D-033

参加型センシングのための空間データベース問合せ処理

Spatial Database Query Processing for Participatory Sensing

趙 セイ † 杉浦 健人 † 姜 仁河 † 佐々木 勇和 ‡ 石川 佳治 ‡§
Jing Zhao † Kento Sugiura† Renhe Jiang† Yuya Sasaki‡ Yoshiharu Ishikawa‡§

1. はじめに

近年，多種のセンサ機能を搭載するモバイルデバイスの普及により，ユーザのモバイルデバイスをセンシング機器として用いる参加型センシング（participatory sensing）が注目されている [1]．参加型センシングは一種のクラウドソーシングであり，複数の参加者から得られるデータに基づき個人あるいはコミュニティに役立つ知識の発見を目指している．一般的なシステムフレームワークでは，サーバがタスクを近接する参加者に割り当て，参加者から観測データ・収集データなどを受け取る．

参加型センシングには，タスクを割り当てる際の重要な要素としてデータの質とコストがある．データの質は割り当てられた参加者の専門知識や興味などによって決まる．例としてレストランの評価を考える．参加者の好みに偏りがあると，レビュー結果にもそれが影響し，結果が十分信頼できないものとなる．信頼性を上げるには，複数の多様な評価者に評価してもらうことが有効であると考えられる．一方，コストは参加者がタスクを完成するための移動コストである．移動コストが高いほど，タスクの完成時間も長くなり，参加者の意欲も低くなる．

そのため，参加者の割り当てにおいては，多様性を持つ，移動コストも小さい参加者グループを選ぶ必要がある．そこで，本研究では図 1 のような参加型センシングにおける問合せ処理フレームワークを提案し，この問題を解決することを目指す．図 1 のフレームワークでは，活動管理者がタスクの要求を含む問合せ (DkNN-問合せ) を参加型センシング (PS) サーバに送信し，PS サーバ側が参加者間の類似性と参加者の位置情報に基づいて，タスクを最適な参加者グループに割り当てる．しかし，参加者の人数が多いとき最適な参加者グループを見つける計算量が大きくなるため，本研究では効率的な問合せ処理アルゴリズムも併せて提案する．

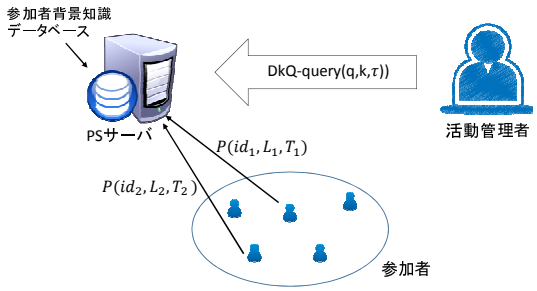

図 1: 参加型センシングにおける問合せ処理フレームワーク

2. 関連研究

参加型センシングにおける既存研究としては，ワーカの履歴情報に基づき，評価スコアを計算し，結果の有効性を満たす一方で，割り当てできるタスク数を最大化することに着目した研究がある [3, 4]．しかし，正確率のみでワーカを評価しているので，ワーカの多様性を考慮してはいない．

一方，クラウドソーシングにおける関連研究では，コストを予算以内に収めつつ，誤り率を最小化する研究がある [2]．ただし，距離など空間コストは考慮されていない．また，ソーシャルネットワークにおける専門グループの形成問題について，特定のスキルを求めるタスクに対し，スキルの要求を満たす一方で通信コストを最小化する手法を提案している [5]．ここでも空間コストは考慮されていない．

そこで，本研究では，参加者の多様性と移動コストの両方を考慮してタスクを割り当てる．

†名古屋大学大学院情報科学研究科
Grad. Sch. of Information Science, Nagoya University
‡名古屋大学未来社会創造機構
Institute of Innovation for Future Society, Nagoya University
§国立情報科学研究所
National Institute of Informatics

3. 問題定義

定義 1: 多様な参加者集合 (Diverse Participants Group)

与えられた参加者集合 X において，任意の二人の参加者間の類似度が指定された閾値 τ より小さいとき，多様な参加者集合と呼ぶ．

定義 2: 多様な k 近傍問合せ (Diverse kNN Query)

問合せ点 q, 割り当てる参加者の人数 k, 閾値 τ が与えられた時，多様な k 近傍問合せは以下の条件を満たす集合 X を返す，なお，$|X| = k$ である．

- **多様性制約**: 任意の二つのオブジェクト間の類似度が閾値 τ より小さい:$\forall p_i, p_j \in R, p_i \neq p_j, sim(p_i, p_j) < \tau$.
- **コスト制約**: オブジェクト集合 R のコストが最小化される: $\min \sum_{p_i \in P} dist(q, p_i)$.

ただし, q はタスクの位置を示す問合せ点であり，P は参加者全体の集合，R は選ばれた参加者の集合である．

4. 問合せ処理アルゴリズム

基本的なアイデアは，問合せ点 q に近い参加者を順に，深さ優先の探索を行うことである．基本となるアルゴリズムを以下に示す．

関数 GetBestAssignment は，最適な k 人の割り当てが見つかればそれを返し，見つからなければ空集合（$\emptyset$）を返す．この関数は引数として参加者集合 P，問合せ点 q，類似度の閾値 τ，割り当てる参加者の人数 k を受け取る．関数中では大域変数として min_cost, R が用いられる．これらは手続き FindAssignment の中でも参照・変更される．手続き FindAssignment は，割り当て可能な参加者の集合 RP（remaining participants），現在の部分的な割り当て PA（partial assignment），PA のコストを引数としてもらい，候補の探索を行う．

ポイントは 4,20 行目で残りの参加者集合について繰返しを行うところである．ここでの処理は $dist(q, p)$ が小さい順に行う．つまり，q の最近傍から処理していく．さらに，候補 PA のコストの下限値は $(k - |PA|) \times dist(q, p) + pcost$ という式で計算する．すなわち，候補 PA のコストに加え，残り $k - |PA|$ 人の参加者のコストである．参加者 p 以降の参加者のコストは，最も小さい場合でも p のコストであるという考えに基づいている．候補 PA のコストは，少なくともこの下限値より高い．そこで，この下限値に基づいて，枝刈りを行う．なお，枝刈りの上限値 min_cost がアルゴリズムの実行中に動的に変化する点に注意する．一方，$dist(q, p) < dist(q, p')$ という制約

は，次に追加する候補者 p' が現在着目している候補者 p よりも遠いことを条件としている．これを入れないと，同じ解を複数回検討してしまうことがある．

Algorithm 1 GetBestAssignment

```
1: function GetBestAssignment(P, q, τ, k)
2:     min_cost ← ∞;                          ▷ 大域変数：最小コスト
3:     RA ← ∅;                                ▷ 大域変数：最終的な割り当て
4:     foreach p ∈ P do                       ▷ dist(q,p) が小さい順に処理
5:         if k × dist(q,p) ≥ min_cost then
6:             break;                         ▷ 見込みがないので終了
7:         end if
8:         FindAssignment(P \ {p}, ∅, p, 0);
9:     end for
10:    return RA;
11: end function
12: procedure FindAssignment(RP, PA, p, pcost)
13:    if PA = ∅ or ∀a ∈ PA, sim(a,p) < τ then   ▷ p は PA の誰とも似てない
14:        PA ← PA ∪ {p};                     ▷ PA に p を追加
15:        pcost ← pcost + dist(q,p);         ▷ コストを更新
16:        if |PA| = k then                   ▷ PA は割り当ての候補
17:            if pcost < min_cost then
18:                min_cost ← pcost;
19:                RA ← PA;                   ▷ 結果を更新
20:            end if
21:        else                               ▷ 探索を続ける
22:            foreach p' ∈ RP such that dist(q,p) < dist(q,p') do   ▷ dist(q,p') が小さい順に処理
23:                if (k − |PA|) × dist(q,p') ≥ min_cost − pcost then
24:                    break;
25:                end if
26:                FindAssignment(RP \ {p'}, PA, p', pcost);
27:            end for
28:        end if
29:    end if
30: end procedure
```

5. 実行例

提案するアルゴリズムを具体的な実行例を用いて説明する．図 2 はアルゴリズムの実行例である．以下では，実行の流れの詳細について述べる．

まずは，最近傍の参加者 p_1 を対象として，5 行目の枝刈り条件をチェックし，満たさない場合，13 行目の多様性の条件をチェックする．それが成立すれば 14,15 行目を実行し，成立しなければ 4 行目に戻し，次の参加者を処理する．その結果，p_1 を PA に入れる．

その次に，割り当てが k 名であるかをチェックし（16 行目），k 名に満たない場合は 21 行目でさらに探索を再帰的に続ける．その結果，p_2 を PA に入れる（PA が $\{p_1, p_2\}$ になる）．残りの参加者集合 $\{p_3, p_4, p_5\}$ において，13 行目の条件を満たす要素はない (つまり，繰返しが終わるまで多様性を満たす k 人を見つけられない) ので，今回の手続きを終了する．

次に，PA が $\{p_1, p_3\}$ になり，k 名に満たないので，p_4 を PA に入れる．この場合，k 名の制約を満たすので，コストが過去のコストより良ければ結果を更新する（17〜19 行目）．その結果，min_cost が 14 に更新される．候補 $\{p_1, p_4\}$ も多様性の条件を満たすが，下限値は 15 である (min_cost より高い) ので，枝刈りする．

続いて，PA は $\{p_2\}$ になり，PA の下限値 (15) も min_cost より高いので，枝刈りを行う．この場合，アルゴリズムを終了し，今までの最適案 $\{p_1, p_3, p_4\}$ を結果として返す．

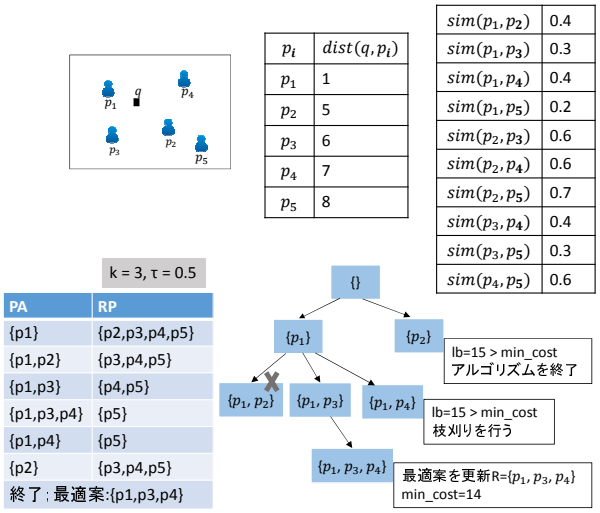

図 2: アルゴリズムの実行例

6. まとめと今後の課題

本稿では，参加型センシングのための問合せ処理に着目し，深さ優先探索を用いた効率的なアルゴリズムを提案した．提案手法では，最近傍の参加者を順次検索しながら処理を進めるため，R 木などの空間索引による実装は容易であると考えられる．

今後の課題としては，次のようなものがある．1) 問合せ点が同時に複数与えられたとき，それらに対する最適な割り当てを求める問合せへの対応，2) 他のコスト評価法への対応，3) 新たな多様性や割り当ての質の定義への対応，4) 実験に基づく評価などである．

[謝辞]

本研究の一部は，科学研究費（25280039）の研究助成によるものである．

参考文献

[1] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory Sensing. In *First Workshop on World-Sensor-Web (WSW'06) (ACM Sensys Workshop)*, pages 117–134, 2006.

[2] C. C. Cao, J. She, Y. Tong, and L. Chen. Whom to Ask? Jury Selection for Decision Making Tasks on Micro-blog Services. *PVLDB*, 5(11):1495–1506, 2012.

[3] L. Kazemi and C.Shahabi. GeoCrowd: Enabling Query Answering with Spatial Crowdsourcing. In *ACM SIGSPATIAL GIS*, pages 189–198, 2012.

[4] L. Kazemi, C. Shahabi, and L. Chen. Geotrucrowd: Trustworthy Query Answering with Spatial Crowdsourcing. In *ACM SIGSPATIAL GIS*, pages 304–313, 2013.

[5] T. Lappas, K. Liu, and E. Terzi. Finding a Team of Experts in Social Networks. In *KDD*, pages 467–476, 2009.