6ZE-6

最速輸送問題に対する高速近似解法の提案及び避難計画への応用に関する研究

†大田 章雄　‡神山 直之　§瀧澤 重志　¶加藤 直樹
†京都大学　‡九州大学　§大阪市立大学　¶京都大学

1. はじめに

数理モデルとしての避難計画はKamiyama et al.[1]やTakizawa et al.[2]によって研究されている．しかし避難計画において効率的な避難を求めるための計算は，従来手法では時間拡大ネットワーク[3]という手法を用いるために，対象となる地図ネットワーク×時間に依存する計算量が必要であり，大都市や高い時間分解能，長い期間での計算には長大な計算時間が発生してしまい，起こりうるさまざまなシナリオに対して繰り返し計算を行うことが実際上は困難になってしまうという問題がある．

これを解決するため，本研究では時間拡大ネットワークを用いることなく最速避難の近似解を計算するアルゴリズムを提案する．提案手法の計算時間は対象となる空間の範囲のみに依存する．さらに計算機のメモリの使用量が小さいため，ワークステーションを用いなくても汎用のPCで計算することが可能であることも利点である．また実在する地域を対象に従来手法と提案手法の比較を行う．

2. 最速避難と時間拡大ネットワーク

ネットワークフローを避難計画に適用するにあたり，道路情報は交差点を点，交差点間の道路を辺とした有向グラフ，すなわち点集合をV，辺集合をAとしたネットワーク$N = (V, A)$として表される．各辺には容量関数$c : A \mapsto \mathbb{Z}^+$，移動時間関数$\tau : A \mapsto \mathbb{Z}^+$が定義される．避難者は避難開始時に各点に存在するとし，避難者の存在する点（出発点）の集合は$S^+ \subset V$，避難所や津波の予測浸水域外など，避難先となる到達点集合を$S^- \subseteq V \setminus S^+$とするとき，各点には次のように供給関数 $sup : V \mapsto \mathbb{Z}$が定義される．

$$\begin{cases} sup(s) > 0, & s \in S^+ \\ sup(t) < 0, & t \in S^- \\ sup(x) = 0, & x \in V \setminus (S^+ \cup S^-) \end{cases}$$

直感的には$sup(s)$は避難開始時に各出発点に存在する避難者数を，$|sup(t)|$は各到達点に収容可能な人数を表す．したがって，すべての避難者が避難するためには$\sum_{v \in V} sup(v) \leq 0$でなければならない．

以上のようなネットワーク上で期間 $T \in \mathbb{Z}^+$が与えられたとき，避難行動をモデル化した動的フローは，いかなる$a \in A, \theta \in \{0, 1, \cdots, T\}$においても次の3条件を満たす関数$f(a, \theta) \mapsto \mathbb{Z}^+$として定められる．

容量制約　$f(a, \theta) \leq c(a)$

流量保存則

$$\sum_{a \in v^-} \sum_{t=0}^{\theta - \tau(a)} f(a, t) - \sum_{a \in v^+} \sum_{t=0}^{\theta} f(a, t) \geq 0, \quad \forall v \in V \setminus (S^+ \cup S^-)$$

端子点制約

$$\sum_{a \in s^+} \sum_{t=0}^{T} f(a, t) = sup(s), s \in S^+$$

$$\sum_{a \in s^-} \sum_{t=0}^{T} f(a, t) \leq sup(s), s \in S^-$$

ここで，v^-, v^+はそれぞれ点 $v \in V$への流入辺の集合と流出辺の集合を表す．

2.1. 時間拡大ネットワーク

時間拡大ネットワークはFord and Fulkerson[3] によって紹介された．これにより動的ネットワークを静的ネットワークに変換し，静的ネットワークに対する解析手法が適用できるため，動的ネットワークを利用した避難計画にも用いられる．各点の供給関数を考慮した時間拡大ネットワークの構成方法は[2]によって説明されている．しかし前述のように入力ネットワークサイズ Tのサイズのネットワークを構成するため，それを利用した解析に時間がかかってしまうという問題がある．

2.2. 最速避難

直感的に最速の避難を考えると，避難完了時間が最小になるような避難（最早完了避難や最小移動時間最早完了避難），あるいはそれだけでなく各時刻で避難完了可能なすべての人が避難完了しているような避難（近傍優先避難）が考えられるが，避難所の容量を考慮した場合それらの間にはトレードオフが存在する場合がある.[2] 最早完了避難はssからstへの最大流量が避難者数の合計となる最小のTで構成される時間拡大ネットワークの最大フロー計算によって得られる．さらに避難時間の総和を最小にする最小移動時間最早完了避難は，最大フロー計算の代わりに辺の移動時間をコストとしたコスト最小最大フロー計算で求めることができる．近傍優先避難は，Tを0から1ずつ増やしていきながら，構成される時間拡大ネットワーク上でssからstへのコスト最小最大フローを追加していき，端子点制約が満たされたところで計算を終了することで得られる．

時間拡大ネットワークを用いない最速輸送アルゴリズムはHoppe and Tardos[4] によって提案されているが，避難が完了する最小のTを求めるために劣モジュラ関数の最小化を繰り返し計算する必要があり，現時点では点数が1000を超えるような劣モジュラ関数の最小化は実用上は計算が困難であるため[5]，前述したような多数のシナリオに対する繰り返し計算は実際には不可能である．

本研究で提案する手法は近傍優先避難を近似的に獲得するものであり，さらにその場合の避難完了時間の遅延をパラメトリックに調整することを目指す．

A Heuristic for Quickest Transshipment Problem and its Application to Evacuation Planning

†Akio Ohta, Department of Architecture and Architectural Engineering, Kyoto University.
‡Naoyuki Kamiyama, Institute of Mathematics for Industry, Kyusyu University.
§Atsushi Takizawa, Department of Architecture and Building Engineering, Osaka City University.
¶Naoki Katoh, Department of Architecture and Architectural Engineering, Kyoto University.

Copyright ©2014 Information Processing Society of Japan.
All Rights Reserved.

3. 提案手法

動的ネットワーク $N = (V, A, c, \tau, sup, S^+, S^-)$と期間$T$が与えられているとする．提案手法はまずここから実行可能な$Chain\ Flow$: 経路, その経路に流すフローの流量, 流し始める時刻, 流し終える時刻, を獲得する．その後にfeasibilityを保ったまま各Chain Flowを流し始める時刻をできる限り早める(時間的圧縮を行う）ことで, 近傍優先避難に近い避難を得ることができる．

3.1. Chain Flowの獲得

下準備: $super\ source\ ss$, $super\ sink\ st$を追加し, stからすべての出発点へ, すべての到達点からstへ容量 ∞, 移動時間 0の辺を追加する．

Algorithm 1 Get Feasible Chain Flow Set

$\Gamma \leftarrow \emptyset$, $\Gamma^{all} \leftarrow \emptyset$, $t \leftarrow 0$
$f \leftarrow min\text{-}cost\ max\text{-}flow$ from ss to st
while $f \neq empty\ flow$ **do**
  $\Gamma_{new} \leftarrow decomposed\ chains$ of f
  $\gamma.start \leftarrow t$, $\gamma \in \Gamma_{new}$
  $\Gamma \leftarrow \Gamma \cup \Gamma_{new}$, $\Gamma^{all} \leftarrow \Gamma^{all} \cup \Gamma_{new}$
  $v \leftarrow fastest\ event\ terminal$
  $execute$ v event
  $t \leftarrow event\ occurrence\ period$ of v
  $t_{max} \leftarrow t + [\Gamma_v]$
  while events exist in $[t, t_{max}]$ **do**
    $v \leftarrow next\ event\ terminal$
    $execute$ v event
    $t \leftarrow event\ occurrence\ period$ of v
    $t_{max} \leftarrow t + [\Gamma_v]$
  end while
  $t \leftarrow t_{max} + 1$
  $f \leftarrow min\text{-}cost\ max\text{-}flow$ from ss to st
end while
$execute$ events that have not occurred yet

Chain Flow集合はΓ^{all}として得られる．アルゴリズム中で$[\Gamma]$はChain Flow集合 Γ中で最も長いChain Flowの長さ(経路上の辺の移動時間の総和）を表し, Γ_v, $v \in (S^+ \cup S^-)$は, もしvがsourceならvから出発するChain Flow集合を, vがsinkならvに到達するChain Flow集合を表す．

3.2. Chain Flowの時間的圧縮

Chain Flowの出発時刻に関して昇順のprioirity queue $tQueue$が与えられているとする．またChain Flowが出発点を出発してから辺 aに到達するまでにかかる時間を$r(a)$と表すと, Chain Flowを時間的に圧縮するアルゴリズムは次のようになる．

Algorithm 2 Compress Chain Flows in Time

$frontier(a) = 0$, $\forall a \in A$
while $tQueue \neq \emptyset$ **do**
  $t \leftarrow tQueue.top$
  $\Gamma \leftarrow \{\gamma \mid \gamma.start = t, \gamma \in \Gamma^{all}\}$
  for $\gamma \in \Gamma$ **do**
    $g \leftarrow minimum\ gap\ between\ (\gamma.start + r(a))\ and\ frontier(a)$, $\forall a \in \gamma.path$
    $\gamma.start \leftarrow \gamma.start - g + 1$
    $\gamma.end \leftarrow \gamma.end - g + 1$
  end for
  for $\gamma \in \Gamma$ **do**
    $frontier(a) \leftarrow max(frontier(a), \gamma.end + r(a))$, $\forall a \in \gamma.path$
  end for
  $tQueue.pop$
end while

4. 計算実験

環境: Intel(R) Xeon(R) CPUE5-2687Wv2@3.40GHz, RAM: 256GB

4.1. 徳島市沖洲地区

点数: 860, 辺数: 2212, 避難者数: 7445人, 避難所: 11箇所, 地区外点: 3箇所 のネットワークに対して近傍優先避難, 最早完了避難, 提案手法の計算を行った．避難開始からの累積避難完了人数の推移を図1に示す．計算時間は, 近傍優先避難·最早完了避難: 7479.8[秒] に対して 提案手法: 8.7[秒]であった．

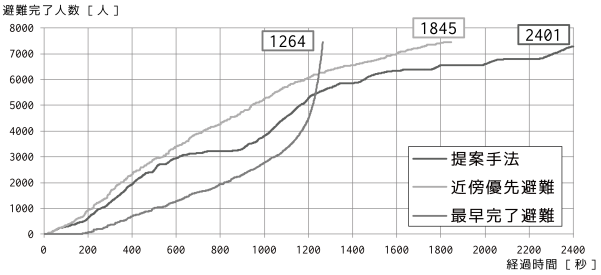

図 1: 徳島市沖洲地区を対象とした計算結果

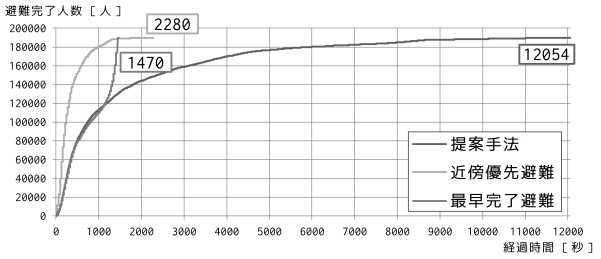

図 2: 大阪市湾岸地域を対象とした計算結果

4.2. 大阪市湾岸地域

点数: 13085, 辺数: 40514, 避難者数: 189489人, 避難所: 197箇所, 浸水範囲外点: 389箇所 のネットワークに対して近傍優先避難, 最早完了避難, 提案手法の計算を行った．避難開始からの累積避難完了人数の推移を図2に示す．計算時間は, 近傍優先避難·最早完了避難: 9372.8[秒] に対して 提案手法: 640.2[秒]であった．ただし, 近傍優先避難·最早完了避難に関してはメモリのオーバーフローを回避するため, 30秒を1単位時間として計算している．(巧みな実装によってオーバーフローを回避した場合でも, 単位時間を1秒とすると計算完了までに5日程度かかるという報告を研究者から受けている.)

5. アルゴリズムの改善

避難完了時間の遅延の抑制, また避難時間の総和を小さくする方法については現在研究中である．

参考文献

[1] N. Kamiyama, A. Takizawa, N. Katoh, and Y. Kawabata: Evaluation of capacities of refuges in urban areas by using dynamic network flows, The 8th International Symposium on Operations Research and Its Applications (LNOR 10), pages 453--460, 2009.

[2] A. Takizawa, M. Inoue, and N. Katoh: An emergency evacuation planning model using the universally quickest flow, The Review of Socionetwork Strategies, 6(1): 15--28, 2012.

[3] D. R. Ford and D. R. Fulkerson: Flows in Networks, Princeton University Press, Princeton, NJ, USA, 1962.

[4] B. Hoppe and E´. Tardos: Polynomial time algorithms for some evacuation problems, In Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms, SODA '94, pages 433--441, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[5] 土村展之: 離散凸解析ソルバODICONとWebアプリケーション(<特集>離散凸解析), オペレーションズ・リサーチ : 経営の科学, 58(6): 339--345, 2013.

Copyright ©2014 Information Processing Society of Japan.
All Rights Reserved.