Vol.2014-AL-150 No.7
2014/11/20

組み合わせ論的 MTS 問題

中薗 拓巳[1]　畑埜 晃平[2]　瀧本 英二[2]

概要：メトリカルタスクシステム（Metrical Task System）問題（以下，MTS 問題）とは，逐次的な意思決定過程をモデル化したもので，ある固定された距離空間 (C, δ) を決定空間とするプロトコルとして，次のように記述される．各時刻 $t = 1, 2, \ldots, T$ において，プレイヤーは損失関数 $l_t : C \to \mathbb{R}_+$ を観測したあとある決定 $c_t \in C$ を選択し，損失 $l_t(c_t) + \delta(c_t, c_{t-1})$ を被る．プレイヤーの目標は，累積損失に関する競合比を最小化することである．各時刻の計算時間が決定空間のサイズ $n = |C|$ に比例するアルゴリズムがいくつか提案されている．しかし，ルーティングやランキングなど多くの問題は，決定空間が組み合わせ論的に定義された指数サイズの問題として定式化されるため，従来手法はそのままでは効率が悪い．例えばルーティング問題では，ある固定されたグラフの全域木の集合が決定空間になるが，そのサイズ n は一般にグラフのサイズに関して指数的である．本研究では，$c \neq c'$ のとき $\delta(c, c') = 1$ を満たす一様距離空間上の MTS 問題を，決定空間 C 上のサンプリングの問題に還元する手法を提案する．すなわち，効率の良いサンプリングアルゴリズムが存在する決定空間 C に対して，MTS 問題を効率よく解くことができる．実際，全域木の集合や s-t 道の集合など，いくつかの組み合わせ論的な決定空間は，効率の良いサンプリングアルゴリズムを持つことを示す．本手法の競合比の上界は $6e \ln n$ で，従来手法の競合比 $O(\log n)$ と同じオーダーを達成している．

キーワード：重み付きマーキングアルゴリズム　一様距離空間　メトリカルタスクシステム

1. はじめに

固定されたネットワーク上の 2 点間のルーティング問題について考える．各時刻において以下の試行を繰り返す．各リンクの混雑度（重み）を観測してから一つの s-t パスを選択する．その後，損失として選択したパスの重みの合計（処理コスト）とパスの選択を変更したときにかかるコスト（移動コスト）の和を被り，次の時刻へ移る．

このように逐次的な意思決定過程をモデル化した問題を Metrical Task System 問題 (MTS 問題) と呼び，次のようなプロトコルとして記述される．

決定空間を $\boldsymbol{C}$ とし，$\delta : \boldsymbol{C} \times \boldsymbol{C} \to \mathbb{R}_+$ を $\boldsymbol{C}$ 上の距離とする．各時刻 $t = 1, 2, \cdots, T$ において，以下を繰り返す．

(1) プレイヤーは損失関数 $\boldsymbol{l}_t : \boldsymbol{C} \to \mathbb{R}_+$ を観測する．
(2) プレイヤーは決定ベクトル $\boldsymbol{c}_t \in \boldsymbol{C}$ を出力する．
(3) プレイヤーは損失 $\boldsymbol{l}_t(\boldsymbol{c}_t) + \delta(\boldsymbol{c}_t, \boldsymbol{c}_{t-1})$ を被る．

プレイヤーの目標は累積損失

$$\sum_{t=1}^{T} \boldsymbol{l}_t(\boldsymbol{c}_t) + \delta(\boldsymbol{c}_t, \boldsymbol{c}_{t-1})$$

を最小化することである．損失の式の第一項を処理コスト，第二項を移動コストという．

最初に示した s-t パス問題の例では決定空間 $\boldsymbol{C} = \{\text{s-t パス}\} \subset \mathbb{R}_+^d$ となる．ただし，d はリンクの数である．すべてのパスは d 次元の $0-1$ ベクトルで表現できる．するとパス $\boldsymbol{c}_t \in \boldsymbol{C}$ と重み $\boldsymbol{l}_t \in [0,1]^d$ に対する処理コストは以下のように定義できる．

$$l_t(\boldsymbol{c}_t) = \boldsymbol{c}_t \cdot \boldsymbol{l}_t.$$

このような MTS 問題を特に組み合わせ論的 MTS 問題と呼ぶ．

この問題では，様々な組み合わせ論的最適化問題を MTS 問題として考えることができる．最初に示した例は，s-t パスにおける MTS 問題である．この決定空間はグラフ上の固定された 2 頂点間のパスの集合となるが，この決定空間を 1 から d までの順列の集合とすると，リストアクセス問題やランキング問題として考えることができる．また，決定空間を d 本の辺からなるグラフにおける全域木の集合とすると，この問題はスパニングツリー問題として考えられる．しかし，s-t パスの問題でも分かるように組み合わせ論的 MTS 問題の決定空間の数 $n(= |C|)$ は次元数 d に対して指数的に大きい場合がある．つまり，アルゴリズムの計

[1] 九州大学システム情報科学研究府情報学専攻
[2] 九州大学システム情報科学研究院情報学部門

算時間が $O(n)$ の場合，これらの問題の計算時間は指数時間かかることになる．したがって，組みあわせ論的 MTS 問題では，一般的な MTS 問題のように累積損失を少なくすることを目指すだけでなく，計算時間も考慮したアルゴリズム設計が必要になる．

2. 準備

2.1 前提知識

決定空間 $\boldsymbol{C} \subset \mathbb{R}_+^d$ とし，距離 $\delta : \boldsymbol{C} \times \boldsymbol{C} \to \mathbb{R}_+$ とする．アルゴリズム A に損失ベクトル系列 $(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T) \in ([0,1]^d)^T$ を与えたときに返す決定ベクトル系列を $(\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_T \in \mathbb{R}_+^d)$ とする．このときアルゴリズムが被る累積損失は処理コストと移動コストのすべての時刻の合計で表され，以下のようになる．

$$\mathrm{cost}_A(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T) := \sum_{t=1}^{T} (\boldsymbol{c}_t \cdot \boldsymbol{l}_t + \delta(\boldsymbol{c}_t, \boldsymbol{c}_{t-1})).$$

すべての損失ベクトル系列を知った上で求められる，累積損失が最小になる戦略をオフライン最適解と呼ぶ．つまり，損失ベクトル系列 $(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T)$ におけるオフライン最適解 $\mathrm{OPT}(\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_T)$ は以下が成り立つ．

$$\begin{aligned} &\mathrm{cost}_{\mathrm{OPT}}(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T) \\ :=& \min_{(\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_T) \in \boldsymbol{C}^T} \sum_{t=1}^{T} (\boldsymbol{c}_t \cdot \boldsymbol{l}_t + \delta(\boldsymbol{c}_t, \boldsymbol{c}_{t-1})). \end{aligned}$$

アルゴリズムの性能は競合比 (Competitive Ratio:CR) と呼ばれるアルゴリズムとオフライン最適解の累積損失の比で測る．つまり，アルゴリズム A の競合比が C 以下であるとは任意の T, 任意の系列 $(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T)$ に対して以下が成り立つことである．

$$C \leq \frac{\mathrm{cost}_A(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T)}{\mathrm{cost}_{\mathrm{OPT}}(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T)}.$$

MTS 問題のアルゴリズムの一般的な性質を示すために，以下を定義する．

定義 1. $(\boldsymbol{l}'_1, \boldsymbol{l}'_2, \cdots, \boldsymbol{l}'_{T'})$ が $(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T)$ の細分であるとは以下が成り立つような $(1, 2, \cdots, T')$ の部分区間 $(I_1, I_2, \cdots, I_T)$ が存在することである．

(1) $I_i \cup I_j = \phi$
(2) $\bigcup_{t=1}^{T} I_t = 1, \cdots, T'$
(3) $t \geq T, \sum_{s \in I_t} \boldsymbol{l}'_s = \boldsymbol{l}_t$

すると以下の補題が成り立つ．

補題 1. 任意のアルゴリズム A 任意の時刻 T, 任意の損失ベクトル系列 $(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T)$ とその任意の細分 $(\boldsymbol{l}'_1, \boldsymbol{l}'_2, \cdots, \boldsymbol{l}'_{T'})$ において以下の関係が成り立つ．

$$\mathrm{cost}_A(\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_T) \leq \mathrm{cost}_A(\boldsymbol{l}'_1, \boldsymbol{l}'_2, \cdots, \boldsymbol{l}'_{T'}).$$

証明の詳細は省略する．詳しくは [1] を参照してほしい．この補題を用いるとアルゴリズムは各時刻において損失ベクトルを都合の良い大きさに任意に分解しても一般性を失わないことがわかる．

距離空間の中でも異なる要素間の距離が 1 , そうでない場合は 0 になるような距離空間を特に一様距離空間 (uniform metric space) と呼ぶ．つまり δ が一様距離空間であるとは $\forall \boldsymbol{c}, \boldsymbol{c}' \in \boldsymbol{C}$ に対して

$$\delta(\boldsymbol{c}, \boldsymbol{c}') = \begin{cases} 1 & (\boldsymbol{c} \neq \boldsymbol{c}') \\ 0 & (\boldsymbol{c} = \boldsymbol{c}') \end{cases}$$

を満たすことをいう．本稿ではこの一様距離空間における MTS 問題について考える．

2.2 先行研究

一様距離空間における MTS 問題に関する研究の代表的なものとして，一様サンプリングに基づくマーキングアルゴリズム [2], ワークファンクションアルゴリズムを用いた Regularization アルゴリズム [3] などがあげられる．これらを含む過去に提案された全てのアルゴリズムの競合比は $O(\log |C|) = O(d)$ を達成させることができる．しかし，これらをそのまま用いると計算時間が d に対して指数時間かかってしまう．

Bansal らは組み合わせ論的 MTS 問題の一つである，k-サーバー問題に対して効率的に問題を解く手法を提案した [5]．また，Blum らは同様にリストアクセス問題に対して効率的に解く手法を提案した [6]．しかし，これらは解ける問題クラスが限定的なため，拡張性に乏しい．

Buchbinder らは 2012 年に一般的な組み合わせ論的 MTS 問題に対して，決定空間 C の凸包内の点を用いて，統一的にかつ，効率的に計算を行う手法を提案した [4]．しかし，アルゴリズムは凸包内部の点を出力するに留まっており，凸包内の点をうまく決定空間 C 上の点に置き換える手法 (ラウンディング) は明記されていない．また，移動コストの定義は移動前と移動後の凸包内の点の距離としている．これも決定空間 $\boldsymbol{C}$ 上の距離との対応付けが記されておらず，移動コストの定義として正当なものか不明確である．この手法はオンライン凸最適化と呼ばれる問題で用いられた手法を利用したものである．しかし，この手法を用いると前述のラウンディングの問題に直面してしまう．

そこで我々は既存手法の中でもラウンディングの問題を考えなくて良いマーキングアルゴリズム (Marking Algorithm) に着目し，それを改良したアルゴリズムを提案する．

2.3 研究成果

我々が提案するアルゴリズムは重み付きマーキングアルゴリズム (Weited Marking Algorithm) と呼ぶ．名前の

通り，マーキングアルゴリズムを改良したアルゴリズムである．競合比はマーキングアルゴリズムが $2\ln n+1$ に対し，重み付きマーキングアルゴリズムは $6e\ln n$ と従来手法に比べ多少劣るが，従来手法のアルゴリズムと同じ $O(\log|C|)=O(d)$ を満たす．そして問題を効率的に解く為の鍵としてそれぞれのアルゴリズムがサンプリング問題を抱えている．このサンプリング問題が我々の手法のほうが既存手法より緩和された問題であると考えている．以下に示すサンプリング問題が多項式時間で解けるかが，その問題クラスを効率的に解けるかどうかの判断基準になる．

まず，マーキングアルゴリズムは，

問題 1.
入力 ： $\boldsymbol{l}\in\mathbb{R}^d$
出力 ： $\boldsymbol{c}\in\boldsymbol{C}$ ただし $\boldsymbol{C_l}=\{\boldsymbol{c}\in C\mid\boldsymbol{c}\cdot\boldsymbol{l}\leq 1\}$ とし，
$\boldsymbol{c}$ は $\boldsymbol{C_l}$ から一様ランダムに選ばれる．
である．

対して, 重み付きマーキングアルゴリズムは，

問題 2.
入力 ： $\boldsymbol{l}\in\mathbb{R}^d$
出力 ： $\boldsymbol{c}\in\boldsymbol{C}$ ただし $\boldsymbol{c}$ は $\exp(-\eta\boldsymbol{c}\cdot\boldsymbol{l})$ に，
比例する確率でランダムに選ばれる．
である．

我々はマーキングアルゴリズムのサンプリング問題は $\#P$ 困難であると予想している．また，近似サンプリングが簡単にできるかどうか考慮する必要がある．対して，重み付きマーキングアルゴリズムのサンプリング問題はオンライン凸最適化問題に用いられる手法の一つである Hedge アルゴリズム [7] が効率的に解ける問題ならば有用である (例．s-t パスなど)．さらに，Bianchi らはこのオンライン凸最適化問題に対し，問題 2. のサンプリング手法を用いたアルゴリズムを提案している [8]．これによると完全マッチング問題，スパニングツリー問題，カットセット問題，ハミルトン閉路問題などの問題クラスが解けるとしている．

3. アルゴリズムの紹介

本稿ではアルゴリズム設計の元となったマーキングアルゴリズムと提案アルゴリズムである重み付きマーキングアルゴリズムを紹介する．

3.1 マーキングアルゴリズム

区間 $[1,T]$ をフェイズと呼ばれる部分区間に分解する．本手法はフェイズごとに累積コストベクトル $\boldsymbol{L}_t$ を保持し，その値に応じて戦略を決定する．フェイズごとに以下を繰り返す．$\boldsymbol{L}_t$ を 0 にリセットし，一様ランダムに決定ベクトル $\boldsymbol{c}\in\boldsymbol{C}$ を一つ選ぶ．その後現在の決定に対する累積損失 $\boldsymbol{c}\cdot\boldsymbol{L}_t$ が 1 に達するまで時刻が進んでも，決定ベクトルは変更しない．現在の決定に対する累積損失が 1 に達したとき，アルゴリズムは累積損失が 1 に達していない決定ベクトルの集合 $\boldsymbol{C}_{\boldsymbol{L}_t}=\{\boldsymbol{c}\in\boldsymbol{C}\mid\boldsymbol{c}\cdot\boldsymbol{L}_t<1\}$ の中から一様ランダムに選ぶ．これを繰り返し $C_{\boldsymbol{L}_t}=\phi$ となる時刻 t で次のフェイズに移る．

現在の決定ベクトルの累積損失が 1 に達する，もしくはフェイズが終了することで，アルゴリズムが前の時刻と異なる決定ベクトルを選択することを遷移と呼ぶ．

Algorithm 1 マーキングアルゴリズム
(1) 入力 $\eta>0$ ，初期化 $t=0$
(2) 初期化 $\boldsymbol{L}_t=\boldsymbol{0}$.　　//フェイズの開始
(3) 一様ランダムに選択 $\boldsymbol{c}\in\boldsymbol{C}$.
(4) Repeat
　(a) 更新 $t=t+1$.
　(b) 損失ベクトルの確認 $\boldsymbol{l}_t\in[0,1]^d$.
　(c) 出力 $\boldsymbol{c}_t=\boldsymbol{c}$.　　//$\boldsymbol{c}$ を選び続ける
　(d) 更新 $\boldsymbol{L}=\boldsymbol{L}+\boldsymbol{l}_t$.
　Until $\boldsymbol{c}\cdot\boldsymbol{L}_t=1$ or $\min_{\boldsymbol{c}^*\in\boldsymbol{C}}\boldsymbol{c}^*\cdot\boldsymbol{L}_t=1$.
(5) $\min_{\boldsymbol{c}^*\in\boldsymbol{C}}\boldsymbol{c}^*\cdot\boldsymbol{L}_t=1$ ならば，(2) へ.　　//フェイズの終了
(6) 一様ランダムに選択 $\boldsymbol{c}\in\{\boldsymbol{c}\in\boldsymbol{C}|\boldsymbol{c}\cdot\boldsymbol{L}_t<1\}$，かつ (4) へ.
.

このプロトコルの (4) の部分に注目していただきたい．繰り返しを抜ける条件として，現在決定のにおける累積損失 $\boldsymbol{c}\cdot\boldsymbol{L}_t$ がちょうど 1 になる時刻 t で遷移することになっている．実際の環境では，このようにある時刻 t でちょうど $\boldsymbol{c}\cdot\boldsymbol{L}_t=1$ となるようなコストベクトル $\boldsymbol{l}_t$ が観測できることは少なく，1 を超えてしまう場合がほとんどである．しかし，本稿では補題 1. を利用して，細分化された環境でアルゴリズムを動かしていると仮定してプロトコルを簡略化している．

実際の環境ではアルゴリズムが現在の戦略 $\boldsymbol{c}\in\boldsymbol{C}$ に対し，$\boldsymbol{c}\cdot\boldsymbol{L}_{t-1}<1$ かつ $\boldsymbol{c}\cdot(\boldsymbol{L}_{t-1}+\boldsymbol{l}_t)>1$ となるようなコストベクトル $\boldsymbol{l}_t$ を観測した場合，以下のような 2 つのコストベクトル $\boldsymbol{l}'_{t'},\boldsymbol{l}''_{t''}$ に分解し，その時刻 t を仮想的な時刻 t' と t'' の二つに分けて処理する．

$$\boldsymbol{l}'_{t'}\ \text{s.t}\ \boldsymbol{c}\cdot(\boldsymbol{L}_t+\boldsymbol{l}'_{t'})=1$$
$$\boldsymbol{l}''_{t''}:=\boldsymbol{l}_t-\boldsymbol{l}'_{t'}\ .$$

$\min_{\boldsymbol{c}^*\in\boldsymbol{C}}\boldsymbol{c}^*\cdot\boldsymbol{L}_t=1$ となる時刻 t も同様にして，時刻を細分化する．

アルゴリズムの計算時間について，問題 1. でも示したように，(6) の状態遷移時のサンプリング問題が大きな鍵である．その他の部分に関しては簡単に多項式時間でアルゴリズムを動かすことができる．(4) の第二条件, および (5) の問題は，オフライン最適化問題であるため，解ける問題クラスは多く存在する．

このアルゴリズムの競合比は $2H_n(<2\ln n+1)$ となる [2]．ただし H_n は n 項の調和級数である．

3.2 重みつきマーキングアルゴリズム

マーキングアルゴリズムと同様，区間 $[1,T]$ をフェイ

ズと呼ばれる部分区間に分解する．また，フェイズごとに累積損失ベクトル $\boldsymbol{L}_t$ を保持し，その値に応じて戦略を決定する．マーキングアルゴリズムと違う点は，まず，L s.t. $ne^{-\eta L} \leq e^{-\eta}$ となる累積損失の上限 L を設定し，決定ベクトルを変更するタイミングが $\boldsymbol{c}\cdot\boldsymbol{L}_t = L$ となる時刻 t となっていることである．また遷移先の決定ベクトルの決定を累積損失したがって確率的に選ぶようになっているという点も相違点である．. $\boldsymbol{c}_t\cdot\boldsymbol{L}_t$ が少ないものから選ばれやすくなるように設計してある．

マーキングアルゴリズムのときと同様，一般性を失わず，細分化された環境でアルゴリズムが動いているとする．

Algorithm 2 重みつきマーキングアルゴリズム

(1) 入力 $\eta > 0, L$ s.t. $ne^{-\eta L} \leq e^{-\eta}$, 初期化 $t = 0$.
(2) 初期化 $\boldsymbol{L}_t = \boldsymbol{0}$.　　//フェイズの開始
(3) 一様ランダムに選択 $\boldsymbol{c} \in \boldsymbol{C}$.
(4) Repeat
　(a) 更新 $t = t + 1$.
　(b) 損失ベクトルの確認 $\boldsymbol{l}_t \in [0,1]^d$.
　(c) 出力 $\boldsymbol{c}_t = \boldsymbol{c}$.　// $\boldsymbol{c}$ を選び続ける
　(d) 更新 $\boldsymbol{L} = \boldsymbol{L} + \boldsymbol{l}_t$
　Until $\boldsymbol{c}\cdot\boldsymbol{L}_t = L$, もしくは $\min_{\boldsymbol{c}^*\in\boldsymbol{C}} \boldsymbol{c}^*\cdot\boldsymbol{L}_t = 1$.
(5) $\min_{\boldsymbol{c}^*\in\boldsymbol{C}} \boldsymbol{c}^*\cdot\boldsymbol{L}_t = 1$ ならば，(2) へ．　//フェイズの終了
(6) Repeat
　(a) $\exp(-\eta\boldsymbol{c}\cdot\boldsymbol{L}_t)$ に比例する確率で $\boldsymbol{c}$ を選択.
　Until $\boldsymbol{c}\cdot\boldsymbol{L}_t < L$.　(4) へ.

重み付きマーキングアルゴリズムを動かしたときの計算時間について考える．ボトルネックとなるのは決定ベクトル変更の際のサンプリング問題である．ここで，プロトコルの (6) に注目する．この部分が解いている問題は $\boldsymbol{C}_{\boldsymbol{l}} = \{\boldsymbol{c} \in C | \boldsymbol{c}\cdot\boldsymbol{l} \leq L\}$ として以下のような問題である．

問題 3.
入力　：　$\boldsymbol{l} \in \mathbb{R}^d$
出力　：　$\boldsymbol{c} \in \boldsymbol{C}_{\boldsymbol{l}}$, ただし $\boldsymbol{c}$ は $\exp(-\eta\boldsymbol{c}\cdot\boldsymbol{l})$ に，
　　比例する確率でランダムに選ばれる．

マーキングアルゴリズムのように累積損失が大きい決定を排除するため，$\boldsymbol{c}\cdot\boldsymbol{L}_t < L$. となる $\boldsymbol{c}$ を選択するまで，サンプリングを繰り返している．しかし，この繰り返し回数は一回の遷移あたり，高々 2 回程度である．これは，決定ベクトル $\boldsymbol{c}$ を $\exp^{-\eta\boldsymbol{c}\cdot\boldsymbol{L}_t}$ に比例する確率で選択させていることと，累積損失の上限 L を $ne^{-\eta L} \leq e^{-\eta}$ となるように設定していることで達成できている．累積損失が大きい決定ほど選ばれなくすることで，累積損失の上限を超えた決定，つまり $\boldsymbol{c}\cdot\boldsymbol{L}_t \geq L$ となる $\boldsymbol{c}$ が選ばれる確率を 1/2 以下に抑えている．

このようにしたことで，マーキングアルゴリズムが抱える，決定ベクトル排除の問題をクリアすることができ，問題 2. を多項式時間で解ける問題に対して，効率的に解くことができる．

重み付きマーキングアルゴリズムの性能について考える．

定理 1. 重み付きマーキングアルゴリズムの競合比は $O(\log n)$ である．

一般性を失わず以下の条件が成り立つとする．

各フェイズにおいて
(1) $\boldsymbol{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_n\}$.
　ただし t_i, s.t. $\min_{t\in[1,T]}\{\boldsymbol{c}_i\cdot\boldsymbol{L}_t \geq L\}$ に対し
　$t_1 \leq t_2 \leq \cdots \leq t_n$.
(2) $\exists\hat{T}$ s.t. $\min_{\boldsymbol{c}^*\in\boldsymbol{C}} \boldsymbol{c}^*\cdot\boldsymbol{L}_{\hat{T}} = 1$.
(3) $\boldsymbol{c}_i\cdot\boldsymbol{L}_{\hat{T}} > L$ となる全ての i に対し
　$\exists t_i$ s.t. $\boldsymbol{c}_i\cdot\boldsymbol{L}_{t_i} = L$.

(2),(3) の条件は補題 1. を利用して，解析のために都合の良いように損失ベクトルを細分化した環境で考えるということを示している．

まず，1 フェイズあたりにオフライン最適解が被る累積損失を求める．フェイズごとに以下の補題が成り立つ．

補題 2. それぞれのフェイズでオフライン最適解が被る累積損失は少なくとも 1 である．

証明. 2 つの場合に分けて考える．(i) オフライン最適解がフェイズ中に一度でも決定ベクトルを変更した場合，移動コストを 1 被る．(ii) オフライン最適解がフェイズ中に決定ベクトルを変更しなかった場合，フェイズの定義から，1 フェイズ毎に被る処理コストは少なくとも 1. □

以上よりオフライン最適解が被る累積損失の下界が求まった．よって競合比を求めるためにはアルゴリズムが被る累積損失の上界を求めればよい．そのためにこれからいくつかの準備をする．

m 番目の要素 $\boldsymbol{c}_m$ が最も遅く累積損失が 1 に達するとする．つまり以下が成り立つような m を定義する．

$$m \text{ s.t. } \min_{1\leq m\leq n} \boldsymbol{c}_m\cdot\boldsymbol{L}_{\hat{T}} = 1.$$

すると条件 (1) より $i \in [1, m-1]$ 番目の要素 $\boldsymbol{c}_i$ は $\boldsymbol{c}_i\cdot\boldsymbol{L}_{t_i} = L$ となる時刻 t_i が小さい順にならんでおり，$i' \in [m, n]$ 番目の決定ベクトル $\boldsymbol{c}_{i'}$ は累積損失が L に達することなくフェイズを終える．つまり，アルゴリズムがこれらの決定ベクトルのどれか一つを選ぶと，以降遷移が起こることなくフェイズを終了する．この決定空間 $\boldsymbol{C}_{end} = \{\boldsymbol{c} \in \boldsymbol{C} | \boldsymbol{c}\cdot\boldsymbol{L}_{\hat{T}} \leq L\}$ をエンドフェイズ集合と呼ぶ．

さらに各フェイズにおいて以下を定義する．

アルゴリズムが損失ベクトル $\boldsymbol{l}_t$ を観測した直後（決定ベクトルを選択する直前）の時刻 t に対し，重みベクトル $\boldsymbol{W}_t$ を次のように定義する．

$$W_{t,i} = \begin{cases} e^{-\eta\boldsymbol{c}_i\cdot\boldsymbol{L}_t} & (\boldsymbol{c}_i\cdot\boldsymbol{L}_t < L) \\ 0 & (\boldsymbol{c}_i\cdot\boldsymbol{L}_t \geq L) \end{cases}.$$

$\boldsymbol{W}_t$ は t に関して単調減少関数である．

また，k 回目の遷移が行われる時刻を τ_k，フェイズ内で

起こった遷移の回数を u とする．

τ_k を用いて時刻をあらわすと $1, 2, \cdots, u$ に対するそれぞれの遷移時の時刻は $\tau_1, \tau_2, \cdots, \tau_u$ となる．特に τ_1 はフェイズの初めの一様ランダムに遷移する時刻になる．

これらを用いて以下の補題が成り立つことを示す．

補題 3. 任意の $\alpha \in [0,1]$, 任意の k に対し， $W_{\tau_k,m} \leq \alpha||\boldsymbol{W}_{\tau_k}||_1$ ならば，

$$\Pr[||\boldsymbol{W}_{\tau_{k+1}}||_1 \leq \alpha||\boldsymbol{W}_{\tau_k}||_1] > \alpha.$$

証明.
i_k を

$$\sum_{i=i_k+1}^{n} W_{\tau_k,i} \leq \alpha||\boldsymbol{W}_{\tau_k}||_1, \text{ and } \sum_{i=i_k}^{n} W_{\tau_k,i} > \alpha||\boldsymbol{W}_{\tau_k}||_1$$

を満たす自然数とする．このような i_k が存在するのは補題 3. の条件が成り立つときのみである．

アルゴリズムが i_k 番目以降の決定ベクトルを選ぶ確率は

$$\Pr[\boldsymbol{c}_i(i \geq i_k) \text{ を選ぶ}] = \frac{\sum_{i=i_k}^{n} W_{\tau_k,i}}{||\boldsymbol{W}_{\tau_k}||_1} > \frac{\alpha||\boldsymbol{W}_{\tau_k}||_1}{||\boldsymbol{W}_{\tau_k}||_1} = \alpha$$

となる．以下に

$$\Pr[||\boldsymbol{W}_{\tau_{k+1}}||_1 \leq \alpha||\boldsymbol{W}_{\tau_k}||_1] = \Pr[\boldsymbol{c}_i(i \geq i_k) \text{ を選ぶ}]$$

を示す．

アルゴリズムが時刻 τ_k で決定ベクトル $\boldsymbol{c}_i$ を選んだとする．すると，次の遷移時刻 τ_{k+1} において $\forall j \leq i$ に対し，$W_{\tau_{k+1},j} = 0$ となる．
このことに注目するとアルゴリズムが i_k 番目以降の決定ベクトルを選ぶとき，

$$\begin{aligned} ||\boldsymbol{W}_{\tau_{k+1}}||_1 &\leq \textstyle\sum_{i=i_k+1}^{n} W_{\tau_{k+1},i} \\ &= \textstyle\sum_{i=i_k+1}^{n} W_{\tau_k,i} \quad \text{（単調減少）} \\ &\leq \alpha||\boldsymbol{W}_{\tau_k}||_1 \end{aligned}$$

よって成り立つ． □

フェイズにおける遷移回数 k , 確率 $\alpha \in [0,1]$ に対し，以下を定義する．

$$\Delta_{k,\alpha} = \min\{\Delta \mid ||\boldsymbol{W}_{\tau_{k+\Delta}}||_1 \leq \alpha||\boldsymbol{W}_{\tau_k}||_1 \text{ or } k+\Delta \geq u\}$$

ここで $\Delta_{k,\alpha}$ は k 回目の遷移時の重みの総和に対し「重みの総和が α 倍以下になる」もしくは「フェイズが終了する」までの遷移回数の期待値と考えることができる．これを用いて以下の補題を証明する．

補題 4. 任意の j, α に対して以下の式が成り立つ．

$$E[\Delta_{1,\alpha^j}] \leq \frac{j}{\alpha}.$$

証明. $\sum_{i=m}^{n} W_{t,i} > \alpha||\boldsymbol{W}_t||_1$ のとき遷移時，α の確率でフェイズが終了，つまり，$\Delta_{k,\alpha}$ の定義式の第 2 条件が成立する．$\sum_{i=m}^{n} W_{t,i} \leq \alpha||\boldsymbol{W}_t||_1$ のとき，補題 3. から，定義式の第 1 条件が成立する．このことから，任意の k に対して

$$E[\Delta_{k,\alpha}] \leq \frac{1}{\alpha}$$

が成り立つ．

$$\begin{aligned} E[\Delta_{1,\alpha^j}] &= E[\Delta_{1,\alpha}] + E[\Delta_{2,\alpha}] + \cdots + E[\Delta_{j,\alpha}] \\ &> \frac{j}{\alpha}. \end{aligned}$$

□

これらの補題を用いて 1 フェイズあたりの累積移動コストを求める．以下の補題を証明する．

補題 5. $\alpha = \ln n$ とすると以下の式が成り立つ．

$$E[u] \leq 2e \ln n.$$

証明. $1 \leq k \leq u$ に対し以下の性質が成り立つことに注目する．

$$\begin{cases} ||\boldsymbol{W}_{\tau_k}||_1 > e^{-\eta} = \frac{1}{n} \\ ||\boldsymbol{W}_{\tau_1}||_1 = n \end{cases}$$

これを踏まえて以下の式について考える．

$$E[\Delta_{1,\alpha^j}] = \min\{\Delta \mid ||\boldsymbol{W}_{\tau_{1+\Delta}}||_1 \leq \alpha^j||\boldsymbol{W}_{\tau_1}||_1 \text{ or } 1+\Delta \geq u\}$$

この式に対し $\alpha = 1/e, j = 2\ln n$ を代入すると，右辺の第 1 条件式 $||\boldsymbol{W}_{\tau_{1+\Delta}}||_1 \leq \alpha^j||\boldsymbol{W}_{\tau_1}||_1$ が成立しなくなり，第 2 条件のフェイズが終了する事象しか起こりえなくなる（補題 4.）．つまり，$\alpha = 1/e, j = 2\ln n$ とすると，$\Delta_{1,\alpha}$ はフェイズあたりの遷移回数の期待値の上界と考えることができる．これより，

$$E[u] \leq E[\Delta_{1,\alpha^j}] = \frac{2\ln n}{1/e} = 2e \ln n.$$

□

補題 6. $\eta = \ln n$ とする．各フェイズにおいて重み付きマーキングアルゴリズムが被る累積損失の期待値は高々 $6e \ln n$ である．

証明. 各フェイズにおいて

$$\text{（累積移動コストの期待値）} = E[u] \leq 2e \ln n.$$

また，1 回の遷移に対し，アルゴリズムが被る処理コストは高々 L であることより，

$$\text{（累積処理コストの期待値）} \leq L \times \text{（累積移動コストの期待値）}.$$

L s.t. $ne^{-\eta L} \leq e^{-\eta}$ より，

$$L \geq \frac{\ln n}{\eta} + 1 = 2$$

$$\begin{aligned} &\text{（累積損失の期待値）} \leq 3 \times \text{（累積移動コストの期待値）} \\ &\leq 6e \ln n. \end{aligned}$$

□

補題 2. 補題 6. を用いて競合比の上界を求めると $6e \ln n$ が求まる．

4. まとめ

4.1 本研究の成果

本稿では，組み合わせ論的 MTS 問題において，重み付きマーキングアルゴリズムという手法を提案した．本手法は競合比が $6e \ln n$ つまり既存のアルゴリズムと同等の $O(\log n)$ かつ，オンライン最適化問題で用いられる Hedge アルゴリズムが効率的に動く問題クラスに対して，既存手法よりも効率的に問題を解くことができる (例．s-t パスなど)．

4.2 今後の課題

本研究の今後の課題として 2 つ考えられる．1 つは提案アルゴリズム重み付きマーキングアルゴリズムの元となったマーキングアルゴリズムが組み合わせ論的 MTS 問題で特定のクラスで効率的に動かすことができないことを明らかにし，提案アルゴリズムの有用性を高めることである．ここで我々はマーキングアルゴリズムで抱えるサンプリングの課題は特定の問題クラスで $\#P$ 困難であると予想している．

もう一つに，重み付きマーキングアルゴリズムが解ける問題クラスをより多く見つけ有用性を高めることである．，今回本稿で考えた一様距離空間だけでない移動コストを持つクラス（オンラインランキング問題等）への適用法も明らかにしていきたい．

参考文献

[1] A. Borodin, R. El-Yaniv : *Online Computation and Competitive Analysis* : Cambridge University Press,123-149(1998)

[2] A. Borodin, N. Linial, M. Saks : An optimal on-line algorithm for metrical task system. *JACM: Journal of the ACM* 39(4), 745 - 763 (1992)

[3] J. Abernethy, P. L. Bartlett, N. Buchbinder ,I. Stanton : A regularization approach to metrical task systems. *In ALT* , (2010)

[4] N. Buchbinder, S . Chen, J. Naor, O.Shamir : Unified algorithms for online learning and competitive analysis. *Journal of Machine Learning Research - Proceedings Track* , 23:5.1 - 5.18 (2012).

[5] N. Bansal, N.Buchbinder, J.Naor : Towards the randomized k-server conjecture: A primal-dual approach. *In SODA* , 40 - 55 (2010).

[6] A. Blum, S. Chawla, A. Kalai : Static optimality and dynamic search-optimality in lists and trees. *Algorithmica* , 36(3) : 249 - 260 (2003).

[7] Y. Freund, R. E. Schapire : A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* , 55 : 119 - 139 (1997).

[8] N . Cesa-Bianchi, G.Lugosi : Combinatorial Bandits *Journal of Computer and Systems Sciences*, 78 : 1404 - 1422 (2012).