出来るだけ多くのタスクを出来るだけ早く完了させる時刻とワーカとタスクの三組みを得ることが出来る．

4.2 最大タスク割当て計算

全時刻のワーカ集合 $W(\varphi)$ とある時刻のタスク集合 $T(\phi_k)$ を入力とし，全時刻をまたいだ割当て候補三組み集合 $\mathcal{CT}$ と最大タスク数 f_{max} を出力する方法を手順に沿って述べる．

まずはじめに，ワーカとタスクの情報からタスク割当てグラフを作成する．既存手法 [5] ではワーカがいつでも仕事が可能であるとしていたため，1 時刻分のタスク割当てグラフを与えられた時刻集合の要素数だけ作成して順番に最適化を行っていた．本研究では時刻をまたいで 1 つのタスク割当てグラフを作成する手法を提案する．それにより，時刻によってワーカ集合やワーカの最大許容エリアが異なる場合を同時に扱うことが可能となり，効率的なタスク割当てを実現することが出来る．

以下，図 5 の例を用いてタスク割当てグラフ作成方法を説明する．ある時刻 ϕ_1 にタスク t_1，t_2 が発生している．また，時刻 ϕ_1 にワーカ w_1 が発生し，時刻 ϕ_2 にワーカ w_2 が発生している．また，ワーカ w_1，w_2 の最大許容範囲内にタスク t_1 と t_2 はどちらとも含まれているとする．この時，タスク割当てグラフに用いる割当て候補三組み集合は，$\{(t_1,w_1,\phi_1),(t_1,w_1,\phi_2),(t_1,w_2,\phi_2),(t_2,w_1,\phi_1),(t_2,w_1,\phi_2),(t_2,w_2,\phi_2)\}$ となる．次に，作成したタスク割当てグラフの最大フロー問題を解き，得られた最大割当てタスク数を出力する．この処理は，節で述べた通りに実行する．

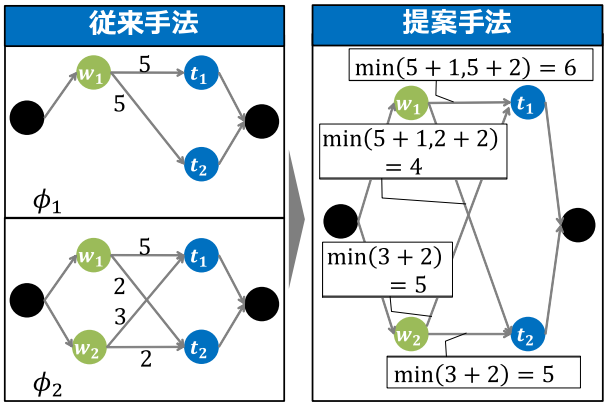

図 5: 提案手法におけるタスク割当てグラフ作成

4.3 最小コスト割当て計算

節で求めた割当て候補三組み集合 $\mathcal{CT}$ と最大割当てタスク数 f_{max} を入力とし，最小コストペア集合 $\mathcal{P}$，及びワーカとタスクとコストの三組み集合 $\mathcal{MT}$ を出力する方法を手順に沿って述べる．

まずはじめに，割当て候補三組みにおいて，あるワーカの割当て候補時刻をすべて取り出す．次に，候補時刻を利用してタスク t_i とワーカ w_j 間のコストを計算する．なお，本研究では出来るだけ早くタスクを完了させることを要件にしているため，コストにタスク完了時刻を利用する．また，タスク自体は写真撮影など簡単な作業を想定しているため，タスク自体に必要な時間を無視し，ワーカがタスクの位置まで移動し終えた時刻をそのままタスク完了時刻とする．ここで，φ_{cand} をワーカの候補時刻集合，$a(t_i, w_j(\phi_k))$ を時刻 ϕ_k のワーカ w_j がタスク t_i まで移動する際の所要時間とすると，タスク t_i とワーカ w_j 間のコスト c_{ij} は以下のように計算出来る．

$$c_{ij} = \min_{\phi_k \in \varphi_{cand}} (a(t_i, w_j(\phi_k)) + \phi_k) \tag{6}$$

つまり，$(t_i, a(w_j(\phi_k)) + \phi_k)$ は，時刻 ϕ_k にワーカ w_j がタスク t_i を割当てられた際のタスク完了時刻を示す．式 (6) を用いることによって，ワーカの候補時刻のなかで最小となるタスク完了時刻を計算することができ，複数時刻のワーカの情報を 1 つのグラフで表現することが可能となる．最後に，計算したコストの情報を利用してタスク割当てグラフを作成し最小コストフロー問題を解く．ここでのコスト最小化処理は，節で述べた通りに実行する．

以下，図 5 の例を用いて説明する．(t_1, w_1) の候補時刻 φ_{cand} は，$\{\phi_1, \phi_2\}$ であるので (t_1, w_1) 間のコストは，$\min(5 + 1, 5 + 2) = 6$ と計算出来る．同様にして (t_2, w_1) は $\min(5 + 1, 2 + 2) = 4$，(t_1, w_2) は $\min(3 + 2) = 5$，(t_2, w_2) は $\min(2 + 2) = 4$ とコストを計算する．これらのコストをタスク割当てグラフに用い，最小コストフロー問題を実行する．

4.4 割当て時刻計算

節で求めた割当て候補三組み集合 $\mathcal{CT}$ と節で求めた最小コストペア集合 $\mathcal{P}$ を入力とし，最小コストのワーカとタスクと時刻の三組み集合 $\mathcal{T}$ を出力する方法を手順に沿って述べる．

まず，最小コストペア集合と一致する要素をもつ割当て候補三組み集合を抽出し，割当て時刻を計算する．この際の割当て時刻は式 (6) にて最小値を得た時の時刻を求めることで計算する．なお，最小値を得た時刻が複数存在した場合は，最短時刻を出力することとすることで一意に計算することが出来る．最後に，最小コストペアと割当て時刻を合わせることで最小コスト三組みを出力する．

提案手法全体の擬似コードを Algorithm 1 に示す．

Algorithm 1 タスク割当て

Require: ワーカ集合 $W(\varphi)$, タスク集合 $T(\phi_k)$, 時刻集合 φ.
Ensure: 最小コスト割当て結果 $\mathcal{T}$.
1: // (1) 最大タスク割当て計算
2: **for** each $w_j(\phi_k)$ **in** $W(\varphi)$ **do**
3: **for** each t_i **in** $T(\phi_k)$ **do**
4: **if** t_i can be done by $w_j(\phi_k)$ **then**
5: candidate triple $\mathcal{CT} \leftarrow (t_i, w_j, \phi_k)$
6: **end if**
7: **end for**
8: **end for**
9: $f_{max} \leftarrow MaxFlow(\mathcal{CT})$
10: // (2) 最小コスト割当て計算
11: **for** each (t_i, w_j) **in** $\mathcal{CT}$ **do**
12: $c_{min} \leftarrow ComputeLeastCost(t_i, w_j, \varphi)$
13: cost triple $\mathcal{MT} \leftarrow (t_i, w_j, c_{min})$
14: **end for**
15: minimum cost pair $\mathcal{P} \leftarrow MinCostFlow(\mathcal{MT}, f_{max})$
16: // (3) 割当て時刻計算
17: **for** each (t_i, w_j) **in** $\mathcal{P}$ **do**
18: $\phi_{opt} \leftarrow ComputeTime(\mathcal{CT})$
19: min. cost triple $\mathcal{T} \leftarrow (t_i, w_j, \phi_{opt})$
20: **end for**

5. 実験

本章では，1 章で述べた (1) 出来るだけ多くのタスクを完了させること，(2) 出来るだけ早くタスクを完了させることという 2 つの要件を提案手法が満たしているかを検証するための実験について述べる．そこで，提案手法の有効性は (1) 締切時刻までの完了タスク数と (2) 平均タスク所要時間の 2 つの観点から検証する．

5.1 データセット

実際の人の移動履歴を反映したデータとして商用チェックインサービスである Gowalla のログデータ [1](Gowalla データ) を利用した．このデータはユーザ ID 及び，チェックインの日時や場所 ID，緯度経度情報を含む．本実験では，カリフォルニア州を含む矩形内の緯度経度を持つデータのうち，2010 年 4 月 1 日から 100 日間分のデータを用いた．そして，チェックインユーザをワーカとみなしたシミュレーション実験を行った．

また，ワーカの行動が現実的なものと一致した実験を行うため，クラウドソーシングサービスに関するアンケートを行い，その結果をワーカのパラメータに利用した．アンケートは 2014 年 10 月にアンケート調査会社を通じて web 経由で行った．被験者は 10 代から 60 代までのスマートフォンユーザで，大学生 (100 人)，非フルタイム勤務労働者 (200 人)，フルタイム勤務労働者 (200 人) の合計 500 人である．携帯端末で現地調査を行うクラウドソーシングサービスについて説明し，サービスを使ってみたいと回答した 366 人に対し，代表的な平日における仕事可能時間 (開始時間および終了時間) と仕事に利用する移動手段を質問した．ここでのパラメータ設定の詳細は節を参照にされたい．

5.2 実験設定

ここでは，タスクとワーカの設定方法について述べる．

タスク発生方法

本実験ではユーザが過去にチェックインした場所はクラウドソーシングでのタスクに指定される場所となりうると考え，タスクの発生場所をチェックイン場所からランダムに選択した．このタスクの発生方法は既存手法でも用いられているものである [5] [18][12]．より具体的には 2010 月 4 月 1 日から 100 日分のチェックイン場所よりランダムに選択することで発生させた．また，タスクの発生パターンは (1) タスクが静的に発生する場合と (2) 動的に発生する場合の 2 通りを試した．

[1]http://snap.stanford.edu/data/loc-gowalla.html より入手

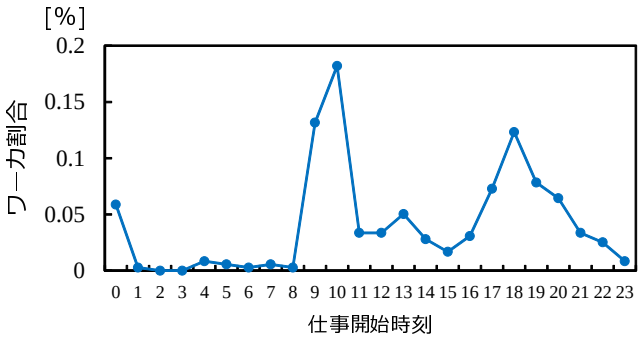

図 6: アンケートから得られたタスク開始時刻のワーカ割合

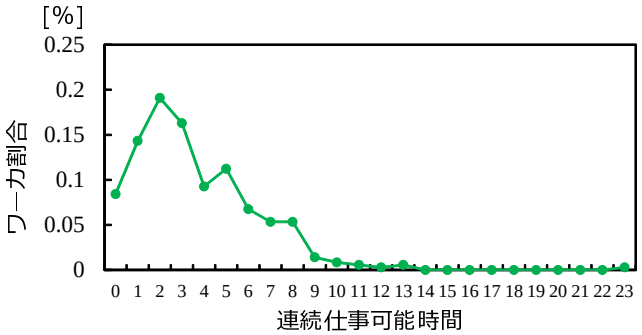

図 7: アンケートから得られた連続仕事可能時間のワーカ割合 (横軸の単位：時間)

静的なタスク発生方法としては，1 日分のタスクが予め決まっている場合を想定し，2010 年 4 月 1 日の 0 時にすべてのタスクを発生させ，締切時刻を 4 月 2 日 0 時とする方法である．また，タスク数は $[100, 200, 300, 400, 500]$ を試し，デフォルト値を 300 に設定した．

動的なタスク発生方法としては，1 日の中でタスクが逐次的に発生する場合を想定し，2010 年 4 月 1 日の 0 時から 10 分おきに決められた数のタスクを発生させ，それぞれの締切時刻をランダムに $[3, 4, 5, 6]$ 時間から選択した．また，10 分おきに発生させるタスク数は $[1, 2, 3, 4, 5]$ を試し，デフォルト値を 3 に設定した．

ワーカのパラメータ設定

今回はワーカのパラメータ設定において，Gowalla データからワーカの場所情報を，web アンケートの結果からワーカのパラメータをソースとして利用した．ある時刻 ϕ_k におけるワーカ集合 $W(\phi_k)$ は，アンケートで回答された仕事時間に関する分布に合うように Gowalla データのチェックインユーザをランダムサンプリングすることで発生させた．ワーカの出現する時刻は，アンケートで回答されたタスク開始時刻の分布 (図 6) を利用した．これによると，8 時以降に仕事を開始するワーカが急激に増え，10 時にピークを迎えたあと再び 18 時にピークを持つような分布をしている．また，ワーカが出現してから消滅するまでの時間は，アンケートで回答された連続仕事可能時間の分布 (図 7) を利用した．これによると，ほとんどが 0 時間から 9 時間までの時間を回答しており，特に 2 時間から 3 時間がピークを持つような分布をしている．

ワーカの最大許容範囲 R_j はワーカのスケジュールから参照した残り仕事可能時間で移動することが出来る最大範囲を設定した．以下にその計算手順を示す．まず，あるワーカの移動速度を設定するために移動手段を決定する．ワーカの移動手段は，アンケートで回答された移動手段別の回答者割合の分布 (2) に合うようにランダムサンプリングすることで設定した．次に移動手段に対応する移動速度 v を用いることで決定する．次にワーカのスケジュールより残り仕事可能時間 ϕ_{rest} を参照する．ワーカは直線で移動するという前提を置けば，w_j の最大許容範囲 R_j の半径は $(v \cdot \phi_{rest})/2$ で計算出来る．なお，ワーカ数は $[100, 200, 300, 400, 500]$ を試し，デフォルト値を 500 に設定した．

表 2: アンケートから得られた移動手段別分布

	自動車	電車	自転車	徒歩
割合 [%]	44.5	37.2	10.3	7.9
速度 [km/h]	19.3	28.5	15	4.8

表 3: 平均タスク所要時間 (min)

	静的タスク	動的タスク
Kazemi	19.2	25.8
CTimeOpt	18.3	23.8

5.3 実験結果

本章では，提案手法を評価した実験結果について述べる．用いた比較手法は以下の通りである．

- Kazemi: 既存のタスク割当て手法 [5]．1 時刻ずつ順番に最小コスト割当て問題を解く (ベースライン)．
- CTimeOpt: ワーカのスケジュールを参照し，時刻間で最大タスク割当て問題と最小コスト割当て問題を解く (提案手法)．
- FTimeOpt: CTimeOpt において最大タスク割当て問題のみを解く．

完了タスク数

はじめに，締切時刻までの完了タスク数の結果を示す．図 8 に静的タスクの割当て開始時刻から締切時刻までの累計完了タスク数を示す．CTimeOpt と Kazemi を比較すると，9 時台まではほとんど変わらないが，10 時ごろから CTimeOpt の完了タスク数が多くなっている．静的タスクの締切時間での完了タスク率は，CTimeOpt が 99.0%，Kazemi が 92.3%であり，提案手法によって約 6.7%のタスク率の向上が実現できた．　これは，CTimeOpt は，最大許容範囲が大きいワーカを有効に割当てることができたためと考えられる．また，CTimeOpt と FTimeOpt を比較すると，締切時刻までの完了タスク数は等しいものの，CTimeOpt の方が立ち上がりが早くなっていることが分かった．これは，最大フロー問題を解いたあとに，完了時刻をコストとした最小コストフロー問題を解くことによって，出来るだけ早くタスクを完了させる効果が得られたと考えられる．

図 9 に締切時刻である 24 時の静的タスク完了状況を地図に表示した結果を示す．CTimeOpt は Kazemi と比較して，タスク密度の低い中心地から外れた場所での割当てに成功していることが分かる．CTimeOpt では，ワーカが仕事が出来る時間帯を事前に把握することで，移動範囲が広いワーカの出現を待って優先的にタスクを割当てることが出来たためと考えられる．一方 Kazemi では，各時刻でのタスク完了数を最大化するため，移動範囲が広いワーカであっても近くのタスクに割当ててしまい，結果として中心地から外れた場所でのタスク完了率が低くなったものと考えられる．

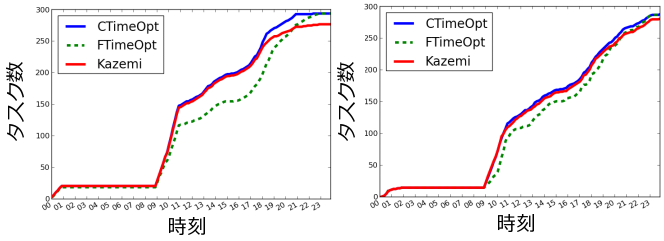

図 8: タスクの累計完了タスク数

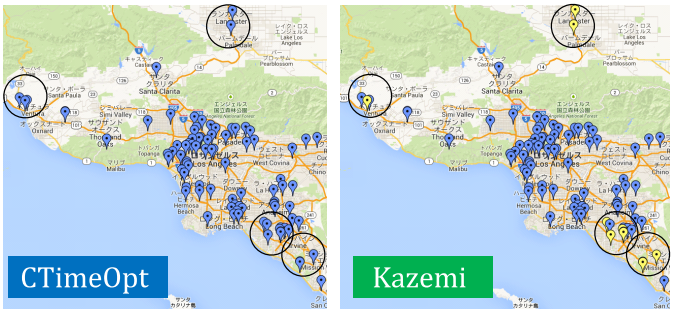

図 9: タスク完了状況 (青：完了，黄色：未完了)

平均タスク所要時間

提案手法でワーカの将来行動を把握したタスク割当てを行ったことによってどれだけ平均タスク所要時間に差が出たのかを Kazemi と CTimeOpt の 2 つを比較することで分析する．なお，Kazemi と CTimeOpt の締切時刻における割当てタスク数はそれぞれ異なるため，ここでは両者ともに割当てたタスクの積集合を抽出し，その中での平均タスク所要時間を比較した．表 3 に 2 手法の平均タスク所要時間を示す．静的タスクでは約 5%，動的タスクでは約 8%ほど平均タスク所要時間を削減するという結果が得られた．これは，コストをタスク完了時刻とした最小コスト割当て問題を解くことによって，タスク所要時間が短くなるワーカを選択して割当てることができたためと考えられる．

パラメータの変化による結果の差

図 10 に静的なタスク発生方法の場合の各パラメータによるタスク完了率の変化を示す．左図はワーカ数を 500 に固定してタスク数を変化させた結果を，右図はタスク数を 300 に固定してワーカ数を変化させた結果を示している．ワーカ数がタスク数よりも大きい場合に特に CTimeOpt のタスク率が高いという結果が得られた．CTimeOpt は後で出現するワーカを有効に利用することが出来るため，あるタスクを実行可能なワーカが複数存在した場合において特に有効であったためと考えられる．また，動的なタスク発生方法の場合も同様の結果が得られた (図 11)．

6. おわりに

本研究は，動的に変化する時空間情報を出来るだけ早く提供することで，ユーザの行動決定に関わる利便性を向上させることを目的とし，クラウドソーシングを用いた動的な時空間情報を収集するシステムを提案した．提案システム実現のため，データ収集の短納期化と網羅性向上の 2 つを目指し，タスクをワーカに割当てる手法に取り組み，ワーカの将来行動を利用したタスク割当て

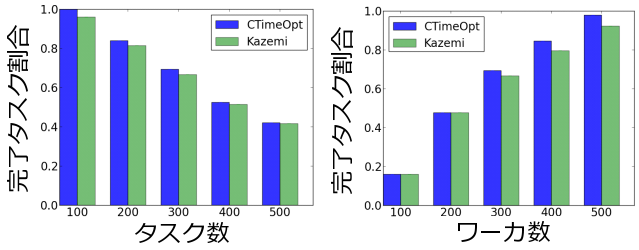

図 10: パラメータによる静的タスク完了率の変化

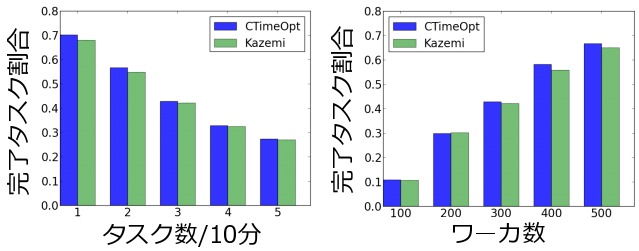

図 11: パラメータによる動的タスク完了率の変化

を提案した．商用チェックインデータを利用した評価実験の結果，提案手法は従来手法と比較して割当てタスク数を約 7%増加させ，平均タスク所要時間が約 8%削減させることを確認した．

今回の実験では問題設定を簡単にするため，ワーカが 1 日に働く回数を 1 回に限定し，1 つのタスクに割当てるワーカ数を一人に限定した．今後は，これらを複数に設定した際の実験を行い，さらなる知見を深めていく予定である．また，ワーカがタスクを断る可能性やワーカの出現確率を考慮し，より現実に即した形での手法の拡張を進める予定である．

[文献]

[1] 経済産業省, “地理空間情報サービス産業の将来ビジョンに関する報告書,” 2008.

[2] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: Real-time event detection by social sensors,” in *Proc. WWW'10*, 2010, pp. 851–860.

[3] R. Lee and K. Sumiya, “Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection,” in *Proc. LBSN'10*, 2010, pp. 1–10.

[4] J. Howe, “The rise of crowdsourcing,” *Wired*, 2006.

[5] L. Kazemi and C. Shahabi, “Geocrowd: Enabling query answering with spatial crowdsourcing,” in *Proc. SIGSPATIAL'12*, 2012, pp. 189–198.

[6] H. Dang, T. Nguyen, and H. To, “Maximum complex task assignment: Towards tasks correlation in spatial crowdsourcing,” in *Proc. iiWAS'13*, 2013, p. 77.

[7] D. Namiot, “Geofence services,” *International Journal of Open Information Technologies*, vol. 1, no. 9, pp. 30–33, 2013.

[8] M. C. Yuen, I. King, and K. S. Leung, “Taskrec: A task recommendation framework in crowdsourcing systems,” *Neural Processing Letters*, pp. 1–16, 2014.

[9] P. Donmez, J. G. Carbonell, and J. Schneider, "Efficiently learning the accuracy of labeling sources for selective sampling," in *Proc. KDD'09*, 2009, pp. 259–268.

[10] C.-J. H. and J. W. V., "Online task assignment in crowdsourcing markets," in *Proc. AAAI'12*, 2012, pp. 45–51.

[11] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing*, 2010, pp. 138–155.

[12] L. Kazemi, C. Shahabi, and L. Chen, "Geotrucrowd: Trustworthy query answering with spatial crowdsourcing," in *Proc. SIGSPATIAL'13*, 2013, pp. 304–313.

[13] C. Chen, S. F. Cheng, A. Gunawan, A. Misra, K. Dasgupta, and D. Chander, "Traccs: Trajectory-aware coordinated urban crowd-sourcing," in *Proc. HCOMP'14*, 2014, pp. 30–40.

[14] D. Ashbrook and T. Starner, "Using GPS to learn significant locations and predict movement across multiple users," *Personal and Ubiquitous Computing*, vol. 7, no. 5, pp. 275–286, 2003.

[15] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "Mining user mobility features for next place prediction in location-based services." in *Proc. ICDM*, 2012, pp. 1038–1043.

[16] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[17] Y. Dinitz, "Dinitz ' algorithm: The original version and even ' s version," in *Theoretical Computer Science*. Springer, 2006, pp. 218–240.

[18] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *Proc. SIGSPATIAL'13*, 2013, pp. 314–323.

羽田野真由美 Mayumi HADANO
2011 年筑波大学生命環境学群生物資源学類卒業．2013 年東京大学新領域創成科学研究科社会文化環境学専攻修士課程修了．同年，日本電信電話株式会社入社．サービスエボリューション研究所勤務．データマイニング，クラウドソーシングの研究に従事．地理情報システム学会会員．

中辻真 Makoto NAKATSUJI
NTT レゾナント株式会社勤務．2010 京都大学大学院情報学研究科博士後期課程修了．電子情報通信学会会員．人工知能学会会員．

戸田浩之 Hiroyuki TODA
日本電信電話株式会社 NTT サービスエボリューション研究所所属．1997 年名古屋大学工学部材料プロセス工学科卒業．1999 年同大大学院工学研究科材料プロセス工学専攻博士課程前期課程修了．同年日本電信電話株式会社に入社．以来，情報検索，データマイニングの研究開発に従事．2007 年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士後期課程修了．博士 (工学)．ACM，情報処理学会，電子情報通信学会，各会員．

小池義昌 Yoshimasa KOIKE
日本電信電話株式会社 NTT サービスエボリューション研究所所属．1989 年東北大学大学院材料化学専攻博士前期課程修了後，日本電信電話株式会社に入社．以来，パターン認識の研究，遠隔教育システムのの研究開発，地域情報検索サービスの研究に従事．