オークションを用いた クラウド・ソーシング配送システム

渡邊 大樹[1]・赤松 隆[2]

[1]学生会員　東北大学　大学院情報科学研究科人間社会情報科学専攻（〒 980-8579 仙台市青葉区荒巻字青葉 6-6-06-408）
E-mail: taiki.watanabe.p3@dc.tohoku.ac.jp
[2]正会員　東北大学教授　大学院情報科学研究科人間社会情報科学専攻（〒 980-8579 仙台市青葉区荒巻字青葉 6-6-06-408）
E-mail: akamatsu@plan.civil.tohoku.ac.jp

クラウドソーシング・デリバリー (CSD) とは，都市内を日々通行する車両を利用する配送システムである．CSD は，ラストワンマイルの配送を現在の配送システムより効率化できる一方で，多数のドライバーと配送業務のマッチングに大規模な計算を要する課題がある．また，マッチングを最適化するためには，ドライバーの各タスクに対する評価値を管理者が知る必要があるが，これは一般的には難しい．本研究では，これら 2 つの課題を解決した CSD マッチングシステムを提案する．まず，オークションをマッチングに導入することで，ドライバーの評価値を反映させた最適マッチングを達成する．続いて，このオークションに適用する計算効率的なオークション・メカニズムを提案することで，計算量の課題を解決する．数値実験により，提案メカニズムは計算時間を劇的に (1/(ノード数) のオーダーで) 減少させること，問題規模の増加に対する計算量増加が小さいことを示した．

Key Words: クラウドソーシング・デリバリー，ラストワンマイル，オークション

1. はじめに

(1) 背景と目的

現在，都市内配送システムは，配送需要を処理しきれないという問題を抱えている．都市内の配送需要は，E-Commerce の急速な発達によって急増し，同時に急速に小口・多頻度化している．現在の配送システムは，トラックによって多量の荷物をまとめて配送する形態であり，小口・多頻度な配送を行うのには適していない．この需要形態と配送形態の不調和によって，現システムは発生する需要を処理しきれなくなりつつある．この事実は，配送需要に無理に対応した結果生じた宅配業者の労働環境の悪化が社会問題となっていることからも明らかである．このような問題を解決するため，小口・多頻度な需要を効率よく処理できる，新たな配送システムが必要である．

“クラウドソーシングデリバリー (CSD)” は，こうした現在の配送システムが抱える問題を解決する，新たな配送システムである．CSD とは，都市内をトリップする予定の車両ドライバーが，依頼された配送をトリップ時に回り道をして行うシステムである．CSD は，少数のトラックに大量の貨物を積載して配送する現在のシステムと比較して，多数の車両に少量ずつ貨物を積載して配送するという特徴を有する．したがって，小口・多頻度な配送需要への親和性は現在のシステムと比べて高い．また，配送がなくても行われるトリップを利用して配送を行うので，配送のためだけにトラックを走らせる現在のシステムに比べて効率的に配送を行うことができる．実際，複数の研究において，CSD 導入によって配送が効率化されるということが示されている (e.g., Archetti et al.(2016)[1], Arslan et al.(2018)[2], Yildiz and Savelsbergh(2019)[3])．

CSD に関する既存研究の多くは，ドライバーとタスクのマッチングを管理者が中央集権的に (i.e., ドライバーの選好を反映せず) 決定することを想定している．このマッチング方法では，最適な (i.e., 総配送費用を最小にする) マッチングは実現できない．なぜなら，ドライバーの選好 (タスク実行に費やす費用) はドライバーの個人情報であり，管理者はドライバーの意思表示なしにその情報を正確に得ることができないからである．したがって，最適なマッチングを実現するためには，ドライバーの選好を反映できるマッチング決定方法の構築が必要である．

また，既存研究で想定されているシステムの規模は比較的小さく，大規模システムでの運用を考えた研究は少ない．大規模なシステムでの運用を考えると，マッチング計算の計算量が膨大になるという新たな課題が生じる．なぜなら，システム規模が増大するにつれて可能なマッチングの数が爆発的に増えるからである．マッ

チングを計算効率的に実行できるシステムを構築し，都市の広範囲をカバーする大規模なシステムを導入することは，都市内のエリアごとに小規模なシステムを導入して都市全体をカバーするよりも望ましい．なぜなら，前者の方が可能なマッチングの数が多い[1]ため，より配送効率の高いマッチングを実現できるからである．したがって，計算効率的なマッチング方法を構築することも，重要な課題である．

本研究では，以上 2 つの課題 (ドライバーの選好をマッチングに反映する方法，計算効率的なマッチング計算方法の構築) を解決した，新たな CSD マッチングを提案する．提案するマッチング手法の特徴は以下の 2 つである:(1) オークションによる市場取引を利用してマッチングを行う，(2) 計算効率的な本研究独自のオークション・メカニズムを利用する．オークションを導入することで，ドライバーの選好 (i.e., タスク実行に費やす費用) を入札額として各ドライバーに表明してもらうことができ，ドライバーの選好をマッチングに反映することが可能になる．また，オークション・メカニズム (オークションにおけるマッチング決定方法) を計算効率的にすることで，大規模システムにも適用可能になる．

提案メカニズムではまず CSD 市場を複数の小市場に分解し，それぞれの小市場に適切にタスクを配分した上で，各小市場ごとにオークションを用いて配分されたタスクをドライバーとマッチングする．この分解により，個々の小市場でのマッチング計算は非常に小規模になり，かつ全ての小市場の計算を並列して行えるため，タスク配分後のマッチングはごく少ない計算量で行うことができる．一方で，小市場へのタスク配分を決定する問題は依然として非常に大規模である．そこで次に，このタスク配分決定問題の効率的解法を提案する．具体的にはまず，問題の構造を活かしてこの問題を交通ネットワーク配分問題に変形し，ネットワーク構造を利用して縮約する．次に，縮約した問題に対する効率的アルゴリズムを提案する．最後に，提案アルゴリズムが計算時間を劇的に (1/(ノード数) のオーダーで) 減少させること，問題規模の増加に対する計算量増加が小さいことを，数値実験によって示す．

市場を用いた CSD システムを扱った既存研究は，著者らの知る限り Allahviranloo et al.(2019)[4] のみである．しかし，この研究は CSD 導入によるドライバーの行動変化を分析したものであり，提案 CSD システムの社会的効率性を分析する本研究とは性質が異なる．また，大規模システムにおけるマッチング決定問題の計算効率的解法に関する研究も少なく，Wang et al.(2016)[5] が唯一のものである．しかし，この研究では 1 つの配送業者のみが CSD を導入している状況を想定しているため，提案されている解法は配送拠点と配送先が 1 対 1 に対応している場合にしか適用できない．複数の配送業者の配送を一つの CSD 運営主体が統合的に扱う，より一般的な状況に適用できるようにするために，配送拠点と配送先の組み合わせを制限しない状況におけるシステムを考えることが望ましい．本研究で提案する解法は，後者のより一般的な状況において適用可能である．

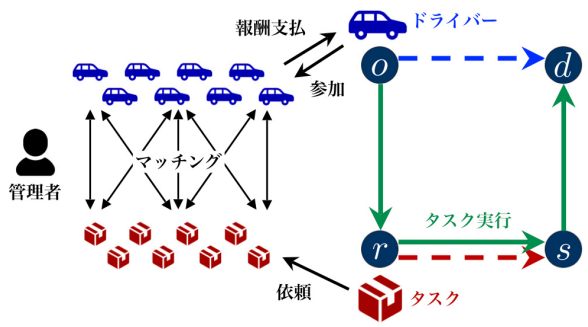
本稿の構成は以下の通りである．まず，第 2 章で本研究で考える状況設定と提案システムの概要を示す．続く第 3 章では，提案システムにおける市場均衡状態を定式化し，その状態が社会的に最適であることを示す．第 4 章では，既存のオークション・メカニズムを大規模 CSD システムに適用することを考え，オークションによって理論上は市場均衡状態を達成できるものの，計算量が膨大になり実用上は適用不可能であることを示す．第 5 章では，この計算量的課題を解決した計算効率的なオークション・メカニズムを提案する．第 6 章では，提案メカニズム内で解くタスク配分決定問題を交通ネットワーク配分問題に変換し，問題を縮約する．第 7 章では，縮約された問題を解く計算効率的アルゴリズムを示す．第 8 章では，このアルゴリズムの効率性を数値実験を通して示す．最後に，第 9 章で結論と今後の課題を述べる．

2. モデル

本章では，本研究で想定する CSD システムの全体像，および関わる主体の行動を示す．本研究では，一つの CSD システムで扱う都市のエリアをノード集合 $\mathcal{N} = \{1, 2, ..., N\}$ とリンク集合 $\mathcal{L}$ によって構成される交通ネットワーク $G(\mathcal{N}, \mathcal{L})$ として扱う．ドライバーのトリップおよび配送業務の起終点は全て $\mathcal{N}$ に含まれるノードとして表す．配送を行うドライバーの総数はネットワークの交通量全体に比べて十分小さく，配送を行うドライバーの交通量はネットワークの各リンクの所要時間に影響を及ぼさないとする．言い換えると，本研究ではネットワークの各リンクの所要時間が交通量に依存しない定数であると仮定する．

想定する CSD システムの全体図を**図–1** に示す．図中の “タスク” とは，システムで用いる配送業務の単位であり，同一起終点の配送業務を標準的な自動車で遂行できる (i.e., 車両の余剰スペースが不足しない程度の) 件数まとめたものである．配送業務をタスク単位にまとめる際は，各タスクができるだけ同質になるように行う．そのため，同一 OD ペアのタスクは同質であり，

[1] 後者は，前者で可能なマッチングのうち,「同じエリアのドライバーとタスクのみがマッチングされる」もののみが実現可能である．



図–1 CSD システムの全体図

タスクはその OD ペア $r, s \in \mathcal{N}$ のみで区別すれば十分である．

このシステムにおける行動主体は，配送を行う “ドライバー”，システムの “管理者” の 2 つである．それぞれの主体の行動を以下で述べる．

“管理者” は，社会的余剰の最大化を目指してタスクとドライバーのマッチングを行う主体である．管理者はまず，配送業務およびシステムに参加するドライバーを募集する．ドライバーの総数が配送タスクの総数を上回った場合は，ドライバーを過去の業務履歴等で選抜し，配送タスクの総数以下となるようにする．次に，ドライバーが配送タスクを取引するためのオークションを実行し，タスクとドライバーを 1 対 1 にマッチングする．その後，ドライバーに配送を依頼し，報酬を支払う．CSD によって実行されなかったタスクは，管理者自身が実行する．管理者自身がタスク rs を実行する際の費用を $\bar{c}_{rs}$ とする．

“ドライバー” は，元々ある OD ペア $o, d \in \mathcal{N}$ 間のトリップ (e.g. 居住地から勤務先等) を予定しており，そのトリップの途中に配送を行う主体である．各ドライバーを a と表し，全ドライバーの集合を $\mathcal{A}$ とする．ドライバーは自らの効用を最大化するように配送タスクの選択を行う．具体的な行動プロセスは以下の通りである．まず，管理者に自らのトリップの OD ペア od を伝え，システムへの参加を表明する．次に，管理者が創設したオークション市場で配送タスクを獲得する[2]．最後に，獲得した配送タスクをトリップの途中で行い，管理者から報酬を受け取る．具体的には，ドライバーはネットワークのノード o から出発し，獲得したタスクの配送元 r に訪れ荷物を受け取る．その後配送先 s を訪れ配送を完了し，d に到着して自身のトリップを完了する．ドライバーはこの配送によって費用 c_a^{rs} を費やす．報酬額はタスクの OD ペアごとに決まる額 u_{rs} である．

[2] オークション市場での具体的なドライバーの行動は 5. 章で解説する．

3. 市場均衡状態

本章では，提案システムにおける市場均衡状態を定式化し，その均衡状態におけるマッチングが社会的余剰を最大化することを示す．

(1) 市場均衡状態の定式化

オークション市場により達成される市場均衡状態では，以下の 3 つの条件が同時に成立する．

a) ドライバーのタスク選択に関する均衡条件

均衡状態では，どのドライバーも自分だけがタスクを変更しても効用を改善できない．ドライバー a がタスク rs を実行する場合のドライバーの効用は，受け取る報酬 u_{rs} とタスク実行に要する費用 c_a^{rs} の差 $u_{rs} - c_a^{rs}$ で定義される．この値を用いて，均衡状態は以下のように定式化される．

$$\begin{cases} u_{rs} - c_a^{rs} = \rho_a & \text{if} \quad y_a^{rs} > 0 \\ u_{rs} - c_a^{rs} \leq \rho_a & \text{if} \quad y_a^{rs} = 0 \end{cases} \qquad \forall a \in \mathcal{A}, r, s \in \mathcal{N} \tag{1}$$

$$\text{where} \quad \rho_a = \max_{r,s} \{u_{rs} - c_a^{rs}\}$$

ここで，y_a^{rs} は，ドライバー a がタスク rs にマッチングされたときに 1，そうでないときに 0 となる変数である．

b) ドライバーの保存則

システムへの参加を表明したドライバーは必ず 1 つのタスクとマッチングされる．この条件は以下のように表現される．

$$\sum_r \sum_s y_a^{rs} = 1 \qquad \forall a \in \mathcal{A} \tag{2}$$

c) 需給均衡条件

均衡状態においてあるタスク rs に注目したとき，タスクの報酬が最大値 $\bar{c}_{rs}$ 未満[3]であれば需給が均衡しており，そうでなければドライバーの労働供給不足である．

$$\begin{cases} \sum_a y_a^{rs} = n_{rs} & \text{if} \quad u_{rs} < \bar{c}_{rs} \\ \sum_a y_a^{rs} \leq n_{rs} & \text{if} \quad u_{rs} = \bar{c}_{rs} \end{cases} \qquad \forall r, s \in \mathcal{N} \tag{3}$$

ここで，n_{rs} は OD ペアが rs であるタスクの数である．

(2) 市場均衡状態の効率性

市場均衡状態の効率性を分析するためのベンチマークとして，社会的余剰を最大化する状態 (SO 状態) を定義する．ドライバー a がタスク rs を落札したときの社会的効用 w_a^{rs} は，$w_a^{rs} \equiv \bar{c}_{rs} - c_a^{rs}$ である．この社会的効用を用いて，社会的最適状態は以下の最適化問題 [SO-P] の解として定義される:

[3] 管理者は自身でそのタスクを実行する場合の費用 $\bar{c}_{rs}$ 以上をドライバーに支払うことはない．

[SO-P]

$$\max_{\boldsymbol{y}} \sum_a \sum_{rs} w_a^{rs} y_a^{rs} \tag{4}$$

$$\text{s.t.} \sum_{rs} y_a^{rs} = 1 \qquad \forall a \in \mathcal{A} \tag{5}$$

$$\sum_a y_a^{rs} \leq n_{rs} \qquad \forall r,s \in \mathcal{N} \tag{6}$$

$$y_a^{rs} \in \{0,1\} \qquad \forall a \in \mathcal{A}, r,s \in \mathcal{N} \tag{7}$$

そして，前節で定式化した提案システム下での市場均衡状態において実現するマッチングは，[SO-P] の最適解として得られるマッチングと一致することが示される．すなわち，以下の命題が成立する．

命題 1. 提案システムの市場均衡状態において実現するマッチングは，社会的総余剰を最大化する．

(証明：付録 I 参照)

4. オークション・メカニズム適用の問題点

本章では，オークションを利用して前述の市場均衡状態を達成することを考え，その際に生じる問題点を整理する．結論を先に述べると，CSD 市場には単純なオークションメカニズムをそのまま適用することはできない．なぜなら，以下の 2 つの問題が生じるからである:

1. マッチングに要する計算量が膨大である
2. ドライバーの入札手続きが複雑である

オークションでは，まず費用 c_a^{rs} をドライバー自身に表明させ，その値を用いて [SO-P] を解きマッチングを決定したのち，タスク価格を市場均衡価格に設定することで市場均衡状態を実現する．より具体的には，一般的な封印入札オークションでは，以下のような手続きでマッチングとタスクの価格決定を行う．

1. **入札**: 各ドライバー a は全てのタスク rs に対して，自らの希望報酬額 b_a^{rs}(ドライバー a がタスク rs を行っても良いと感じる最低の報酬額であり，真の値は c_a^{rs}) を表明する．
2. **マッチング決定**：オークション管理者は，勝者決定問題 (問題 [SO-P] において w_a^{rs} を $\bar{c}_{rs} - b_a^{rs}$ に置き換えた問題) を解き，マッチング y_a^{rs} を決定する
3. **価格決定**: 各ドライバーの入札額 b_a^{rs} に応じて，タスクの報酬を決定する．

オークション・メカニズムによりマッチングを決定する最大の利点は，希望報酬額 c_a^{rs} の値を正確に反映した最適マッチングを達成できることである．理論上は，[SO-P] を解きさえすれば社会的最適マッチングを得ることは可能である．しかし，希望報酬額がドライバーのみ知る個人情報であり，管理者はこの値を観測することはできない．したがって，希望報酬額を得られるような工夫をしない限り，管理者が [SO-P] を解いて最適マッチングを決定することは不可能である．オークションは，ドライバー自身に希望報酬額を表明させることでこの課題を解決している．

しかし，ドライバーが正直な表明を行う (i.e., $b_a^{rs} = c_a^{rs}$ と表明する) 保証はない．実際，メカニズムの設計 (e.g., 価格決定の方法) によっては，ドライバーが虚偽の表明を行うことで利得を増加させることができる場合もある．この場合，管理者は係数 c_a^{rs} の値を正しく得ることができないため，オークションによって市場均衡状態を実現することはできない．

この問題点を解決したメカニズムとしてもっとも基本的なものが，Vickrey[6],Clarke[7],Groves[8] によって提案された VCG メカニズムである．VCG メカニズムは価格決定方法を工夫し，ドライバーが虚偽の表明を行うことで利得を増加させることができないように設計されている．したがって，VCG メカニズムを適用すれば市場均衡状態が実現する．

しかし，VCG メカニズムによりドライバーの虚偽入札の問題を解決したとしても，マッチングを決める際に解く [SO-P] が非常に大規模であり，現実的な時間で解くことができないという問題が残る．これは，前述した問題点の 1 つ目である．具体的には，[SO-P] の変数 y_a^{rs} は a,r,s の組に対して 1 つ存在するため，変数の数は $|\mathcal{A}| \times |\mathcal{N}|^2$ である．さらに，ドライバー数 $|\mathcal{A}|$ は一般的にドライバーの OD ペア数 $|\mathcal{N}|^2$ より多いため，[SO-P] の問題規模 (変数の数) は $O(|\mathcal{N}|^4)$ のオーダとなる．そのため，例えば 100 ノード程度のネットワークであっても変数は最大 1 億個にもなり，現実的な時間で解くことは難しい．

加えて，前述した問題点の 2 つ目である，ドライバーの入札手続きが複雑であるという問題もある．VCG メカニズムにおいてドライバーは全てのタスク OD ペアに入札しなければならないが，タスク OD ペアの数は最大 $|\mathcal{N}|^2$ と膨大になるため，この入札手続きは非常に大きい．

以上の 2 つの問題から，VCG メカニズムを CSD 市場に適用することはできない．次章以降では，この問題点を解消したオークション・メカニズムを提案する．

5. 提案オークション・メカニズム

本章では，前章で示した問題点を解決したオークション・メカニズムを提案する．提案メカニズムではまず，ドライバーの OD ペアごとに市場を分割し，同一 OD

ペアのドライバーのみが参加する小市場を OD ペア数分作る．その上で，以下の 2 フェイズの手続きでマッチングを決定する:

フェイズ 1: 管理者が各小市場にタスクを最適に配分する (マスター問題).

フェイズ 2: 各小市場ごとに独立にオークションを行う (サブ問題).

提案メカニズムは，社会的最適マッチングを定義する最適化問題である [SO-P] を 2 段階の問題に分解し，階層的に解くことに相当する．より具体的には，[SO-P] を各 OD ペアへの最適タスク配分を決定するマスター問題 (フェイズ 1 に相当) と，マスター問題で得られたタスク配分数を与件として OD ペアごとに最適マッチングを決定するサブ問題 (フェイズ 2 に相当) に分解し，この 2 つの問題を順に解くことでマッチングを決定する．サブ問題は [SO-P] と同じ形式の問題であり，既存のオークション・メカニズムによりマッチングを決定する．マスター問題では，管理者が中央集権的に最適タスク配分を決定することができる．具体的には，サブ問題を解いた結果 (i.e., 最適値関数) を過去のマッチング結果を基に推定し，この推定値を最小化するタスク配分を決定する．これに必要なのは管理者のもつ過去のマッチング履歴のみであり，ドライバーの意思表示は要さない．

提案メカニズムにより，前章で挙げた問題点の 1 つである計算量は大幅に減少する．まず，フェイズ 2 に対応するサブ問題は，VCG メカニズムなどの既存オークション・メカニズムを用いて，[SO-P] と比較して非常に少ない計算量で解くことができる．なぜなら，OD ペア別に分割された個々の問題はマッチングすべきドライバーの数が少ないごく小規模な問題であり，さらに全ての問題を並列して解くことができる．また，フェイズ 1 に対応するマスター問題は大規模であるが，CSD 特有の問題構造を活かして問題を大幅に縮約することが可能である．この問題の縮約により，マスター問題も計算効率的に解くことができる問題となる．以上のように，マスター問題，サブ問題ともに計算効率的に解くことができるため，提案メカニズムは既存のメカニズムと比較して計算量を大幅に削減できる．

また，残る問題点であるドライバーの入札手続きの煩雑さも軽減される．フェイズ 1 で小市場へのタスク配分が最適化されるため，各 OD ペアごとの小市場には，その OD ペアのドライバーが行うと非効率的なタスクは配分されない．そのため，各小市場において選択できるタスクの種類は減少する．ドライバーは全てのタスクに対して入札を行うため，タスクの種類が減少することで入札手続きが軽減される．

以降では，上述のような提案メカニズムを構成する各要素を解説する．まず本章において，前述した [SO-P] の分解・階層化を行い，提案メカニズムの各フェイズに対応する最適化問題を示す．さらに，サブ問題の最適値関数の推定値を導出し，マスター問題を管理者が中央集権的に解ける問題に変形する．続いて次章以降では，変形されたマスター問題を縮約し，効率的アルゴリズムを構築する．

(1) 提案メカニズムの手続き

本節では，提案メカニズムの各フェイズにおける手続きをより具体的に示す．具体的には，[SO-P] を提案メカニズムのフェイズ 1 に対応する [SO-P/Master] およびフェイズ 2 に対応する [SO-P/Sub] に分解し，提案メカニズムがこの 2 つの問題を順に解くことで社会的最適マッチングを達成することを示す．

まず，各 OD ペア od へのタスク rs の配分量を表す集計的変数 f_{od}^{rs} を用いて，[SO-P] を以下のように [SO-P-2] に等価変形する:

[SO-P-2]

$$\max_{\boldsymbol{y},\boldsymbol{f}} \sum_{od} \left[\sum_{a\in\mathcal{A}_{od}} \sum_{rs} w_a^{rs} y_a^{rs} \right] \tag{8}$$

$$\text{s.t.} \sum_{rs} y_a^{rs} = 1 \qquad \forall a \tag{9}$$

$$\sum_{od} f_{od}^{rs} \le n_{rs} \qquad \forall rs \tag{10}$$

$$\sum_{a\in\mathcal{A}_{od}} y_a^{rs} = f_{od}^{rs} \qquad \forall od, rs \tag{11}$$

$$f_{od}^{rs} \in \mathbb{Z}_+ \qquad \forall od, rs \tag{12}$$

$$y_a^{rs} \in \{0,1\} \qquad \forall a, rs \tag{13}$$

ここで，$\mathcal{A}_{od}$ は OD ペアが od であるドライバーの集合，$q_{od} = |\mathcal{A}_{od}|$ である．[SO-P-2] は，[SO-P] の制約条件 (6) を等価な条件 (10),(11) に置き換えた問題であることから，[SO-P] と等価であることがわかる．

さらに [SO-P-2] を，1 つのマスター問題とドライバー OD ペア数個のサブ問題に階層化する．マスター問題では各 OD ペアへの集計的な最適タスク配分 $\boldsymbol{f}^*$ を決め，サブ問題ではマスター問題で得られた $\boldsymbol{f}^*$ を与件として，OD ペアごとに個々のドライバーへの最適タスク割当 $\boldsymbol{y}^*$ を決定する．このように階層化された問題をマスター問題，サブ問題 の順で解くことで，[SO-P-2] の解，すなわち [SO-P] の解を得ることができる．

まず，サブ問題はドライバーの OD ペアごとに分解可能である．なぜなら，サブ問題で用いる変数 $\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{f}$ は全てドライバーの OD ペアごとに分けて扱うことができるからである．OD ペア od に関するサブ問題 [SO-P/Sub(od)] は以下のような問題である:

[SO-P/Sub(od)]

$$\max_{\boldsymbol{y}_{(od)}} z_{od}(\boldsymbol{y}_{(od)}|\boldsymbol{f}) \equiv \sum_{a\in\mathcal{A}_{od}}\sum_{rs} w_a^{rs} y_a^{rs} \tag{14}$$

$$\text{s.t.} \sum_{rs} y_a^{rs} = 1 \qquad \forall a \in \mathcal{A}_{od} \tag{15}$$

$$\sum_{a\in\mathcal{A}_{od}} y_a^{rs} = f_{od}^{rs} \qquad \forall rs \tag{16}$$

$$y_a^{rs} \in \{0,1\} \qquad \forall a \in \mathcal{A}_{od}, rs \tag{17}$$

ここで，$\boldsymbol{y}_{(od)} = \{y_a^{rs}\}_{\forall rs, a\in\mathcal{A}_{od}}$ である．この問題は，OD ペア od に割り当てられたタスクを各ドライバーに最適にマッチングする問題であり，これをオークションメカニズムを用いて ($\forall o, d \in \mathcal{N}$ について並列に) 解くことが提案メカニズムのフェイズ 2 に相当する．

一方，マスター問題 [SO-P/Master] は以下のような問題である:

[SO-P/Master]

$$\max_{\boldsymbol{f}} \sum_{od} z_{od}^*(\boldsymbol{f}) \tag{18}$$

$$\text{s.t.} \sum_{rs} f_{od}^{rs} = q_{od} \qquad \forall od \tag{19}$$

$$(10), (12) \tag{20}$$

$$\text{where} \quad z_{od}^*(\boldsymbol{f}) = \max_{\boldsymbol{y}_{(od)}} \{ z_{od}(\boldsymbol{y}_{(od)}|\boldsymbol{f}) \ |(15), (16), (17)\} \tag{21}$$

ここで，$z_{od}^*(\boldsymbol{f})$ は，サブ問題 [SO-P/Sub(od)] の最適値関数である．また，制約条件 (19) は，OD ペア od へのタスク配分総数が，OD ペア od 内のドライバー数に一致するための条件である．この条件が満たされなければ，[SO-P/Sub(od)] が実行不可能となる．この問題は，各 OD ペアで最適マッチングが達成された時の総余剰を最大化するタスク配分 $\boldsymbol{f}$ を決定する問題であり，これを解くことが提案メカニズムのフェイズ 1 に相当する．

以上のように，提案メカニズムはフェイズ 1 で [SO-P/Master] を解き，フェイズ 2 で [SO-P/Sub] をオークションにより解くことで社会的最適マッチングを実現する．

残る問題は，[SO-P/Master] をどのように解くかである．[SO-P/Master] を解くためには [SO-P/Sub] の最適値が必要である．しかし，提案メカニズムでは [SO-P/Master], [SO-P/Sub] の順で最適化問題を解くため，提案メカニズムでマッチングを決定するためには，[SO-P/Sub] を解かずに [SO-P/Sub] の最適値 $z_{od}^*(\boldsymbol{f})$ を得る必要がある．そこで次節では，$z_{od}^*(\boldsymbol{f})$ を $\boldsymbol{f}$ を用いた Closed-form の関数として表現し，[SO-P/Master] を [SO-P/Sub] の結果を用いず解くことができる問題に変形する．

(2) [SO-P/Master] の Closed-form 表現

本節では，[SO-P/Master] の目的関数を $z_{od}^*(\boldsymbol{f})$ の期待値を用いて計算することで，[SO-P/Master] を Closed-form の問題に変形する．$z_{od}^*(\boldsymbol{f})$ の最適値の期待値は，[SO-P/Sub(od)] を解くことなく，各 OD ペアにおけるドライバーの私的効用分布を用いて計算することができる．最適値の真値は実際に [SO-P/Sub(od)] を解かなければ得られないが，OD ペア内のドライバーが増えるほど真値と期待値の間の誤差は小さくなる．そのため，同一 OD ペアのドライバーがある程度以上の数存在すれば[4]，期待値を使って [SO-P/Master] を解いたとしても，得られる最適タスク配分に大きな誤差は生じないと考えられる．

本研究では，私的効用分布が以下のように表されると仮定する:

仮定 1. ドライバー a の OD ペアが od であるとき，ドライバーの費やす費用 c_a^{rs} は，以下のような形で表される:

$$c_a^{rs} = C_{od}^{rs} - \epsilon_a^{rs} \tag{22}$$

ここで，C_{od}^{rs} は管理者が観測可能な交通費用，ϵ_a^{rs} は観測不可能な私的効用を表す．$\mathcal{A}_{od}$ 内におけるドライバーの私的効用 ϵ_a^{rs} の頻度分布は，od, rs のごとに i.i.d. である Gumbel 分布 (分布パラメータ θ)[5]に従い，管理者はこの頻度分布を観測できる．

この分布の下で，$z_{od}^*(\boldsymbol{f})$ の期待値は以下のような $\boldsymbol{f}$ に関する Closed-form の関数となる．

補題 1. [SO-P/Sub(od)] の最適値関数 $z_{od}^*(\boldsymbol{f})$ は，OD ペア od のドライバーが十分多ければ以下のように表される:

$$z_{od}^*(\boldsymbol{f}) = \sum_{rs} W_{od}^{rs} f_{od}^{rs} - \frac{1}{\theta}\sum_{rs} f_{od}^{rs} \ln \frac{f_{od}^{rs}}{q_{od}} \tag{23}$$

$$\text{where} \quad W_{od}^{rs} = \bar{c}_{rs} - C_{od}^{rs} \tag{24}$$

(証明:付録 II 参照)

さらに，[SO-P/Sub(od)] の目的関数値は私的効用を含み，管理者は計算できなかったが，この期待値はドライバーの私的効用を含まないため，管理者が直接計算することができる．私的効用の分布情報は必要だが，これは管理者が観測できる．具体的には，分布の観測に必要な観測交通費用は道路ネットワークの混雑状況などから観測可能であり，分散パラメータは過去に行われたオークションで得られたドライバーの私的効用の実現値から統計的に推定すれば良い．

[4] こうなるようにネットワークの解像度を調節すれば

[5] ϵ_a^{rs} の頻度分布が i.i.d. であるという仮定は，提案メカニズムの手続きをわかりやすく示すために導入するものであり，Network GEV 型選択モデル[9] を用いることで i.i.d. でない分布を仮定したケースにも一般化可能である．

提案メカニズムでは，補題 1 で得られた $z^*_{od}(\boldsymbol{f})$ を [SO-P/Master] に代入し，さらに $\boldsymbol{f}$ の整数制約を実数制約に緩和した問題 [SO-A-P] を解くことでタスク配分を決定する．[SO-A-P] は以下のような問題である:

[SO-A-P]

$$\max_{\boldsymbol{f}\geq\boldsymbol{0}} \sum_{od}\sum_{rs} W^{rs}_{od} f^{rs}_{od} - \frac{1}{\theta}\sum_{od}\sum_{rs} f^{rs}_{od} \ln \frac{f^{rs}_{od}}{q_{od}} \tag{25}$$

$$\text{s.t. } (10), (19) \tag{26}$$

[SO-A-P] の解は，その双対問題を解くことでも得ることができる．双対問題は以下の命題により得られる．

命題 2. [SO-A-P] の双対問題 [SO-A-D] は以下のように与えられる:

[SO-A-D]

$$\min_{\boldsymbol{v}} \sum_{rs} n_{rs} v_{rs} + \sum_{od} q_{od} V_{od} \tag{27}$$

where

$$\begin{aligned} V_{od} &\equiv E[\max_{rs}\{w^{rs}_a - v_{rs}\}] \\ &= \frac{1}{\theta}\ln\sum_{rs}\exp[\theta\{W^{rs}_{od} - v_{rs}\}] \end{aligned} \tag{28}$$

(証明:付録 II 参照)

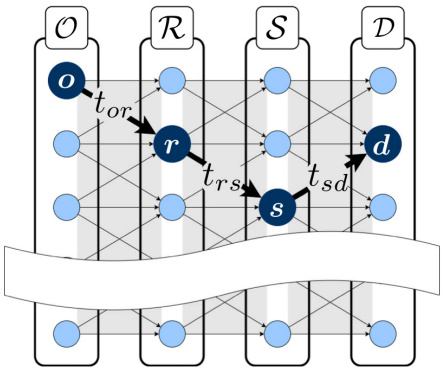
以上により，タスク配分を決定する [SO-P/Master] は [SO-P/Sub] を解く前に管理者が直接解くことができる問題に変形された．すなわち，提案メカニズムは [SO-A-P]，[SO-P/Sub] をこの順に解くことで社会的最適マッチングを得ることができる．しかし，[SO-A-P] は [SO-P] と同じく大規模な問題であり，直接解くと莫大な計算時間を要する．なぜなら，変数 f^{rs}_{od} の数が $O(|\mathcal{N}|^4)$ のオーダであるからである．これは [SO-P] の問題規模のオーダと同じであり，前述の通りネットワーク規模に対して急激に問題規模が増大する．そこで次章では，[SO-A-P] を縮約する変形を行う．

6. タスク配分決定問題の縮約

本章では，[SO-A-P] を縮約し，効率的に解くことができる問題に変換する．具体的にはまず，ドライバーとタスクのマッチングを経路への交通配分として表現できる仮想的な交通ネットワークを構築する．続いて，[SO-A-P] をその仮想ネットワーク上の交通量配分問題に変換する．最後に，作られた交通量配分問題を，仮想ネットワークの構造を利用して縮約する．

(1) 仮想ネットワークの構築

本節では，交通量配分問題への変形の際の配分対象とする仮想的な交通ネットワーク $G_v(\mathcal{N}_v, \mathcal{L}_v)$ を構築する．第 2 章で示したように，ドライバーは配送の際に 2 ノード間の移動を $o-r, r-s, s-d$ の 3 回行う．そこで，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ ではこの 3 回の移動を 3 本のリンクとして表現し，ドライバーとタスクのマッチングを経路への交通配分として表現する．



図–2 仮想ネットワーク $G_v(\mathcal{N}_v, \mathcal{L}_v)$

より具体的には，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ は以下のような構造を持つネットワークである:

- ノード集合 $\mathcal{N}_v$ は $\mathcal{N}$ のコピー 4 つ $(\mathcal{O}, \mathcal{R}, \mathcal{S}, \mathcal{D})$ の和集合である．各ノードの番号は自身のコピー元のノード番号と同じとする．
- ノードは $|\mathcal{N}|$ 行 4 列の格子状に並んでいる．各列は左から順に $\mathcal{O}, \mathcal{R}, \mathcal{S}, \mathcal{D}$ に含まれる全てのノードが並んでいる．
- 1,2,3 列目の各ノードから，1 つ右隣の列の全てのノードに向かって有向リンクが存在する．このリンク全ての集合を $\mathcal{L}_v$ とする．
- 各リンク $ij \in \mathcal{L}_v$ のリンクコスト τ_{ij} は，以下のように設定する:

$$\begin{cases} \tau_{or} = t_{or} & \forall o \in \mathcal{O}, r \in \mathcal{R} \\ \tau_{rs} = t_{rs} - \bar{c}_{rs} & \forall r \in \mathcal{R}, s \in \mathcal{S} \\ \tau_{sd} = t_{sd} & \forall s \in \mathcal{S}, d \in \mathcal{D} \end{cases} \tag{29}$$

$G_v(\mathcal{N}_v, \mathcal{L}_v)$ の経路への交通配分がドライバーとタスクのマッチングを表していることは，以下のようにして確かめられる: $G_v(\mathcal{N}_v, \mathcal{L}_v)$ は，OD ペア $o \in \mathcal{O}, d \in \mathcal{D}$ の経路が必ず 3 本のリンクで構成されている．さらに経路 $o-r-s-d$ の交通費用が，OD ペア $o, d \in \mathcal{N}$ のドライバーがタスク $r, s \in \mathcal{N}$ を実行するときの観測社会的余剰 $-W^{rs}_{od}$ に一致する．すなわち，ドライバー 1 人を $G_v(\mathcal{N}_v, \mathcal{L}_v)$ の経路を通行する車両 1 台として表現すれば，OD ペア $o, d \in \mathcal{N}$ のあるドライバー a とタスク $r, s \in \mathcal{N}$ をマッチングした際の社会的余剰 $w^{rs}_a = W^{rs}_{od} + \epsilon^{rs}_a$ は，ドライバー a を表す車両 1 台を $G_v(\mathcal{N}_v, \mathcal{L}_v)$ 上の経路 $o-r-s-d$ に配分する際の認知経路費用と解釈できる．したがって，費用の観点から

見れば，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ の経路に車両 1 台を配分することは，1 人のドライバーとタスクをマッチングさせることと等価である．

(2) 交通量配分問題への変形および縮約

前節で示した $G_v(\mathcal{N}_v, \mathcal{L}_v)$ への交通配分と CSD システムにおけるマッチングの対応を利用すれば，[SO-A-P] は $G_v(\mathcal{N}_v, \mathcal{L}_v)$ 上の交通量配分問題とみなすことができる．[SO-A-P] は，“社会的最適マッチングにおいて，各タスク $r, s \in \mathcal{N}$ にマッチングされた OD ペア $o, d \in \mathcal{N}$ のドライバー数 f_{od}^{rs} を求める問題” である[6]．マッチングを $G_v(\mathcal{N}_v, \mathcal{L}_v)$ への交通配分に置き換えれば，これは “社会的最適交通配分において，各経路 $o-r-s-d$ に配分された OD ペア $o \in \mathcal{O}, d \in \mathcal{D}$ の車両数 f_{od}^{rs} を求める問題” と解釈できる．これはすなわち，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ 上の交通量配分問題である．

$G_v(\mathcal{N}_v, \mathcal{L}_v)$ 上での各車両の認知経路費用が観測経路費用 $-W_{od}^{rs}$ と Gumbel 分布に従う誤差項の和で表現される状況を考えているため，この交通量配分問題は LOGIT 配分問題である．したがって，[SO-A-P] は以下のような状況における $G_v(\mathcal{N}_v, \mathcal{L}_v)$ 上の LOGIT 配分問題と等価である．

- 全ての $o \in \mathcal{O}, d \in \mathcal{D}$ 間の OD フローは q_{od} であり，それ以外の OD ペアの OD フローは 0 である．
- 各車両 a は，経路 $o-r-s-d$ の経路費用を $\tau_{or} + \tau_{rs} + \tau_{sd} - \epsilon_a^{rs}$ と知覚する．ϵ_a^{rs} は私的効用値であり，1 に示す通りの性質をもつ．
- 全ての $(r, s) \in \mathcal{R} \times \mathcal{S}$ について，この 2 ノードを結ぶリンクに容量 n_{rs} が存在する．

さらに，LOGIT 配分問題は起点別リンクフローのみを用いた形式 (arc-node 形式) に等価変形することができる (証明は付録に示す)．具体的には，[SO-A-P] は以下の問題 [SO-L-P] に再定式化できる:

[SO-L-P]

$$\min_{\boldsymbol{x} \geq \boldsymbol{0}} \quad \sum_{o \in \mathcal{O}} \sum_{ij \in \mathcal{L}_v} \tau_{ij} x_{ij}^o - \sum_o H^o(\boldsymbol{x}^o) \tag{30}$$

$$\text{s.t.} \quad \sum_{o \in \mathcal{O}} x_{rs}^o = n_{rs} \qquad \forall r \in \mathcal{R}, s \in \mathcal{S} \tag{31}$$

$$\sum_{i \in NI_k} x_{ik}^o - \sum_{j \in NO_k} x_{kj}^o = \begin{cases} q_{od} & k = d \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in \{\mathcal{R}, \mathcal{S}, \mathcal{D}\} \tag{32}$$

$$\text{where} \quad H^o(\boldsymbol{x}^o) = -\frac{1}{\theta} \left(\sum_{ij \in \mathcal{L}_v} x_{ij}^o \ln \frac{x_{ij}^o}{\sum_{i \in NI_j} x_{ij}^o} \right) \tag{33}$$

ここで x_{ij}^o は，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ において起点が $o \in \mathcal{O}$ であるリンク $ij \in \mathcal{L}_v$ のフローを表す．また，$NI_k(NO_k)$ はノード $j \in \mathcal{N}_v$ に流入 (から流出) するリンクの下流 (上流) 側ノード集合である．

一方，[SO-A-D] は以下の [SO-L-D] に再定式化できる:

[SO-L-D]

$$\min_{\boldsymbol{u}} \ z_{LD}(\boldsymbol{v}) \equiv - \sum_{r \in \mathcal{R}, s \in \mathcal{S}} n_{rs} v_{rs} - \sum_{o \in \mathcal{O}} \sum_{d \in \mathcal{D}} q_{od} \mu_d^o(\boldsymbol{v}) \tag{34}$$

変数 v_{rs} はリンク rs に賦課される通行料金と解釈できる[7]．μ_k^o は，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ 上で，リンク rs に費用 $-v_{rs}$ が足されている状態における，起点 $o \in \mathcal{O}$ からノード $k \in \mathcal{N}_v$ までの期待最小費用を表す．具体的には，μ_k^o は以下のように定義される:

$$\mu_r^o = \tau_{or} \qquad \forall r \in \mathcal{R} \tag{35}$$

$$\mu_s^o(\boldsymbol{v}) = -\frac{1}{\theta} \ln \sum_{r \in \mathcal{R}} \exp[-\theta\{\tau_{or} + v_{rs} + \tau_{rs}\}] \qquad \forall s \in \mathcal{S} \tag{36}$$

$$\mu_d^o(\boldsymbol{v}) = -\frac{1}{\theta} \ln \sum_{s \in \mathcal{S}} \exp[-\theta\{\mu_s^o(\boldsymbol{v}) + \tau_{sd}\}] \qquad \forall d \in \mathcal{D} \tag{37}$$

このように問題を起点別リンク変数で表現することで，問題の規模を縮約できる．具体的には，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ の経路数は $|\mathcal{N}|^4$，起点別リンクフロー変数の数は $3|\mathcal{N}|^3$ であるから，起点別リンクフロー変数を用いて再定式化することにより，問題規模のオーダーを $1/|\mathcal{N}|$ にすることができる．

7. アルゴリズム

本章では，前章で変換されたマスター問題を解くアルゴリズムを提案する．提案アルゴリズムでは双対問題 [SO-L-D] を解き，その解を利用してマスター問題の解 (i.e., OD ペアへのタスク配分 $\boldsymbol{f}$) を得る．これは，

[6] 元の定義に近い形で言い換えれば，[SO-P/Sub] を解くのみで社会的最適マッチングを達成できるように各 OD ペアへのタスク配分 $\boldsymbol{f}^*$ を決める問題である．

[7] [SO-A-D] と [SO-L-D] が等価であることから，v_{rs} の最適値は元々の定義であるタスク実行による管理者の効用に一致する．

[SO-L-D] が無制約最適化問題であり，[SO-L-P] に比べて効率的に解くことができるからである．

[SO-L-D] を解くアルゴリズムは，Nesterov[10] が提案した加速勾配法を基に構築することが望ましい．加速勾配法は，1 次法として理論上最適なアルゴリズムである．Nesterov[10] は，目的関数と勾配の情報のみを用いたアルゴリズムの最悪収束率は，どれだけ小さくても $O(1/k^2)$ であること，および自身の提案した加速勾配法がその最悪収束率を達成できることを示した．この事実から，加速勾配法は最悪収束率の観点から 1 次法として理論上最適であることが示される．

Newton 法などの 2 次法ではなく 1 次法が望ましい理由は，[SO-L-D] は Hessian の計算量およびメモリ使用量が膨大であり，2 次法での求解が非効率的であるからである．[SO-L-D] の目的関数，勾配，Hessian は，要する計算量がそれぞれ $O(|\mathcal{N}|^3), O(|\mathcal{N}|^3), O(|\mathcal{N}|^5)$，値を保存するために必要なメモリ使用量は $O(1), O(|\mathcal{N}|^2), O(|\mathcal{N}|^4)$ である (これらの計算量の導出は付録に示す)．計算量，メモリ使用量ともに，Hessian は目的関数，勾配と比較して計算量オーダーが $|\mathcal{N}|^2$ 倍になる．したがって，CSD サービス規模が極端に小さくない限り，1 次法がより効率的であると判断できる．

(1) メインアルゴリズム

本研究では，加速勾配法の一種である FISTA[11] に，効率化手法である Adaptive restart scheme[12] を適用したアルゴリズムを基にする．具体的なアルゴリズムを Algorithm 1 に示す．Algorithm 1 中の関数 P, Q は以下のように定義される:

Algorithm 1 Main Algorithm

Require: $\epsilon > 0, L_0 > 0, \eta > 1$

Ensure: $\boldsymbol{v}^*$

$\boldsymbol{v}_0 \Leftarrow \mathbf{0}, \hat{\boldsymbol{v}}_0 \Leftarrow \mathbf{0}, l = 0, t_0 = 1$

while $\max_{rs} \left\{ \left| \frac{\partial z_D}{\partial v_{rs}}(\boldsymbol{v}_{l+1}) \right| \right\} > \epsilon$ **do**

$z_D(\boldsymbol{v}_l), \nabla z_D(\boldsymbol{v}_l) \Leftarrow LogitAssignment(\boldsymbol{v}_l)$

$\iota \Leftarrow \min_{\iota \in \mathbb{Z}_{\geq 0}} \{\iota \mid P(\boldsymbol{v}_l, L_l, \iota) \leq Q(\boldsymbol{v}_l, L_l, \iota)\}$

$L_{l+1} = \eta^\iota L_l$

if $\nabla z_D(\boldsymbol{v}_l) \cdot (\boldsymbol{v}_{l+1} - \boldsymbol{v}_l) > 0$ **then**

$t_{l+1} = 1$

else

$t_{l+1} = \frac{1+\sqrt{1+4{t_l}^2}}{2}$

end if

$\hat{\boldsymbol{v}}_l = \boldsymbol{v}_l - \frac{1}{L_{l+1}} \nabla z_D(\boldsymbol{v}_l)$

$\boldsymbol{v}_{l+1} = \hat{\boldsymbol{v}}_l + \frac{t_l - 1}{t_{l+1}} (\hat{\boldsymbol{v}}_l - \hat{\boldsymbol{v}}_{l-1})$

$l \Leftarrow l + 1$

end while

Return $\boldsymbol{v}^* = \boldsymbol{v}^l$

$$P(\boldsymbol{v}_l, L_l, \iota) \equiv z_D\Big(\boldsymbol{v}_l - \frac{1}{\eta^\iota L_l} \nabla z_D(\boldsymbol{v}_l)\Big) - z_D(\boldsymbol{v}_l) \quad (38)$$

$$Q(\boldsymbol{v}_l, L_l, \iota) \equiv -\frac{1}{2\eta^\iota L_l} ||\nabla z_D(\boldsymbol{v}_l)||^2 \quad (39)$$

Algorithm 1 中でもっとも計算負荷が大きいステップは，目的関数および勾配を計算するサブアルゴリズム $LogitAssignment$ である．したがって，このサブアルゴリズムを計算効率的にすることが重要である．そこで次節では，このサブアルゴリズムを解説する．

(2) 目的関数・勾配の計算アルゴリズム

[SO-L-D] の目的関数 z_{LD} およびその勾配 ∇z_{LD} の値は LOGIT 配分アルゴリズムによって計算することができる．まず，目的関数は (34)，その勾配は以下の (40) によって計算できる．

$$\frac{\partial z_{LD}(\boldsymbol{v})}{\partial v_{rs}} = -n_{rs} + \sum_o x_{rs}^o(\boldsymbol{v}) \qquad \forall r, s \quad (40)$$

ここで，$x_{rs}^o(\boldsymbol{v})$ は，リンク rs のコストを $\tau_{rs} - v_{rs}$ とした [VN] において LOGIT 配分を行った際の，起点が i であるリンク rs のフローである．目的関数および勾配の計算に必要な期待最小費用 μ_d^o および起点別リンクフロー x_{rs}^o は，リンク rs のコストを $\tau_{rs} - v_{rs}$ とした [VN] において容量制約のない LOGIT 配分を行うことで得ることができる．したがって，$z_{LD}, \nabla z_{LD}$ の値も LOGIT 配分アルゴリズムによって計算できる．

加えて，6 章で行った問題の縮約を活かす効率的な計算を行うためには，適用する LOGIT 配分アルゴリズムは経路列挙を必要としない LOGIT 配分アルゴリズム (e.g., Dial アルゴリズム，Markov 連鎖配分) でなければならない．[VN] 上での LOGIT 配分の場合，これらのアルゴリズムは Algorithm 2 に帰着する．Algorithm 2 中の p_{sd}^o, p_{rs}^o はそれぞれ，起点が o であるフローのうちリンク sd, rs を流れるものの割合であり，以下のように計算される:

$$p_{sd}^o = \exp[-\theta\{\mu_s^o(\boldsymbol{v}) + \tau_{sd} - \mu_d^o(\boldsymbol{v})\}] \quad (41)$$

$$p_{rs}^o = \exp[-\theta\{\tau_{or} + \tau_{rs} - v_{rs} - \mu_s^o(\boldsymbol{v})\}] \quad (42)$$

また，[SO-L-D] の最適解から主問題 [SO-A-P] の解 $\boldsymbol{f}^*$(i.e., 各 OD ペアへのタスク配分) を求める際にも，このサブアルゴリズムの結果を利用する．具体的には，Algorithm 1 の終了前最後の反復において Algorithm 2 で計算した p_{rs}^o, p_{sd}^o を用いて，以下のように計算できる．

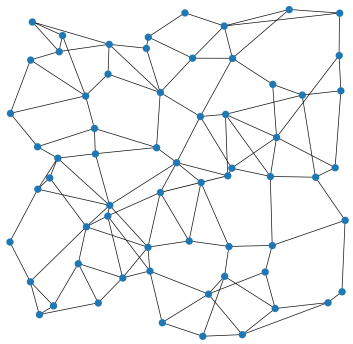
$$f_{od}^{rs*} = q_{od} p_{sd}^o p_{rs}^o \qquad \forall od, rs \quad (43)$$

8. 数値実験

本章では，6 章，7 章で提案した [SO-A-P] の解法の効率性を，数値実験を通して確認する．具体的には，[SO-

Algorithm 2 *LogitAssignment*

for $(s,o) \in \mathcal{O} \times \mathcal{S}$ **do**
  Calculate μ_s^o by (36)
end for
for $(d,o) \in \mathcal{D} \times \mathcal{O}$ **do**
  Calculate μ_d^o by (37)
end for
for $(s,d,o) \in \mathcal{S} \times \mathcal{D} \times \mathcal{O}$ **do**
  Calculate p_{sd}^o by (41)
end for
Calculate $z_{LD}(\boldsymbol{v})$ by (34)
for $(r,s,o) \in \mathcal{R} \times \mathcal{S} \times \mathcal{O}$ **do**
  Calculate p_{rs}^o by (42)
end for
for $(o,s) \in \mathcal{O} \times \mathcal{S}$ **do**
  $X_s^o \Leftarrow \sum_{d \in \mathcal{D}} q_{od} p_{sd}^o$
end for
for $(r,s) \in \mathcal{R} \times \mathcal{S}$ **do**
  $G_{rs} \Leftarrow -n_{rs} + \sum_{o \in \mathcal{O}} p_{rs}^o X_s^o$
end for
Return $\nabla z_{LD}(\boldsymbol{v}) = \{G_{rs}\}_{\forall rs},\ z_{LD}(\boldsymbol{v})$



図–3 ランダム・ネットワークの一例

A-P] を直接解く場合 (以下，直接解法) と，提案手法で解く場合のそれぞれについて，収束に要する CPU 時間を比較する．この比較を通して，提案手法が直接解法と比較して大幅に効率的であることを示す．

(1) 実験の概要

本実験では，配送タスクの起点となるノードが，いくつかの特定のノード (以下，タスク起点ノード) のみに限定された状況を想定する．これは，配送業務の起点となりうるのは，配送業者の集配所や小売店店舗など，限定された場所のみだからである．このため，本実験において，[SO-A-P] の問題規模はネットワークのノード数 $|\mathcal{N}|$ とタスク起点ノード数 $|\mathcal{R}|$ の二つによって決定される．

表–1 実験に用いるノード数，タスク起点数の設定

Case 1		Case 2	
$\|\mathcal{N}\|$	$\|\mathcal{R}\|$	$\|\mathcal{N}\|$	$\|\mathcal{R}\|$
49	8	64	2
81	8	64	4
121	8	64	8
196	8	64	16
289	8	64	32
400	8	64	64

問題規模を決定する 2 要素 $|\mathcal{N}|, |\mathcal{R}|$, がアルゴリズムの計算時間に与える影響を独立に調べるため，実験は $|\mathcal{N}|$ のみを変化させる Case 1 と，$|\mathcal{R}|$ のみを変化させる Case 2 に分けて行う．各 Case における具体的な $|\mathcal{N}|, |\mathcal{R}|$ の値を**表–1** に示す．1 つの $|\mathcal{N}|, |\mathcal{R}|$ の設定に対して，構造の異なる 10 のネットワークを用いて実験を行い，実験結果をその最小値，最大値，平均値によって示す．各ネットワークにおける実験結果は，q_{od}, n_{rs} の値を変えて 10 回計算を実行した平均値を用いる．

(2) 入力・実験環境

道路ネットワーク $G(\mathcal{N}, \mathcal{L})$ として用いるネットワークは，都市道路ネットワークに近い構造をもち，かつ形状および接続構造がランダムになるように設計されたものである．64 ノードのランダム・ネットワークの一例を**図–3** に示す．具体的な生成方法は付録に示す．タスク起点ノードの位置はランダムに設定した．

ドライバーの OD 需要 q_{od} は全てのノード間に，タスクの OD 需要 n_{rs} は各タスク起点ノードと全ノードの間に，それぞれランダムな整数値を設定した．ドライバーの総数は $10 \times$ (ドライバー OD ペア数) $= 10|\mathcal{N}|^2$ とした．

直接解法に用いるアルゴリズムは，[SO-A-P] のような制約付き凸計画問題に対して用いられる Frank-Wolfe 法とした．ただし，Frank-Wolfe 法全体の所要時間ではなく，Frank-Wolfe 法のプロセス中で解く必要のある以下のサブ問題 (i.e.Hitchcock 問題) を，LP ソルバーを用いて解く際の所要時間を比較対象とした．

$$\max_{\boldsymbol{f}} \sum_{od} \sum_{rs} A_{od}^{rs,n} f_{od}^{rs} \tag{44}$$

$$\text{s.t. } (10), (12), (19) \tag{45}$$

ここで，$A_{ij}^{rs,n}$ は Frank-Wolfe 法における n 反復目の暫定解 $\boldsymbol{f}^n$ を用いて以下のように計算される定数である:

$$A_{od}^{rs,n} = \left(W_{od}^{rs} - \frac{\partial H_{od}}{\partial f_{od}^{rs}}(\boldsymbol{f}^n) \right) \tag{46}$$

実験に使用した計算機は Ryzen 9 3950X 型 CPU，

RAM 64GB 搭載のものである．直接解法は MATLAB(R2019a) の LP ソルバー[8]で解いた結果，提案手法は Python 3.7 で実装し解いた結果を用いている．提案手法におけるアルゴリズム内の backtracking のパラメータ η は 1.5，LOGIT パラメータ θ は 5，収束判定定数 ϵ は 10^{-2} を用いた．

(3) 結果

Case 1 の実験結果を**図–4** に，Case 2 の実験結果を**図–5** に示す．Case 1 の直接解法の結果は，121 ノード以下のみ示している．これは，196 ノード以上は CSD システムとして実用的でないほど時間がかかると推測され，さらに 289 ノード以上はメモリ不足により実行不可能であったためである．

Case 1 の結果から，提案手法について以下の知見を得ることができる．

1-1 直接解法の所要時間を計測した範囲で，10 倍〜100 倍程度効率的である．

1-2 問題規模の増加に対する計算時間の増加が緩やかである．

1-3 ネットワーク構造の違いは計算時間に大きな影響を及ぼさない[9]．

知見 1-1 から，直接解法を計測した比較的小規模な問題のみに限っても，提案手法により計算時間が大幅に短縮されていることがわかる．さらに知見 1-2 から，直接解法を計測できなかった大規模問題に関しては，提案手法による計算時間の短縮はさらに大きくなると考えられる．また，知見 1-3 によって，提案手法の優位性がネットワーク構造によらない頑健なものであることも示すことができる．この頑健性を示す理由は以下のように説明することができる：提案手法はオリジナルネットワークではなく仮想ネットワーク $G_v(\mathcal{N}_v, \mathcal{L}_v)$ を用いる．$G_v(\mathcal{N}_v, \mathcal{L}_v)$ 構築時に利用する都市道路ネットワーク $G(\mathcal{N}, \mathcal{L})$ の情報は，ノードと各ノードペア間の最短経路費用のみであり，ネットワーク構造は含まれない．したがって，提案手法は $G(\mathcal{N}, \mathcal{L})$ のネットワーク構造を明示的には扱わず，計算時間がネットワーク構造に依存することはない．

一方，Case 2 の結果から，提案手法について以下の知見を得ることができる．

2-1 タスク起点数の増加に対して計算時間がほとんど増加しない．

この特徴は，6 章で行った問題の縮約によって得られたものである．このような特徴をもつ理由をより具体的に説明するため，縮約前の問題 [SO-A-D] の規模を決定する $G_v(\mathcal{N}_v, \mathcal{L}_v)$ の経路数，縮約後の問題 [SO-L-D] の問題規模を決定する $G_v(\mathcal{N}_v, \mathcal{L}_v)$ の起点別リンク変数の数が，それぞれタスク起点数 $|\mathcal{R}|$ の増加に対してどう増加するかを考える．まず，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ の経路数は $|\mathcal{N}|^3|\mathcal{R}|$ 個である．したがって，$|\mathcal{R}|$ が最小 ($|\mathcal{R}| = 1$) のとき，変数の数は $|\mathcal{N}|^3$ であり，$|\mathcal{R}|$ が最大 ($|\mathcal{R}| = |\mathcal{N}|$) のときは $|\mathcal{N}|^4$ 個となる．一方，起点別リンク変数の数は $2|\mathcal{N}|^2|\mathcal{R}| + |\mathcal{N}|^3$ である[10]．したがって，$|\mathcal{R}|$ が最小のとき変数の数は $|\mathcal{N}|^3 + 2|\mathcal{N}|^2$ であり，$|\mathcal{R}|$ が最大のときは $3|\mathcal{N}|^3$ 個となる．両者を比べると，経路数は $|\mathcal{R}|$ の増加によって最大 $|\mathcal{N}|$ 倍に増加するのに対し，起点別リンク変数はせいぜい 3 倍程度しか増加しないことがわかる．これが，知見 2-1 が得られる理由と推測される．

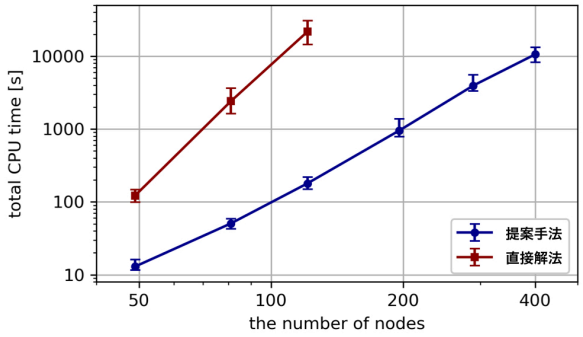
9. 終わりに

本研究ではまず，市場によりマッチングを決定する CSD システムを提案し，そのシステムが社会的余剰を最大にするマッチングを実現することを示した．具体的には，市場取引により実現する市場均衡状態におけるマッチングが，社会的余剰を最大化することを示した．この市場均衡状態は個々のドライバーの行動によって自律分散的に決定するため，中央集権的に決定する既存研究のマッチングとは異なり，提案システムは個々のドライバーの評価値を正確に反映した最適マッチングが実現可能である．

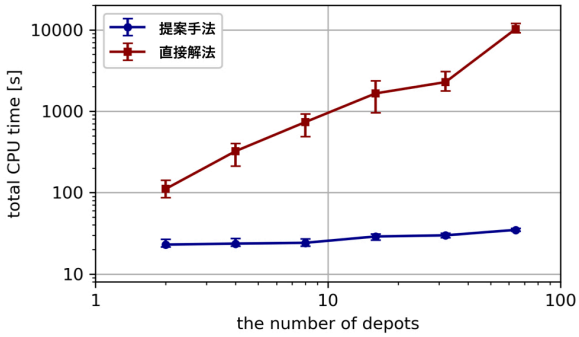
続いて，大規模な CSD システムに適用できるオークションメカニズムを提案した．このオークションメカニズムでは，市場を複数の小市場に分割した上で，(1) 各小市場への最適タスク配分，(2) 小市場別のオークションの 2 フェイズでマッチングを決定する．市場の分割によりオークションを行う個々の市場規模を小さくすることで，大規模な市場においてもオークションによる市場取引でマッチングを決定することを可能にした．

最後に，提案メカニズムのフェイズ 1 で解く最適タスク配分決定問題の効率的解法を提案した．具体的には，配分決定問題を交通ネットワーク配分問題に変換し，さらに arc-node 形式に変換することで縮約した上で，縮約された問題を解くアルゴリズムを構築した．数値実験により，提案解法は直接解法と比較して計算時間を数桁のオーダーで減少させていることに加え，計算時間の増加オーダーも大幅に減少させていることが

[8] linprog 関数，アルゴリズムは dual-simplex 法
[9] 様々な構造のランダム・ネットワークに対してほぼ同じ計算時間であったため．
[10] これは，$G_v(\mathcal{N}_v, \mathcal{L}_v)$ のリンク数が $\mathcal{O} - \mathcal{R}$ 間，$\mathcal{R} - \mathcal{S}$ 間にそれぞれ $|\mathcal{N}||\mathcal{R}|$ 本，$\mathcal{S} - \mathcal{D}$ 間に $|\mathcal{N}|^2$ 本であることからわかる．



図–4 Case 1



図–5 Case 2

示された．

本研究で得た成果は，今後物流需要が加速度的に増加していくであろうことを考えれば，CSD システム運用において必要不可欠なものである．e-commerce サービスの規模は日々拡大していることなどから，ラストワンマイル配送の需要が今後加速度的に増加していくことは容易に想像できる．そのため，CSD システムをラストワンマイル配送に導入する場合，システムが扱う配送件数は必然的に膨大になる．したがって，その膨大な配送タスクを扱うためには，本研究で提案した必要な最適マッチング決定問題の効率的解法が必要となる．

また，本研究の成果は，都市内のラストワンマイル配送に限らず，様々な規模の配送に対して同様に適用することができる．CSD システムが効率的である理由の一つは，配送業務とドライバーのマッチングを，多数の配送業者が独立に行うのではなく，1 つの管理主体が一元的に行うことができることである．このような，配送業務を一元管理することによる配送の効率化は，ラストワンマイル配送に限らず，あらゆる規模の配送業務に対して同様に行うことができる[11]．したがって，本研究で提案した CSD システムにおけるマッチング問題の解法を用いて，配送業務を一元管理するシステムを構築することで，あらゆる規模の配送業務を効率化することが可能である．

謝辞： 本研究は，日本学術振興会・科学研究費補助金・基盤研究 (B) (課題番号:18H01551) の助成を受けた研究の一部です．ここに記し，感謝を表します．

[11] 実際，このような配送業務の管理方法は “Physical Internet[13])” として提唱され，配送効率を劇的に改善する構想として期待されている．

付録 I 命題 1 の証明

KKT 条件より，ある解 $\boldsymbol{y}^*$ に関して以下の 3 つの条件を満たす制約条件 (5),(6) の Lagrange 乗数 $\boldsymbol{\rho}, \boldsymbol{v}$ が存在するならば，$\boldsymbol{y}^*$ は [SO-P] の最適解である．

$$\begin{cases} w_a^{rs} - v_{rs} = \rho_a & \text{if} \quad y_a^{rs*} > 0 \\ w_a^{rs} - v_{rs} \leq \rho_a & \text{if} \quad y_a^{rs*} = 0 \end{cases} \qquad \forall a, rs \qquad \text{(I.1)}$$

$$\sum_r \sum_s y_a^{rs*} = 1 \qquad \forall a \qquad \text{(I.2)}$$

$$\begin{cases} \sum_a y_a^{rs*} = n_{rs} & \text{if} \quad v_{rs} > 0 \\ \sum_a y_a^{rs*} \leq n_{rs} & \text{if} \quad v_{rs} = 0 \end{cases} \qquad \forall r, s \qquad \text{(I.3)}$$

$\bar{\boldsymbol{v}}, \boldsymbol{d}$ はそれぞれ制約条件 (5),(6) の Lagrange 乗数である．ここで，v_{rs} を管理者の効用 $\bar{c}_{rs} - u_{rs}$，ρ_a をドライバー a の市場均衡における効用とすると，条件 (I.1),(I.2),(I.3) は市場均衡条件 (1),(2),(3) に一致する．したがって，市場均衡状態におけるマッチング $\boldsymbol{y}$ には (I.1)-(I.3) を満たす Lagrange 乗数が存在するため，市場均衡状態は [SO-P] の最適解である．以上より，市場均衡状態において社会的余剰が最大化されることが示された．(証明終)

付録 II 補題 1 の証明

まず，[SO-P/Sub(od)] の双対問題 [SO-D/Sub(od)] を定式化する:

[SO-D/Sub(od)]

$$\min_{\boldsymbol{u} \geq \boldsymbol{0}} \sum_{rs} v_{rs} f_{od}^{rs} + \sum_a \max_{rs} \{w_a^{rs} - v_{rs}\} \qquad \text{(II.1)}$$

[SO-D/Sub(od)] の目的関数第 2 項は，OD ペア od のドライバーが十分多ければ以下のように集計化することができる:

$$\sum_a \max_{rs} \{w_a^{rs} - v_{rs}\} = q_{od} V_{od}(\boldsymbol{v}) \qquad \text{(II.2)}$$

ここで，$V_{od}(\boldsymbol{v})$ は OD ペア od のドライバーの期待最大効用である:

$$V_{od}(\boldsymbol{v}) = E[\max_{rs}\{w_a^{rs} - v_{rs}\}] \tag{II.3}$$

$$= \frac{1}{\theta}\ln\left[\sum_{rs}\exp[\theta\{W_{od}^{rs} - v_{rs}\}]\right] \tag{II.4}$$

$$\text{where}\quad W_{od}^{rs} = \bar{c}_{rs} - C_{od}^{rs} \tag{II.5}$$

したがって，[SO-D/Sub(od)] は集計化問題 [SO-A-D(od)] に帰着する:

[SO-A-D(od)]

$$\min_{\boldsymbol{u}\geq\boldsymbol{0}} \sum_{rs} v_{rs} f_{od}^{rs} + q_{od} V_{od}(\boldsymbol{v}) \tag{II.6}$$

続いて，$z_{od}^*(\boldsymbol{f})$ を，[SO-D/Sub(od)] の目的関数の最適値として表現する．強双対定理より，[SO-P/Sub(od)] の最適値 $z_{od}^*(\boldsymbol{f})$ は，その双対問題の最適値と一致する．すなわち，$z_{od}^*(\boldsymbol{f})$ は以下のように表現できる:

$$z_{od}^*(\boldsymbol{f}) = \min_{\boldsymbol{u}\geq\boldsymbol{0}} \left\{ \sum_{rs} v_{rs} f_{od}^{rs} + q_{od} V_{od}(\boldsymbol{v}) \right\} \tag{II.7}$$

以上より，補題 1 が示された．

付録 III　[SO-A-P],[SO-A-D] の arc-node 形式への変換

まず，[SO-A-P] と [SO-L-P] の等価性を示す．制約条件が等価であることはほぼ自明である．まず，制約条件 (19) はフロー保存則であり，これが起点別リンクフローで表現された条件 (32) と等価であることはよく知られている．また，リンク rs の容量制約を表す (10) も，リンクフローを用いて (31) のように表せることは明らかである．そのため，ここでは目的関数の等価性のみを示す．

目的関数の等価性は，以下のように示すことができる．[SO-A-P]，[SO-L-P] の目的関数第 1 項は，いずれもネットワーク上で費やされる観測可能な交通費用の総和を表している．第 2 項の等価性は，Akamatsu[14] により証明されている．以上より，目的関数の等価性が示された．したがって，[SO-A-P] と [SO-L-P] は等価である．

続いて，[SO-A-D] と [SO-L-D] の等価性を示す．そのためには，$-V_{od}(\boldsymbol{v})$ と $\mu_d^o(\boldsymbol{v})$ の等価性を示せば十分である．この等価性は以下のように示される:

$$\mu_d^o(\boldsymbol{v}) = -\frac{1}{\theta}\ln\sum_{s\in\mathcal{S}}\exp[-\theta(\mu_s^o(\boldsymbol{v}) + \tau_{sd})] \tag{III.1}$$

$$= -\frac{1}{\theta}\ln\left[\sum_{s\in\mathcal{S}}\Big\{\exp[-\theta\tau_{sd}] \tag{III.2}$$

$$\times\sum_{r\in\mathcal{R}}\exp[-\theta(\tau_{or} + v_{rs} + \tau_{rs})]\Big\}\right] \tag{III.3}$$

$$= -\frac{1}{\theta}\ln\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}}\exp[-\theta(-W_{od}^{rs} + v_{rs})] \tag{III.4}$$

$$= -V_{od}(\boldsymbol{v}) \tag{III.5}$$

したがって，[SO-A-D] と [SO-L-D] は等価である．

付録 IV　[SO-A-D] の Hessian の計算量

問題 [SO-A-D] の目的関数の Hessian $\boldsymbol{H}$ は以下のような $|\mathcal{R}||\mathcal{S}| \times |\mathcal{R}||\mathcal{S}|$ の行列である．

$$\boldsymbol{H} = \begin{bmatrix} h_{11}^{11} & \cdots & h_{rs}^{11} & \cdots & h_{RS}^{11} \\ \vdots & & & & \\ h_{11}^{r's'} & & \ddots & & \vdots \\ \vdots & & & & \\ h_{11}^{RS} & & \cdots & & h_{RS}^{RS} \end{bmatrix} \tag{IV.1}$$

$$h_{rs}^{r's'} = \begin{cases} \sum_{od} q_{od} p_{od}^{rs} p_{od}^{r's'} & \text{if } (r,s) \neq (r',s') \\ \sum_{od} q_{od} p_{od}^{rs}(1 - p_{od}^{rs}) & \text{otherwise} \end{cases} \tag{IV.2}$$

ただし，$R = |\mathcal{R}|, S = |\mathcal{S}|$ であり，$p_{od}^{rs} = f_{od}^{rs}/q_{od}$ である．

以下では，(IV.2) の場合分けごとに，全成分の計算を行った場合に要する計算量を示す．なお，計算は勾配の計算アルゴリズム 2 と同様，仮想ネットワーク G_v の構造を利用した効率的な方法によって行う．

まず $(r,s) \neq (r',s')$ のとき，$h_{rs}^{r's'}$ は以下のように変形される:

$$h_{rs}^{r's'} = \sum_{od} q_{od} p_{od}^{rs} p_{od}^{r's'} \tag{IV.3}$$

$$= \sum_{o} p_{rs}^{o} p_{r's'}^{o} \left(\sum_{d} q_{od} p_{sd}^{o} p_{s'd'}^{o} \right) \tag{IV.4}$$

ここで，p_{rs}^o, p_{sd}^o はそれぞれ (42), (41) によって計算される[12]．この変形から，括弧内と外を別々に計算する以下のアルゴリズムにより，G_v の構造を利用した効率的な計算ができる．このアルゴリズムでは，全ての (r,r',s,s') ペアに対して $|\mathcal{O}|$ 回の反復計算を要するた

[12] 実際に最適化アルゴリズム中で Hessian を使う場合，これらの値は勾配計算時にすでに計算されており，改めて計算する必要はない．

Algorithm 3 *NonDiagonal*

for $(s, s', o) \in \mathcal{S} \times \mathcal{S} \times \mathcal{O}$ **do**
  $X^o_{ss'} = 0$
  for $d \in \mathcal{D}$ **do**
    $X^o_{ss'} \Leftarrow X^o_{ss'} + p^o_{sj} p^o_{s'j'}$
  end for
end for
for $(r, r', s, s') \in \mathcal{R} \times \mathcal{R} \times \mathcal{S} \times \mathcal{S}$ **do**
  $h^{r's'}_{rs} = 0$
  for $o \in \mathcal{O}$ **do**
    $h^{r's'}_{rs} \Leftarrow h^{r's'}_{rs} + p^o_{rs} p^o_{r's'} X^o_{ss'}$
  end for
end for

め，計算量は $O(|\mathcal{N}|^5)$ である[13]．

対角成分 ($(r, s) = (r', s')$ のとき) は，(IV.2) に示す式通りに計算しても，計算量は $O(|\mathcal{N}|^4)$ である．以上から，Hessian の計算量は $O(|\mathcal{N}|^5)$ である．

付録 V　実験用ネットワークの生成方法

1. ノード数 $|\mathcal{N}|$ を定める．
2. 1 辺 $5\sqrt{|\mathcal{N}|}$ の正方形 2 次元空間を $\sqrt{|\mathcal{N}|} \times \sqrt{|\mathcal{N}|}$ の格子状に分割し，各格子上のランダムな位置に，ノードを一つずつ生成する．
3. 都市空間の中心に遠い方から順に，$\forall n \in \mathcal{N}$ に対して，以下の操作を行う．
   (a) n と接続されていないノードのうち，n とのユークリッド距離が最も近いノードとの間にリンクを張る．
   (b) リンクコストを，その 2 ノード間のユークリッド距離とする
4. 再び，都市空間の中心に遠いノードから順に，$\forall n \in \mathcal{N}$ に対して以下の操作を行う．
   - n に接続されていないノードのうち，n とのユークリッド距離が最も近い 4 つのノードを選ぶ．
   - 選ばれたノードのうち，n とのネットワーク上での距離が最も遠いノードとの間にリンクを張る．
   - リンクコストを，その 2 ノード間のユークリッド距離とする．

[13] 厳密には対角成分を計算しないぶん必要反復数は少ないが，対角成分は $|\mathcal{N}|^2$ 個であるから，対角成分の計算を省くことによる計算量減少は $O(|\mathcal{N}|^3)$ である．これは全体の計算量オーダに影響を及ぼさない．

参考文献

1) Archetti, C., Savelsbergh, M., and Speranza, M. G.: The vehicle routing problem with occasional drivers, *European Journal of Operational Research*, Vol.254, No.2, pp.472–480, 2016.
2) Arslan, A. M., Agatz, N., Kroon, L., and Zuidwijk, R.: Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers, *Transportation Science*, Vol.53, No.1, pp.222–235, 2018.
3) Yildiz, B. and Savelsbergh, M.: Service and capacity planning in crowd-sourced delivery, *Transportation Research Part C: Emerging Technologies*, Vol.100, pp.177–199, 2019.
4) Allahviranloo, M. and Baghestani, A.: A dynamic crowdshipping model and daily travel behavior, *Transportation Research Part E: Logistics and Transportation Review*, Vol.128, pp.175–190, 2019.
5) Wang, Y., Zhang, D., Liu, Q., Shen, F., and Lee, L. H.: Towards enhancing the last-mile delivery: An effective crowd-tasking model with scalable solutions, *Transportation Research Part E: Logistics and Transportation Review*, Vol.93, pp.279–293, 2016.
6) Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders, *The Journal of finance*, Vol.16, No.1, pp.8–37, 1961.
7) Clarke, E. H.: Multipart pricing of public goods, *Public choice*, Vol.11, No.1, pp.17–33, 1971.
8) Groves, T. et al.: Incentives in teams, *Econometrica*, Vol.41, No.4, pp.617–631, 1973.
9) 原祐輔, 赤松隆: Network GEV 型経路選択モデルを用いた確率的利用者均衡配分, 土木学会論文集 D3 (土木計画学), Vol.70, No.5, pp.I_611–I_620, 2014.
10) Nesterov, Y. E.: A method for solving the convex programming problem with convergence rate $O(1/k^2)$, *Dokl. akad. nauk Sssr*, Vol. 269, pp. 543–547, 1983.
11) Beck, A. and Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM journal on imaging sciences*, Vol.2, No.1, pp.183–202, 2009.
12) O'donoghue, B. and Candes, E.: Adaptive restart for accelerated gradient schemes, *Foundations of computational mathematics*, Vol.15, No.3, pp.715–732, 2015.
13) Montreuil, B.: Toward a physical internet: meeting the global logistics sustainability grand challenge, *Logistics Research*, Vol.3, No.2-3, pp.71–87, 2011.
14) Akamatsu, T.: Decomposition of path choice entropy in general transport networks, *Transportation Science*, Vol.31, No.4, pp.349–362, 1997.

(2020. 10. 2 受付)

AN AUCTION-BASED CROWDSOURCED-DELIVERY SYSTEM

Taiki WATANABE and Takashi AKAMATSU