Exact and Efficient Inference for Collective Flow Diffusion Model via Minimum Convex Cost Flow Algorithm

Yasunori Akagi,[1] Takuya Nishimura,[1] Yusuke Tanaka,[1] Takeshi Kurashima,[1] Hiroyuki Toda[1]
[1]NTT Service Evolution Laboratories, NTT Corporation,
1-1 Hikari-no-oka, Yokosuka-Shi, Kanagawa, 239-0847, Japan
{yasunori.akagi.cu, takuya.nishimura.fk, yusuke.tanaka.rh, takeshi.kurashima.uf, hiroyuki.toda.xb}@hco.ntt.co.jp

Abstract

Collective Flow Diffusion Model (CFDM) is a general framework to find the hidden movements underlying aggregated population data. The key procedure in CFDM analysis is MAP inference of hidden variables. Unfortunately, existing approaches fail to offer exact MAP inferences, only approximate versions, and take a lot of computation time when applied to large scale problems. In this paper, we propose an exact and efficient method for MAP inference in CFDM. Our key idea is formulating the MAP inference problem as a combinatorial optimization problem called Minimum Convex Cost Flow Problem (C-MCFP) with no approximation or continuous relaxation. On the basis of this formulation, we propose an efficient inference method that employs the C-MCFP algorithm as a subroutine. Our experiments on synthetic and real datasets show that the proposed method is effective both in single MAP inference and people flow estimation with EM algorithm.

1. Introduction

With recent advances in GPS, Wi-Fi, and various sensors, the importance of location information has grown and is being utilized in various fields. However, it is often difficult to obtain data about individual movements because of privacy concerns or the difficulty of tracking individuals over time. Instead, aggregated count data is relatively easy to obtain as it does not include individual movement information. For example, mobile spatial statistics (Terada, Nagata, and Kobayashi 2013), which is the hourly population data of fixed size square grids calculated from mobile network operational data, are available for purchase in Japan. As another example, traffic data is often obtained not in the form of tracking data of individual cars, but in the form of count data acquired by cameras or sensors installed on road networks (Yang and Zhou 1998; Morimura, Osogami, and Idé 2013).

Although there are various uses for these aggregated count data, their applicability is limited because they do not contain explicit information about people movements. In order to utilize such data, Collective Graphical Model (CGM)(Sheldon and Dietterich 2011), which enables us to conduct learning and inference with aggregated count data, was proposed. In particular, Collective Flow Diffusion Model (CFDM) (Kumar, Sheldon, and Srivastava 2013), which is a special case of CGM, has been proposed to infer people flows between the areas by modeling individual movements via a Markov chain approach; it has been applied to the analysis of the hidden movements behind observed count data in a traffic network (Kumar, Sheldon, and Srivastava 2013), urban space (Iwata et al. 2017; Akagi et al. 2018; Iwata and Shimizu 2019), amusement park (Du, Kumar, and Varakantham 2014) and exhibition halls (Tanaka et al. 2018).

An important function in CFDM analysis is MAP (maximum a posteriori) inference of the number of moving people from observed population data and parameters of the probabilistic model. This process is mainly used in two ways: (i) As a method for recovering people flow given observed population data and a human mobility model. Even if we can design a probabilistic model of human mobility using domain knowledge or estimate the model using another small set of movement (trajectory) data, we have to conduct MAP inference in order to know the number of people moving between areas. (ii) As a method for conducting E-step in the EM (Expectation Maximization) algorithm to estimate people flow and parameters of the probabilistic model simultaneously. Although E-step was implemented by the well-designed MCMC (Sheldon and Dietterich 2011) in the original CFDM proposal, its scalability was problematic. In order to address this issue, a method that uses MAP inference as an alternative to the regular expectation operation was widely used in subsequent research (Iwata et al. 2017; Akagi et al. 2018; Tanaka et al. 2018).

Although methods for realizing MAP inference for CFDM are very important, previous proposals have several crucial drawbacks. (i) They do not provide exact MAP inference because they use continuous relaxation and Stirling's approximation. (ii) Each optimum solution element is non-integer because of continuous relaxation. As a result, the optimum solutions are dense with many non-zero elements and each solution occupies a lot of memory. (iii) When we deal with large scale problems, a lot of computation time is still needed to solve the optimization problem.

In this paper, we propose a novel method for MAP inference in CFDM that overcomes the shortcomings of previous approaches. Our key idea is formulating the MAP inference problem in CFDM as a combinatorial optimization problem called (non-linear) Minimum Cost Flow Problem (MCFP). Moreover, we prove that all cost functions of the MCFP are "discrete convex functions", discrete analogues of the continuous convex function. This fact indicates that the formulated MCFP is a Minimum Convex Cost Flow Problem (C-MCFP) variant, which is an efficiently solvable subclass of MCFP. On the basis of this formulation, we propose an efficient inference method that employs the C-MCFP algorithm as a subroutine. The proposal has the following advantages:

1. It offers exact MAP inference as no approximation is used.
2. Optimum solution elements are integers, which is consistent with the number of moving people. Moreover, the solution tends to be sparse and we can hold it with less memory by use of the sparse matrix data structure.
3. By utilizing efficient algorithms for C-MCFP, fast estimation is possible. In addition, it is easy to use in practice because it is not necessary to set hyperparameters, and the calculation time is relatively insensitive to the probabilistic models and the optimum solutions.

Our results are significant in that they bridge two distinct research topics, graph algorithms and CFDM inference. This work is the first to regard CFDM inference as a discrete optimization problem on a graph (all efficient existing methods transform the inference problem into a continuous optimization problem via approximation). Our non-trivial finding of the discrete convexity of the cost function is an important key in revealing the hidden relationship between graph algorithms and inference in collective flow diffusion.

Experiments on synthetic and real datasets show that the proposed method is effective for MAP inference in terms of both running time and solution quality such as sparsity. Of particular note, running time is accelerated 10 times or more and sparsity of optimum solution is dramatically increased in most cases. Moreover, we use the proposal to conduct people flow estimation via the EM algorithm and confirm its effectiveness.

2. Problem Setting

For positive integer k, we denote $[k] := \{1, \dots, k\}$. Suppose that the target space is discretized into n distinct areas. The people who were in area $i \in [n]$ at timestep t will stay in i or move to another area to be observed in area $j \in \Gamma_i$ at timestep $t+1$, where $\Gamma_i \subseteq [n]$ is the set of areas that can be moved to from area i. This process will be repeated for each $t \in [T-1]$, where T is the total number of timesteps.

The problem we address in this paper is formulated as follows. Suppose we are given the population of area i at timestep t, $N_{t,i} (i \in [n], t \in [T])$. Our goal is to estimate the number of people who leave area i at time t and whose next area is j at time $t+1$, $M_{tij} (i \in [n], j \in [n], t \in [T-1])$. Figure 1 shows an example of this problem setting.

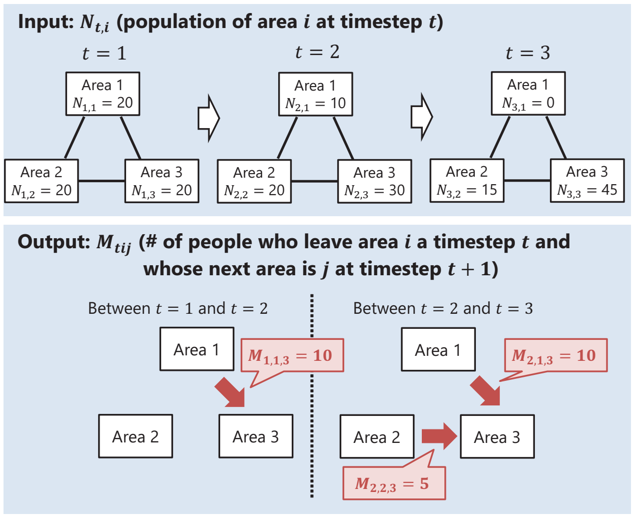

Figure 1: An example of the problem setting where the number of areas $n = 3$ and the number of total timesteps $T = 3$.

3. Background

3.1 Collective Flow Diffusion Model (CFDM)

Let $\boldsymbol{\theta}_i = \{\theta_{ij}\}_{j\in\Gamma_i}$ $(\sum_{j\in\Gamma_i} \theta_{ij} = 1)$ be the transition probability from area i to other areas (including i itself). We here assume $\boldsymbol{\theta}_i$ does not depend on timestep t. Given population $N_{t,i}$ and transition probability $\boldsymbol{\theta}_i$, the transition population $\boldsymbol{M}_{ti} = \{M_{tij}\}_{j\in\Gamma_i}$ $(t \in [T-1], i \in [n])$ is assumed to be decided by the following multinomial distribution: $\boldsymbol{M}_{ti} \sim \mathrm{Multi}(N_{t,i}, \boldsymbol{\theta}_i)$. Given $\boldsymbol{N} = \{N_{t,i} \mid t \in [T], i \in [n]\}$ and $\boldsymbol{M} = \{\boldsymbol{M}_{ti} \mid t \in [T-1], i \in [n]\}$, the likelihood function of $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i \mid i \in [n]\}$ is given by

$$P(\boldsymbol{M} \mid \boldsymbol{N}, \boldsymbol{\theta}) \propto \prod_{t=1}^{T-1} \prod_{i\in[n]} \left(\frac{N_{t,i}!}{\prod_{j\in\Gamma_i} M_{tij}!} \prod_{j\in\Gamma_i} \theta_{ij}^{M_{tij}} \right). \tag{1}$$

In addition, the population in each area, $N_{t,i}$, and the transition population between areas, $\boldsymbol{M}_{ti}$, satisfy the following two relationships $N_{t,i} = \sum_{j\in\Gamma_i} M_{tij}, N_{t+1,i} = \sum_{j\in\Gamma_i} M_{tji}$ $(t \in [T-1], i \in [n])$, which represent the law of conservation in the number of people.

Our purpose is to estimate the true number of people moving between areas. We consider two problems: (i) estimation of $\boldsymbol{M}$ given $\boldsymbol{N}$ and $\boldsymbol{\theta}$, and (ii) estimation of $\boldsymbol{M}$ and $\boldsymbol{\theta}$ given only $\boldsymbol{N}$. The first problem, includes, for example, the case where it is possible to design a human movement model (i.e. $\boldsymbol{\theta}$) in the target space based on domain knowledge, geographical information, or other data related to people movement such as a small amount of trajectory data. The second problem corresponds to the case that there is no clue as to $\boldsymbol{\theta}$ and it is necessary to estimate everything from $\boldsymbol{N}$.

In any case, an important subroutine in achieving our purpose is solving the following MAP inference problem:

$$
\begin{aligned}
\max_{\boldsymbol{M}}. \quad & P(\boldsymbol{M} \mid \boldsymbol{N}, \boldsymbol{\theta}) \\
\text{s.t.} \quad & N_{t,i} = \sum_{j \in \Gamma_i} M_{tij} \quad (t \in [T-1], i \in [n]), \\
& N_{t+1,i} = \sum_{j \in \Gamma_i} M_{tji} \quad (t \in [T-1], i \in [n]), \\
& M_{tij} \in \mathbb{Z}_{\geq 0} \quad (t \in [T-1], i \in [n], j \in \Gamma_i).
\end{aligned} \tag{2}
$$

In the first problem, the optimum solution of (2) is the desired answer. A common approach to solving the second problem is to estimate, alternatively, $\boldsymbol{M}$ and $\boldsymbol{\theta}$ by the EM algorithm considering $\boldsymbol{M}$ as a hidden variable and $\boldsymbol{\theta}$ as parameter of a probabilistic model. Since high computational cost is incurred in calculating the expected value of hidden variable $\boldsymbol{M}$ by MCMC, a method to replace the expected value with the solution of the MAP inference problem has already been proposed (Sheldon et al. 2013) and is being widely used to conduct E-step. This approach solves the optimization problem (2) iteratively.

3.2 Minimum Cost Flow Problems

(Non-linear) Minimum Cost Flow Problem (MCFP) is a combinatorial optimization problem defined as follows. Let $G = (V, E)$ be a directed graph, where each node $i \in V$ has supply value $b_i \in \mathbb{Z}$, and each edge $(i, j) \in E$ has capacity $l_{ij} \in \mathbb{Z}_{\geq 0}$ and cost function $c_{ij} : \mathbb{Z}_{\geq 0} \to \mathbb{R}$. If $b_i > 0$ we call node i to be source, and if $b_i < 0$ we call sink. MCFP is the problem of finding a minimum cost flow on G that satisfies the supply constraints at all nodes and capacity constraints at all edges. MCFP can be described as follows:

$$
\begin{aligned}
\min_{\boldsymbol{x} \in \mathbb{Z}^{|E|}}. \quad & \sum_{(i,j) \in E} c_{ij}(x_{ij}) \\
\text{s.t.} \quad & \sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(j,i) \in E} x_{ji} = b_i \quad (i \in V), \\
& 0 \leq x_{ij} \leq l_{ij} \quad ((i,j) \in E).
\end{aligned} \tag{3}
$$

Note that this paper considers the problems that restrict feasible $\boldsymbol{x}$ to integer values i.e. $\boldsymbol{x} \in \mathbb{Z}^{|E|}$. Generally, MCFP (3) is NP-hard and difficult to solve efficiently. However, special cost functions make it possible to derive efficient optimization algorithms. For example, MCFP with linear cost functions, which is the most famous special case of MCFP, is polynomial-time solvable and many efficient algorithms have been developed (Kiraly and Kovacs 2012). Moreover, Minimum Convex Cost Flow Problem (C-MCFP), in which every cost function c_{ij} satisfies "discrete convexity" $c_{ij}(x+1) + c_{ij}(x-1) \geq 2 \cdot c_{ij}(x)$ $(x = 1, 2, \ldots)$, is known to be an efficiently solvable subclass of MCFP (Ahuja, Magnanti, and Orlin 1993).

4. Proposed Method

4.1 Formulation as C-MCFP

We show that the optimization problem (2) can be formulated as C-MCFP. After taking the logarithm of the objective function (1) and omitting terms that do not depend on $\boldsymbol{M}$, the objective function equals $\sum_{t \in [T-1]} \sum_{i \in [n]} \sum_{j \in \Gamma_i} (-\log M_{tij}! + M_{tij} \log \theta_{ij})$. Since we can split (2) into independently solvable $T-1$ subproblems by t, all we have to do is solve the minimization problems described as follows for each $t \in [T-1]$:

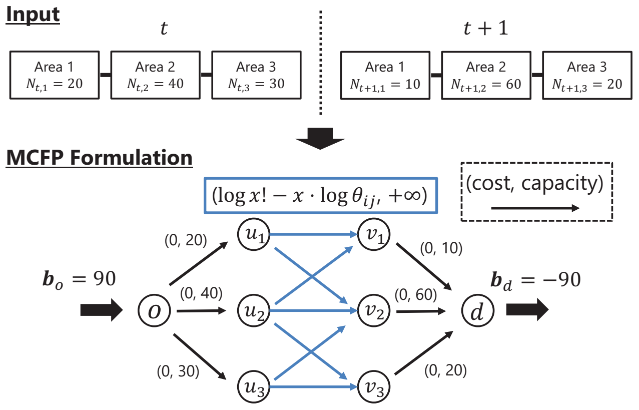

Figure 2: An example of MCFP formulation when the number of areas $n = 3$. o is the source and d is the sink of the flow network. The capacity of edge (o, u_i) equals to $N_{t,i}$ and the capacity of edge (v_i, d) equals to $N_{t+1,i}$.

$$
\begin{aligned}
\min_{\boldsymbol{M}_t}. \quad & \sum_{i \in [n]} \sum_{j \in \Gamma_i} (\log M_{tij}! - M_{tij} \log \theta_{ij}) \\
\text{s.t.} \quad & N_{t,i} = \sum_{j \in \Gamma_i} M_{tij} \quad (i \in [n]), \\
& N_{t+1,i} = \sum_{j \in \Gamma_i} M_{tji} \quad (i \in [n]), \\
& M_{tij} \in \mathbb{Z}_{\geq 0} \quad (i \in [n], j \in \Gamma_i).
\end{aligned} \tag{4}
$$

In order to formulate the problem (4) as MCFP, we construct an instance by the procedure described below (an example is shown in Figure 2):

1. Let $V = \{u_i\}_{i \in [n]} \cup \{v_i\}_{i \in [n]} \cup \{o, d\}$. u_i and v_i correspond to area i at timestep t and timestep $t+1$, respectively. o is the source node and d is the sink node of the flow network.
2. For $i \in [n]$, add edge (o, u_i) with cost function 0 (constant function) and capacity $N_{t,i}$.
3. For $i \in [n]$, add edge (v_i, d) with cost function 0 and capacity $N_{t+1,i}$.
4. For $i \in [n]$ and $j \in \Gamma_i$, add edge (u_i, v_j) with cost function $f_{ij}(x) := \log x! - x \cdot \log \theta_{ij}$ and capacity $+\infty$.
5. Set $b_o = \sum_{i \in [n]} N_{t,i}$, $b_d = -b_o = -\sum_{i \in [n]} N_{t,i}$ and $b_{u_i} = b_{v_i} = 0$ $(i \in [n])$.

For the MCFP instance constructed above, the following holds.

Proposition 1. *For $\boldsymbol{M}_t^*$ defined by $M_{tij}^* = x_{u_i v_j}^*$ $(i \in [n], j \in \Gamma_i)$ where $\boldsymbol{x}^*$ is the optimum solution of the MCFP* *instance constructed above, $\boldsymbol{M}_t^*$ is an optimum solution of the optimization problem* (4).

Proof of Proposition 1. Let $\boldsymbol{x}$ to be a feasible solution of the constructed MCFP. From the non-negativity of x_{ij} and flow conservation constraints at node o and d, $x_{ou_i} = N_{ti}$ and $x_{v_i d} = N_{t+1,i}$ ($\forall i \in [n]$) must be satisfied. From these facts and flow conservation constraints at node u_i and v_i, $N_{t,i} = \sum_{j\in\Gamma_i} x_{u_i v_j}$ and $N_{t+1,i} = \sum_{j\in\Gamma_i} x_{v_j u_i}$ ($\forall i \in [n]$) hold. Since we restrict $\boldsymbol{x}$ to integer values and total MCFP cost is $\sum_{i\in[n]}\sum_{j\in\Gamma_i}(\log x_{u_i v_j}! - x_{u_i v_j}\log\theta_{ij})$, the constructed MCFP is equivalent to the optimization problem (4), so the proposition holds. □

Proposition 2. *For the MCFP instance constructed above, all cost functions satisfy discrete convexity, i.e.* $c_{ij}(x+1) + c_{ij}(x-1) \geq 2\cdot c_{ij}(x)$ $(x = 1, 2, \ldots)$.

Proof of Proposition 2. It is clear that a constant function satisfies discrete convexity, so it is sufficient to check for f_{ij}. We have $f_{ij}(x+1) + f_{ij}(x-1) - 2\cdot f_{ij}(x) = \log(x+1)! + \log(x-1)! - 2\cdot\log x! = \log(x+1) - \log x \geq 0$. This confirms the discrete convexity of f_{ij}. □

Proposition 1 says that by solving MCFP we can get an optimum solution of problem (4). Proposition 2 shows that the constructed MCFP instance belongs to C-MCFP. Since C-MCFP is an efficiently solvable subclass of MCFP as described in 3.2, we can design efficient algorithms to tackle the original MAP inference problem (4).

Note that problem (4) may not have any feasible solution if $\sum_{i\in[n]} N_{t,i} \neq \sum_{i\in[n]} N_{t+1,i}$ holds or $|\Gamma_i|$ ($i \in [n]$) is small. Such cases occur frequently when dealing with noisy real data. Even in such cases, our method with slight modification can output reasonable solutions. We describe this modification in Section 4.3.

4.2 Algorithm

We describe here an algorithm that can find exact optimum solutions of C-MCFP, called Capacity Scaling algorithm (CS) (Minoux 1986). CS is an algorithm that successively augments flow along the shortest path from source to sink in a residual graph, which is an auxiliary graph calculated from the current flow. By maintaining a scalar value, called potential, on each node and modifying edge costs to ensure that they are non-negative, we can utilize Dijkstra's algorithm (Dijkstra 1959), which is a fast algorithm for shortest path search in graphs with non-negative edge costs. In order to reduce the number of shortest path searches, CS is designed to carry sufficiently large number of flows in each path augmentation. The algorithm utilized in our work is the one described in Chapter 14.5 of (Ahuja, Magnanti, and Orlin 1993). Although this algorithm is based on the idea of (Minoux 1986), some changes have been made, so its computation complexity differs from that of (Minoux 1986).

Given a C-MCFP instance with graph $G = (V, E)$, Theorem 14.1 of (Ahuja, Magnanti, and Orlin 1993) claims that CS runs in $O(|E|\cdot\log U\cdot S)$, where $U := \max_{i\in V}|b_i|$ is the maximum absolute value of flow demand and S is the time complexity for solving a shortest path problem in graph G

Algorithm 1 Algorithm for solving MAP inference problem (2) via capacity scaling algorithm

```
Require: Population of each area and time $\boldsymbol{N}$, transition matrix $\boldsymbol{\theta}$
  for all $t \in [T-1]$ do
    Construct C-MCFP instance based on $\boldsymbol{N}_t, \boldsymbol{N}_{t+1}, \boldsymbol{\theta}$ by the procedure described in Section 4.1
    Get optimum solution $\boldsymbol{x}^*$ of constructed C-MCFP by capacity scaling algorithm
    for all $i \in [n]$ do
      for all $j \in \Gamma_i$ do
        $M^*_{tij} \leftarrow x^*_{u_i v_j}$
      end for
    end for
  end for
  return $\boldsymbol{M}^*$
```

with non-negative edge costs. According to Dijkstra's algorithm with binary heap, S is bounded by $O(|E|\cdot\log|V|)$, so the total time complexity is $O(|E|^2\cdot\log|V|\cdot\log U)$. When this algorithm is used to solve problem (4), its time complexity is $O(m^2\cdot\log n\cdot\log F)$, where n is the number of areas, m is the number of edges of the adjacency graph between the areas determined by Γ_i ($i \in [n]$) and $F := \sum_{i\in[n]} N_{t,i}$ is the total population of targeted areas. Note that, the total complexity does not depend on the maximum value of edge capacity, and it is guaranteed that the algorithm runs efficiently even if the graph contains an edge with infinite capacity.

CS is a suitable algorithm for solving our problem in the following sense: When dealing with real-world datasets, sometimes F is extremely large (for example, in mobile spatial statistics in the Greater Tokyo Area, which consists of population distribution data by time and area, F is about 10^6–10^7). Therefore, the algorithm used to solve the formulated C-MCFP should have sub-linear time complexity with respect to F. Accordingly, CS is appropriate since its time complexity is proportional to $\log F$.

The overall algorithm for solving the original MAP inference problem (2) is summarized in Algorithm 1.

4.3 Handling with Infeasible Cases

As mentioned in Section 4.1, when dealing with real-world data, there may not be feasible solution to problem (4). To address this problem and output a reasonable solution, we add a few more steps in the instance construction procedure described in Section 4.1.

First, we add edge (o, d) with linear cost function Cx, where C is a sufficiently large constant, and capacity $+\infty$. Next, we set $b_o = S$, $b_d = -S$, $b_{u_i} = b_{v_i} = 0$ ($i \in [n]$), where $S := \max(\sum_{i\in[n]} N_{t,i}, \sum_{i\in[n]} N_{t+1,i})$. This newly formulated MCFP always has a feasible solution and still belongs to C-MCFP, so we can solve this by CS.

In this case, $\boldsymbol{M}_t^*$ calculated from the optimum solution of the MCFP does not necessarily satisfy the population conservation law $N_{t,i} = \sum_{j\in\Gamma_i} M^*_{tij}, N_{t+1,i} = \sum_{j\in\Gamma_i} M^*_{tji} (i \in [n])$, which are the constraints of the original problem (4). We can interpret these discrepancies as follows: $N_{t,i} - \sum_{j\in\Gamma_i} M^*_{tij}$ is outflow from area i to somewhere outside the targeted areas, and $N_{t+1,i} - \sum_{j\in\Gamma_i} M^*_{tji}$ is inflow from somewhere outside the targeted areas to area i between timesteps t and $t+1$.

5. Experimental results

Here, we use numerical experiments to demonstrate the practical utility of the proposed method. All experiments are conducted on a 64-bit CentOS 7.3 machine with Xeon(R) Gold 6126 CPU(2.60GHz)x2 and 512 GB memory. The capacity scaling algorithm is implemented in C++ (g++ 4.8.5 with the -O3 option); other codes were written in python 2.7.12 with SciPy (Jones et al. 2001).

5.1 Compared methods

We compare the proposed method with commonly used ones used in CFDM inference (Iwata et al. 2017; Akagi et al. 2018; Tanaka et al. 2018). In this method, we solve an optimization problem that has the following objective function $f(\boldsymbol{M}_t) + \frac{\lambda}{2}\cdot g(\boldsymbol{M}_t)$ under constraints $M_{tij} \in \mathbb{R}_{\geq 0}$, where

$$f(\boldsymbol{M}_t) = \sum_{i\in[n],j\in\Gamma_i} \left(M_{tij}\log M_{tij} - M_{tij}(1+\log\theta_{ij})\right),$$

$$g(\boldsymbol{M}_t) = \sum_{i\in[n]}\left[(N_{t,i} - \sum_{j\in\Gamma_i} M_{tij})^2 + (N_{t+1,i} - \sum_{j\in\Gamma_i} M_{tji})^2\right]$$

and λ is a hyperparameter. This problem is derived by applying Stirling's approximation and continuous relaxation to the objective function of (4), and adding constraints of people conservation law to objective function as penalty terms. λ controls the strength of penalty terms. This optimization problem has a convex objective function and bound constraints, so we can get the global optimum by L-BFGS-B method (Byrd et al. 1995), which is implemented in scipy.optimize. Our experiments explored three methods with λ values of $\{1, 10, 100\}$.

5.2 MAP inference: Synthetic data

First, we compare running times and characteristics of the optimum solutions of MAP inference problem (2) obtained by each method using synthetic data. We randomly generate synthetic instances of the MAP inference problem (2). We consider an $L \times L$ grid space, where each cell corresponds to one area. Γ_i is set to be $[n]$ for $\forall i \in [n]$ (i.e. we consider the "fully connected" situation). We set $T = 2$ and $\boldsymbol{N}_{t,i} \sim \mathrm{Multi}(F, \boldsymbol{p}_t)$ $(t = 1, 2)$, where F is the total population in the grid space and $\boldsymbol{p}_1, \boldsymbol{p}_2 \sim \mathrm{Dirichlet}(\mathbf{1})$. $\boldsymbol{\theta}$ is generated in two ways as follows.

1. $\boldsymbol{\theta}_i \sim \mathrm{Dirichlet}(\mathbf{1})$ for each $i \in [n]$ independently. We call this generation procedure "Dirichlet".
2. $\theta_{ij} = \exp(-\mathrm{dist}(i,j))/\sum_{j\in\Gamma_i}\exp(-\mathrm{dist}(i,j))$, where $\mathrm{dist}(i,j)$ is the Euclidean distance between cell i and j. We call this procedure "Exponential decay". This procedure reflects the characteristics typical of movements that people are likely to take over short distances rather than long ones.

To clarify the dependence of computation time on the number of areas, L^2, and total population, F, we solve the MAP inference problem for $L = 10, 20, 30$ fixing F to 10^4, and for $F = 10^4, 10^5, 10^6$ fixing L to 20. We generate 10 random instances for each evaluation.

The average running times (seconds) for 10 instances by each algorithm are summarized in Table 1. Each experiment is executed with the time limit of 1000 seconds. If running time exceeds the time limit, running time of the trial is recorded as 1000 seconds. In such a case, the averaged value is underestimated. To clarify this, we tag average running time in the table with ">" if the time limit is exceeded in even one instance. In the parentheses, standard deviation of running times are shown if all 10 trials are completed in the time limit. L-BFGS-B methods have longer running time than the proposed method and varies with parameter settings and instances. This unstable behavior will be problematic in practical usage. The proposed method outperforms all other methods in all settings. In particular, it offers the advantage that it can solve problems with small computational time and work stably even when L and F are large.

In order to compare the characteristics of optimum solutions output by the proposed method and L-BFGS-B ($\lambda = 1$), we solve two examples with $L = 5, F = 10^2$, "Exponential decay" and $L = 5, F = 10^3$, "Exponential decay" instances and checked the solutions in detail. The results are shown in Figure 3. In this figure, the $L^2 \times L^2$ optimum solution matrix obtained by each method are presented as a heatmap. To investigate the sparsity structure of the solution, the maximum value of heatmap is set to 1 and minimum value to 0. While the solution obtained by L-BFGS-B is blurred and contains a lot of small but non-zero elements (elements with light colors) because of continuous relaxation, proposed method is able to produce sparse solutions. We calculated the sparseness of each solution by (# of near-zero ($< 10^{-4}$) elements)/(# of whole elements); the yielded values are 90%, 67% with proposed method and 0%, 0 % with L-BFGS-B. This implies that the memory needed to hold the solution can be reduced significantly by using sparse matrix structure. Although we can get sparse solutions by rounding the solutions of existing methods, this operation violates the constraint of population conservation and degrades solution quality.

5.3 MAP inference: Real data

We evaluate running times and characteristics of the optimum solutions using real-world spatio-temporal population data. We use mobile spatial statistics (Terada, Nagata, and Kobayashi 2013), which is the hourly population data for fixed size square grids calculated from mobile network operational data. We use Tokyo and Kanagawa prefecture data, which is the main part of the capital region of Japan, on April 1st, 2015 (weekday) and April 5th, 2014 (holiday). $\boldsymbol{N}_t$ is the population of each area at the clock time of t-hour for $t \in \{0, 1, \ldots, 22\}$ on each day. In order to compare the performances of the methods at different cell width, we aggregate population data of each cell and made datasets with cell sizes of 5km × 5km, 2km × 2km, and 1km × 1km. The resulting datasets contain 200, 1017, 3711 cells,

Table 1: The average running time (seconds) of 10 synthetic instances when F is fixed to 10^4 (above) and when L is fixed to 20 (below). The best running time is highlighted for each problem size. Values with ">" are underestimates due to the time limit. Standard deviation is shown in parentheses if all 10 trials are completed in the time limit.

type of $\boldsymbol{\theta}$	Dirichlet			Exponential decay		
L	10	20	30	10	20	30
Proposed	**0.05 (0.00)**	**0.61 (0.01)**	**4.54 (0.16)**	**0.03 (0.00)**	**0.46 (0.03)**	**6.29 (2.60)**
L-BFGS-B ($\lambda = 1$)	6.51 (0.91)	132.86 (15.46)	357.32 (39.76)	13.51 (2.00)	273.25 (18.86)	>911.22 (−)
L-BFGS-B ($\lambda = 10$)	7.40 (1.27)	143.14 (13.25)	387.09 (56.31)	13.87 (1.69)	281.40 (19.18)	>936.14 (−)
L-BFGS-B ($\lambda = 100$)	9.65 (2.01)	169.83 (17.19)	440.77 (69.87)	15.79 (1.36)	297.40 (20.42)	>975.64 (−)

type of $\boldsymbol{\theta}$	Dirichlet			Exponential decay		
F	10^4	10^5	10^6	10^4	10^5	10^6
Proposed	**0.71 (0.09)**	**4.19 (0.85)**	**14.25 (1.56)**	**0.68 (0.22)**	**2.44 (0.58)**	**4.93 (0.94)**
L-BFGS-B ($\lambda = 1$)	140.16 (15.34)	434.25 (114.80)	>804.52 (−)	323.87 (30.86)	>1000.00 (−)	>1000.00 (−)
L-BFGS-B ($\lambda = 10$)	149.29 (14.35)	503.72 (117.16)	>880.68 (−)	340.96 (41.54)	>1000.00 (−)	>1000.00 (−)
L-BFGS-B ($\lambda = 100$)	175.65 (18.26)	793.54 (146.68)	>899.83 (−)	356.24 (48.56)	>1000.00 (−)	>887.22 (−)

Table 2: The average running time (seconds) for real data. The best running time is highlighted for each cell width. Values with ">" are underestimates due to the time limit. Standard deviation is shown in parentheses if all 10 trials are completed in the time limit.

dataset	April 1st, 2015			April 5th, 2015		
cell width	5km	2km	1km	5km	2km	1km
Proposed	**0.84 (0.16)**	**9.16 (1.49)**	**59.40 (22.38)**	**0.41 (0.01)**	**6.52 (1.15)**	**54.00 (10.70)**
L-BFGS-B ($\lambda = 1$)	196.46 (139.61)	>1000.00 (−)	>1000.00 (−)	68.76 (25.43)	>940.84 (−)	>1000.00 (−)
L-BFGS-B ($\lambda = 10$)	14.96 (34.63)	>1000.00 (−)	>1000.00 (−)	10.90 (19.85)	>1000.00 (−)	>1000.00 (−)
L-BFGS-B ($\lambda = 100$)	2.04 (0.73)	>811.94 (−)	>1000.00 (−)	0.99 (0.89)	>697.78 (−)	>1000.00 (−)

respectively. We construct $\boldsymbol{\theta}$ by the same procedure as "Exponential decay" in the synthetic data experiment and set $\Gamma_i = \{j \mid j \in [n], \mathrm{dist}(i, j) \leq 5\}$, where $\mathrm{dist}(i, j)$ is Euclidean distance between cell i and cell j in the grid space.

The results are summarized in Table 2. Time limit is set to be 1000 seconds, and average running time standard deviation are calculated in the same way as in the experiment on synthetic data. As shown, proposed method is able to solve all instances in about 60 seconds. On the other hand, compared methods fail to process 2km × 2km and 1km × 1km datasets regardless of the value of λ. This shows the effectiveness of the proposed method.

5.4 EM algorithm: Synthetic data

As mentioned, MAP inference is used for conducting E-step of EM algorithm to estimate the number of moving people and probabilistic model parameters. Here, we compare EM algorithm performance achieved with the proposed method and with the existing method using simulation data. We consider people movement in an $L \times L$ sized grid space ($L = 10, 12$). We construct transition matrix $\boldsymbol{\theta}^{\mathrm{true}}$ by $\theta_{ij} \propto s_i \cdot \exp(-\beta \cdot \mathrm{dist}(i, j))$, where $s_i > 0$ ($i \in [n]$) is a parameter that represents how likely people are to gather at area j, and β is a parameter that controls the decay of transition probability with increasing distance between i and j. This transition matrix is a variant of the one used in (Akagi et al. 2018). We set $\beta^{\mathrm{true}} = 0.5$ and s_i^{true} as follows: first, we randomly selected 3 areas from $[n]$ and set $s_i^{\mathrm{true}} = 10$. For other areas, we set $s_i^{\mathrm{true}} = 1$. We generate the population of each area, $\boldsymbol{N}$, and number of moving people between areas, $\boldsymbol{M}$, by simulating people movement following the procedure written in Section 3.1 until timestep $T = 10$ using transition matrix $\boldsymbol{\theta}^{\mathrm{true}}$. We set initial population $N_{1,i}$ to 10^4 ($i \in [n]$).

Our task is to estimate the number of moving people, $\boldsymbol{M}$, from observed population $\boldsymbol{N}$ by the EM algorithm. For details of the EM algorithm, please see (Akagi et al. 2018). In the algorithms, Γ_i is set to be $[n]$ for $\forall i \in [n]$. We evaluate algorithm performance by Normalized Absolute Error (NAE) of $\boldsymbol{M}$, which is calculated by $\sum_{t,i,j} \left| M_{tij}^{\mathrm{true}} - M_{tij}^{\mathrm{estimated}} \right| / \sum_{t,i,j} M_{tij}^{\mathrm{true}}$. EM algorithm is iterated 200 times for each method. Figure 4 plots NAE versus the elapsed time for the EM algorithm with proposed method and previous method. It can be seen that the proposed method yields better NAE values more quickly than the previous method, especially at large L. For example, in the case of $L = 12$, it took the L-BFGS-B method about 9657 seconds to reach 1.15 for NAE (the dashed line in Figure 4). The proposed method, on the other hand, took only 24 seconds or so, which is about 400 times faster.

6. Related Work

Several methods have been proposed to realize MAP inference efficiently in CGM, which is a general framework including CFDM, (Sheldon et al. 2013; Sun, Sheldon, and Kumar 2015; Nguyen et al. 2016; Vilnis et al. 2015). Note that existing methods provide non-exact MAP inference and output non-integer solutions. In (Akagi et al. 2018), an ef-

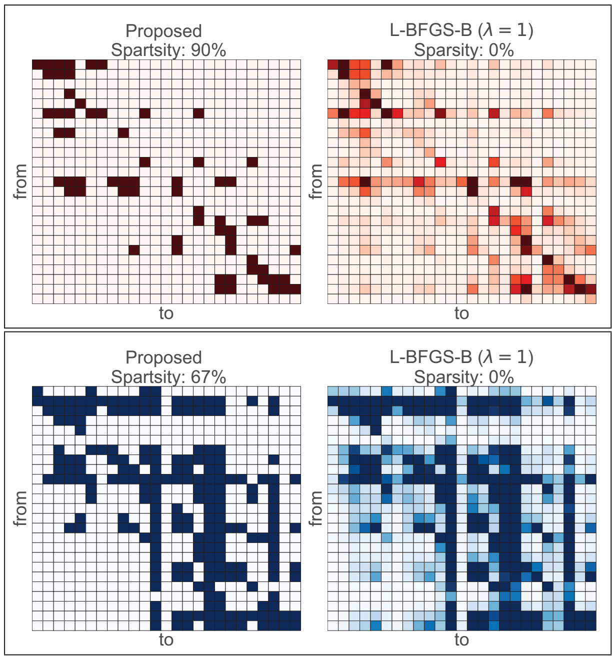

Figure 3: Comparison of optimum solution matrix in an $L \times L$ grid space obtained by proposed method and L-BFGS-B ($\lambda = 1$) with $\boldsymbol{\theta}$ type of "Exponential decay". The left is when $(L, F) = (5, 10^2)$ and the right is when $(L, F) = (5, 10^3)$, where F is the total population of the targeted areas. Sparsity pattern of obtained $L^2 \times L^2$ solution matrix is presented as a heatmap. (i, j)-element of solution matrix represents the number of moving people from area i to area j. In order to investigate sparsity structure of solutions, maximum value of color map is set to be 1 and minimum value is 0. The output of L-BFGS-B method is blurred and contains a lot of small but non-zero elements. In contrast, solution by proposed method is noticeably sparse.

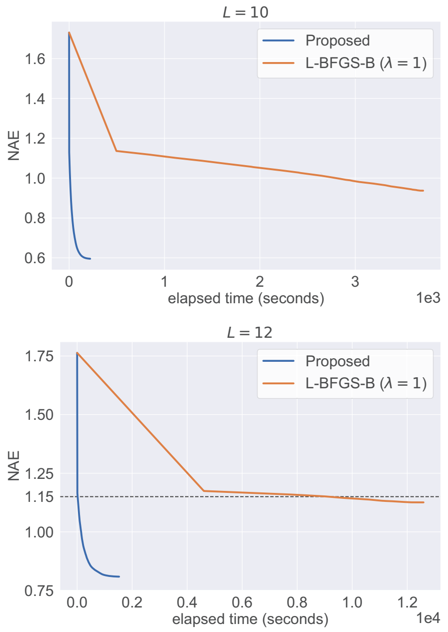

Figure 4: NAE (Normalized Absolute Error) as a function of elapsed time for EM algorithm with each MAP inference method.

ficient optimization method for CFDM is proposed, but it can be used only under a specially factorized probabilistic model, which is designed to model human movements in urban spaces. In contrast, the proposal of this paper is widely available and poses no excessive constraints on the underlying transition model structure.

There is a lot of work on people flow estimation via CFDM. For example, (Iwata et al. 2017; Akagi et al. 2018; Iwata and Shimizu 2019) deal with the estimation of people flows in urban spaces by utilizing variational inference, a factorized probabilistic model, or neural networks. In (Kumar, Sheldon, and Srivastava 2013) and (Tanaka et al. 2018), the inflow and outflow of each area at each timestep are assumed to be available, while (Tanaka et al. 2018) considers a time delay between before and after movement. Thus, there are many variations in terms of the observation model and the probabilistic model underlying movement. The method proposed herein can be used as a subroutine in any of these approaches by appropriately constructing instances of MCFP to suit the problem.

Attempts to estimate human movement from aggregated count data have received a lot of attention. As a particularly relevant study, Xue et al. proposed an algorithm for recovering personal trajectories from aggregated count data for the purpose of evaluating privacy risk for publishing such data (Xu et al. 2017). Sheldon et al. proposed a method to reconstruct sample paths of a Markov chain from partial observations for the purpose of analyzing bird migration patterns (Sheldon, Elmohamed, and Kozen 2008). Although those methods are similar to our method in the sense of solving combinatorial assignment problems to recover movement from aggregated data, there are two distinct differences: (i) Those methods focus on recovering each individual trajectory, not the collective movement of targets. (ii) Those method do not have a mechanism to estimate the parameters of movement models.

Many studies on another direction, predicting population or people flow in cities, have been published (Konishi et al. 2016; Zhang et al. 2019; Jiang et al. 2019). Their approach is to forecast future city dynamics at each area from past data or other features in a supervised way, using classical regression models or deep learning architecture, etc. Our purpose is estimating people flows between areas from only population snapshots at incremental timesteps in a unsupervised way, which is a totally different task from future prediction.

7. Conclusion

In this paper, we proposed a novel method for MAP inference in collective flow diffusion model. First, we showed that the MAP inference problem can be formulated as a minimum convex cost flow problem. Based on this formulation, we proposed an efficient algorithm for MAP inference problem using capacity scaling algorithm. Extensive evaluations on both real and synthetic datasets showed that our algorithm outperforms previous alternatives in terms of running time and optimum solution quality.

References

Ahuja, R. K.; Magnanti, T. L.; and Orlin, J. B. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc.

Akagi, Y.; Nishimura, T.; Kurashima, T.; and Toda, H. 2018. A fast and accurate method for estimating people flow from spatiotemporal population data. In *IJCAI*, 3293–3300.

Byrd, R. H.; Lu, P.; Nocedal, J.; and Zhu, C. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing* 16(5):1190–1208.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1(1):269–271.

Du, J.; Kumar, A.; and Varakantham, P. 2014. On understanding diffusion dynamics of patrons at a theme park. In *AAMAS*, 1501–1502.

Iwata, T., and Shimizu, H. 2019. Neural collective graphical models for estimating spatio-temporal population flow from aggregated data. In *AAAI*, 3935–3942.

Iwata, T.; Shimizu, H.; Naya, F.; and Ueda, N. 2017. Estimating people flow from spatiotemporal population data via collective graphical mixture models. *ACM Transactions on Spatial Algorithms and Systems* 3(1):1–18.

Jiang, R.; Song, X.; Huang, D.; Song, X.; Xia, T.; Cai, Z.; Wang, Z.; Kim, K.-S.; and Shibasaki, R. 2019. Deepurbanevent: A system for predicting citywide crowd dynamics at big events. In *KDD*, 2114–2122. ACM.

Jones, E.; Oliphant, T.; Peterson, P.; et al. 2001–. SciPy: Open source scientific tools for Python.

Kiraly, Z., and Kovacs, P. 2012. Efficient implementations of minimum-cost flow algorithms. *Acta Univ. Sapientiae* 4(1):67–118.

Konishi, T.; Maruyama, M.; Tsubouchi, K.; and Shimosaka, M. 2016. Cityprophet: City-scale irregularity prediction using transit app logs. In *Ubicomp*, 752–757. ACM.

Kumar, A.; Sheldon, D.; and Srivastava, B. 2013. Collective diffusion over networks: Models and inference. In *UAI*.

Minoux, M. 1986. Solving integer minimum cost flows with separable convex cost objective polynomially. In *Netflow at Pisa*. Springer. 237–239.

Morimura, T.; Osogami, T.; and Idé, T. 2013. Solving inverse problem of Markov chain with partial observations. In *NIPS*, 1655–1663.

Nguyen, T.; Kumar, A.; Lau, H. C.; and Sheldon, D. 2016. Approximate inference using DC programming for collective graphical models. In *AISTATS*, 685–693.

Sheldon, D. R., and Dietterich, T. G. 2011. Collective graphical models. In *NIPS*, 1161–1169.

Sheldon, D.; Sun, T.; Kumar, A.; and Dietterich, T. 2013. Approximate inference in collective graphical models. In *ICML*, 1004–1012.

Sheldon, D.; Elmohamed, M.; and Kozen, D. 2008. Collective inference on markov models for modeling bird migration. In *NIPS*, 1321–1328.

Sun, T.; Sheldon, D.; and Kumar, A. 2015. Message passing for collective graphical models. In *ICML*, 853–861.

Tanaka, Y.; Iwata, T.; Kurashima, T.; Toda, H.; and Ueda, N. 2018. Estimating latent people flow without tracking individuals. In *IJCAI*, 3556–3563.

Terada, M.; Nagata, T.; and Kobayashi, M. 2013. Population estimation technology for mobile spatial statistics. *NTT DOCOMO Technical Journal* 14(3):10–15.

Vilnis, L.; Belanger, D.; Sheldon, D.; and McCallum, A. 2015. Bethe projections for non-local inference. In *UAI*, 892–901.

Xu, F.; Tu, Z.; Li, Y.; Zhang, P.; Fu, X.; and Jin, D. 2017. Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data. In *WWW*, 1241–1250.

Yang, H., and Zhou, J. 1998. Optimal traffic counting locations for origin–destination matrix estimation. *Transportation Research Part B: Methodological* 32(2):109–126.

Zhang, J.; Zheng, Y.; Sun, J.; and Qi, D. 2019. Flow prediction in spatio-temporal networks based on multitask deep learning. *IEEE Transactions on Knowledge and Data Engineering*.