数理解析研究所講究録
978巻 1997年 89-96

オンライン k-server ゲームの計算複雑性

The Computational Complexity of On-line k-server Game

山家 明男　　櫻井 幸一
Akio YANBE　Kouichi SAKURAI

九州大学大学院 システム情報科学研究科 情報工学専攻
〒812-81 福岡市東区箱崎 6-10-1
Department of Computer Science and Communication Engineering
Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-81, Japan
Phone.092-642-4050 Fax.092-632-5204
{yanbe,sakurai}@csce.kyushu-u.ac.jp

概要: 本研究では，オンラインアルゴリズムを扱うモデルとしてよく知られている k-server 問題を二人ゼロ和ゲームとして扱う．k-serer 問題を，最適戦略に関して様々な結果が知られている別のゲーム (Mean Payoff Game) に変換することにより，k-server 問題の最適戦略の存在性とその計算複雑性について考察を行なった．結果として，k-server 問題には純粋最適戦略が存在することを示した．計算複雑性に関しては，我々の最適戦略を計算する方法では指数時間かつ指数領域かかることがわかった．

キーワード：k-server 問題，Mean Payoff Game，オンラインアルゴリズム，競合比，最適戦略，計算複雑性．

Abstract: We investigate the k-server problem for which on-line algorithms have been studied from the competitive point of view. In this paper, we regard the k-server problem as a two-person zerosum game. We consider the existance and the computational complexity of optimal strategies of the k-server problem by translating it to mean payoff games that have been studied on optimal strategies. As a result, we show that there exists a pure optimal strategy for the k-server problem, and that our method requires exponential time and exponential space to compute an optimal strategy of the k-server problem.

key words: k-server problem, mean payoff game, on-line algorithms, competitive ratio, optimal strategy, computational complexity.

1 はじめに

k-server 問題は，距離空間 M(n 頂点，d: 距離関数) と M の頂点上を自由に動かせる $k(<n)$ 個の server として与えられる．一つの要求はある一つの頂点に対して出されるが，この頂点上に server がのっていなかった場合いずれかの server をこの頂点まで動かしてこなければならない．ここでの問題は，頂点への連続した要求列に応じて，server の配置を動的に変更して，server の移動距離の和 (コスト) をなるべく小さくすることを目的とする問題である．

k-server は全ての入力列を事前に知ることはできないので，オンラインで server の配置を決定する必要がある．オンラインアルゴリズムに対しては競合比 (オフラインアルゴリズムとの性能比，competitive ratio) を使った解析が一般に行なわれている．オンラインアルゴリズム A の競合比が c であるとは，全ての要求列 ρ に対して，

$$\mathrm{cost}_A(\rho) \leq c \cdot \mathrm{opt}(\rho) + b$$

が成り立つことである．ここで $\mathrm{opt}(\rho)$ はオフラインで ρ を処理した時のコスト，$\mathrm{cost}_A(\rho)$ は A で ρ を処理した時のコスト，b は初期状態に依存する定数である．

k-server 問題では競合比が k 以下になるオンラインアルゴリズムは存在しないこと (下限) が知られており [MMS90]，また現在知られている最も効率の良いアルゴリズムの競合比 (上限) は $2k+1$ である [KP95]．競合比の上限と下限が一致するようなオンラインアルゴリズムを最適競合アルゴリズムという．その他，k-server 問題を制限したものについてもいくつか結果が知られている．2-server 問題に対しては競合比が 2 のアルゴリズムがある [MMS90] が，$k \geq 3$ になると競合比が k のものは知られていない．また，M の頂点数 $n = k+1$ の場合 [MMS90] や，距離空間 M の代わりに頂点が一直線にならんでいたり [CKPV91]，木の形にならんでいる場合 [CL91] については競合比が k になる最適競合オンラインアルゴリズムが知られている．

また，k-server 問題よりも一般的なオンラインシステムとして Metrical Task System(MTS) というモデルが考えられており，k-server 問題は MTS に変換可能である [MMS90]．k-server 問題から構成された MTS の状態数は $\binom{n}{k}$ となる．MTS には競合比が $2n-1$ の適最競合アルゴリズムが知られており [BLS92]，この結果を k-server 問題に利用すると，$2\binom{n}{k}-1$ という k-server 問題の競合比の上限が得られる [MMS90]．

また，競合比はすべての例題を考えた時のオフラインコストとオンラインコストの比の最大値である．MTS の $2n-1$ という競合比もすべての MTS の例題を考えた時の最大の比である．Burley らは [BS95]，個々の例題に対しては競合比が $2n-1$ より小さくなるオンラインアルゴリズムがあり，個々の例題に対して最適競合オンラインアルゴリズムを設計する複雑さは PSPACE 困難であることを示している．

本研究では，k-server 問題を Mean Payoff Game [ZP95] という二人ゼロ和ゲームに変換する方法を提案し，k-server 問題に対してオンラインアルゴリズムを設計する計算複雑性をゲームの最適戦略を計算する複雑さとして考える．これによって，server の数が定数の場合には，MPG で既に知られている結果を使って，最適戦略が非決定性多項式時間で計算可能であることを示し，また，この場合のランダム要求に対する最適戦略は決定性の多項式時間で計算できることを示す．

2 定義

2.1 k-server 問題

k-server 問題 [MMS88, MMS90] は距離空間 $M = \{V, d\}$ で定義される．ここで，V は頂点集合，d は三角不等式を満たす対称距離関数とする．つまり，すべての頂点 x, y, z に対して，

$$d(x,x) = 0, d(x,y) = d(y,x), d(x,y) \leq d(x,z) + d(z,y)$$

である．距離空間 M において，k 個の server が頂点上に存在しており，各 server は頂点から頂点へ移動することができる．M 上に k 個の server が存在しているとき，この配置を，server が存在する k 個の頂点集合 S で表す．また，server を配置 S_i から S_j へ遷移させる時の最短距離を $D(S_i, S_j)$ で表す．初期配置は常に S_0 と表される．

要求列 ρ は，k-server によって処理される距離空間 M 上の頂点の列として表される．一つの要求を処理するには，一つの server をその要求頂点に移動させることが必要である．もし，要求列が $\rho = r_1 r_2 r_3 \cdots r_l$ ならば，k 個の server は，$r_j \in S_j$ を満たしながら $S_0 S_1 S_2 \cdots S_l$ と配置を変えていかなければならない．ステップ j において，要求 r_j を処理するコストは，k 個の server を S_{j-1} から S_j に移すためのコストになる．つまり，$D(S_{j-1}, S_j)$ である．要求列 ρ を処理するコストは全てのステップでかかるコストの和となる．本論文中では，このコストを要求の数で割ったもの，つまり 1 要求あたりの平均コストとしても考える．

2.2 Mean Payoff Game

Mean Payoff Game は有向グラフ $G = (V_1, V_2, E)$，枝の重み関数 $w : E \to R$ およびスタート頂点 $a_0 \in V_1 \cup V_2$，で与えられる．有向グラフではどの頂点も枝の次数は 1 以上とする．

この有向グラフ上でのプレイは，まずスタート頂点 a_0 に駒が置かれ，この駒を二人のプレイヤーが次のように動か

していく．V_1 頂点上に駒がある時にはプレイヤーIが，V_2 頂点上に駒がある時にはプレイヤーIIが駒を動かす．この時通った枝を $e_1, e_2, \cdots$ とすると，プレイヤーIは利得 $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} w(e_i)$ を大きくすることを目的とし，プレイヤーIIは小さくすることを目的とする．

EhrenfeuchtとMycielskiは [EM79]，このMPGでは最適値 $\mathcal{V}$ が存在し，プレイヤーIには少なくとも $\mathcal{V}$ 以上の利得を保証する戦略があり，プレイヤーIIには利得が多くて $\mathcal{V}$ 以下になることを保証する戦略があることを示した．この時の戦略を最適戦略という．さらに，この最適戦略は，各頂点で枝を一本選び常にその枝に沿って駒を動かす純粋戦略の形で存在することをも示している．

MPGの最適戦略を求める計算複雑性は非決定性多項式時間であり，Mean Payoff Gameとある値 c の組が与えられた時に，MPGの最適値が c より大きいかどうかということを問う判定問題はNP ∩ co-NPであることが知られている [ZP95]．

2.3 Disconted Payoff Game

Discounted Payoff Game(DPG) はMean Payoff Gameのdiscounted versionであり，代表的な二人ゼロ和ゲームである単純確率ゲーム [YS96] と重要な関係を持つものである．DPGではMPGと同様に二人のプレイヤーが駒を動かしていくが，利得の計算がMPGとは少し違っている．λ を $0 < \lambda < 1$ の実数とする．プレイヤーによって i 番目に選ばれた枝 e_i の重みには $(1-\lambda)\lambda^{i-1}$ がかけられ，ゲームの利得は $(1-\lambda)\sum_{i=0}^{\infty} \lambda^{i-1} w(e_i)$ となる．プレイヤーIはこの利得を大きくすることが目的で，プレイヤーIIは利得を小さくすることが目的である．λ はdiscounting factorと呼ばれる．

DPGは有向グラフ $G = (V_1, V_2, E)$，枝の重み関数 $w : E \to R$，スタート頂点 $i \in V_1 \cup V_2$，および実数 λ で与えられる．有向グラフではどの頂点も枝の次数は1以上である．$V = V_1 \cup V_2 = \{1, 2, \cdots, n\}$ とする．$x_i(= x_i(\lambda))$ は頂点 i からスタートする時のdiscounted gameの利得とする．

このDPGの最適値や最適戦略の存在は次の定理によって示されている．

定理 2.1 ([ZP95]) discounted payoff gameの最適値 $\overline{v} = \{v_1, v_2, \cdots, v_n\}$ はつぎの方程式の唯一の解である．

$$v_i = \begin{cases} \max_{(i,j)\in E}\{(1-\lambda)w_{ij} + \lambda v_j\} & \text{if } i \in V_1, \\ \min_{(i,j)\in E}\{(1-\lambda)w_{ij} + \lambda v_j\} & \text{if } i \in V_2. \end{cases}$$

discounting factorが λ であるDPGの最適値を $\mathcal{V}(\lambda)$ とする．λ を1に近付けると $\mathcal{V}(\lambda)$ はMean Payoff Gameの最適値Vに近付くので，Mean Payoff GameはDiscounted Payoff Gameに帰着できることになる [ZP95]．

このDPGついては，我々は次の定理を得ている．

定理 2.2 ([YS95]) プレイヤーI(またはII) をランダムプレイヤーとしたDPGの最適値および純粋最適戦略は多項式時間で計算できる．

2.4 Metrical Task System

Metrical Task System [BLS92] とは，n 状態を持ち l 種類の要求を入力とするオンラインシステムである．具体的に書くとMetrical Task System=(S, T, A, B, s_0) であり，$S = \{1, 2, \cdots, n\}$ はシステムの状態集合，$T = \{1, 2, \cdots, l\}$ は入力タスク集合，A は状態遷移コスト行列で，A の要素 a_{ij} は状態を i から j に遷移させるのに必要なコストである．A は対称行列であり，三角不等式 $(a_{ij} + a_{jl} \leq a_{il})$ を満たすものとする．B はタスク処理コスト行列で，B の要素 b_{ij} は状態 i でタスク j を処理するのに必要なコストである．また，$s_0 \in S$ はシステムの初期状態である．

このシステムは1単位時間に次の三つの動作を行う．

step 1. システムが状態 i にあり，l 種類のタスクのうち一つ t を受けとる．

step 2. システムは状態 i から状態 j へ遷移する．

step 3. 状態 j でタスク t を処理する．

このうち，step 2において状態 i から状態 j に遷移するには a_{ij} のコストがかかり，step 3において状態 j でタスク t を処理するには b_{jt} のコストがかかる．つまり，1単位時間で $a_{ij} + b_{jt}$ のコストがかかることになる．

入力タスク列 I は $t_1 t_2 t_3 \ldots$ と表し，t_i が時間 i での入力とする．これに対して，スケジュール σ は $s_0 s_1 s_2 s_3 \ldots$ と表し，状態 s_i でタスク t_i が処理されることになる．なお，入力タスクを入れる側 (Taskmaster) は入力するタスクがどの状態で処理されるのかを事前に知ることはできない．これはつまり，i 番目の入力タスク t_i はちょうど時間 i で入力され，t_i は $[i, i+1)$ の期間で処理が行われるという仮定によるものである．

システム側 (Scheduler) はこの単位時間あたりの平均コストをなるべく低くおさえられるように，状態の遷移の仕方を考えなければならない．これはつまり，入力を読んでその処理を行うために次に移る状態を決定するスケジュール σ を構成することであり，事前に入力タスク列を知ることができないことからオンラインアルゴリズムがこれを行うことになる．

MTSは k-server 問題をより一般化したモデルであり，k-server 問題はMTSに変換することができる [BLS92]．MTSをMean Payoff Gameに変換する方法がZwickら [ZP95] によって示されているが，これで求められる最適戦

略は，入力側がシステムの状態遷移を一つ先読みできるという仮定を設けたうえでの最適戦略であり，MTSの本来の最適戦略を考えることができるようなMPGへの変換法があるかどうかは未解決である．

3 最適戦略と最適競合アルゴリズム

我々はk-server問題をserver側と要求側の二人のプレイヤによるゼロ和ゲームとして扱った．ミニマックス定理 [Suzuki] により二人ゼロ和ゲームでは必ず最適戦略と最適値V_{opt}が存在する．server側の最適戦略は要求側がどんな戦略をとってもコストの大きさをV_{opt}より大きくはできないような戦略であり，逆に要求側の最適戦略はserver側がどんな戦略をとってもコストの大きさをV_{opt}より小さくはできないような戦略である．両方最適戦略をとった時のコストはV_{opt}である．

要求側が最適戦略を使って入力してくる要求列ρ_{opt}を，競合オンラインアルゴリズムを使って処理した場合のコストを$\mathrm{cost}_{on}(\rho_{opt})$，serverの最適戦略を使って処理した場合のコストを$\mathrm{cost}_{opt}(\rho_{opt})$とすると，一般的には$\mathrm{cost}_{on}(\rho_{opt}) \geq \mathrm{cost}_{opt}(\rho_{opt})$である．等号が成り立つのは，要求側の戦略に対して，server側の最適戦略によるserverの動きと競合アルゴリズムによるserverの動きが一致した場合だけである．つまり，最悪の場合を考えると最適戦略の方が優れていることになる．ただし，その他の要求列に対してオンラインアルゴリズムと最適戦略とではどちらの方がコストが小さく押えられるかについては，解析することは難しいと思われる．

[MR95]において，k-server問題の限定版であるページング問題(頂点間の距離が全て1になっている)に競合比による解析を導入する動機が述べられている．ページング問題をゲームと考えると，server側の最適戦略はserverがのっている頂点に要求が来た時にはserverを動かさず，空頂点に要求が来た時にのみ任意のserverを要求頂点に動かすようなやりかたである．空頂点に要求が来た時にどのserverをその要求頂点に持ってくるかを決定するアルゴリズムが幾つか考えられている(これはどれも最適戦略を満たしている)が，そのアルゴリズムのうちどれを使っても，コストの和がnとなるような長さnの要求列が存在している．つまり，常にserverの存在しない頂点を要求頂点とするような要求列である．このため，ページング問題に対するアルゴリズムを最悪コストで評価してもすべて同じ評価となってしまう．ところが，その幾つかのページングアルゴリズムに対して競合比による評価を行なうと，競合比が有限であるもの(競合アルゴリズム)とそうでないものが現れてくる．よって，ページング問題では最適戦略であり，かつ，競合比が有限の競合アルゴリズムを使うのが適当だと思われる．

しかし，k-server問題では競合比による解析しか行なわれておらず，ゲームとしての最適戦略を解析することは重要な研究であり，競合アルゴリズムと最適戦略を比較するための有効な方法を発見することも，大きな目的と考えられる．

4 k-server問題の最適戦略

我々はこの節で，k-server問題を二人ゼロ和ゲームと考える．二人のプレイヤはserver側と要求側であり，server側はコストを小さくしようとし要求側はコストを大きくしようとするものとする．まず，k-server問題を別のゲームに変換して，その最適戦略の存在について調べることを試みた．

二人ゼロ和ゲームでは，確率を含む混合戦略まで考えれば必ず最適戦略が存在することが知られているが，確率を含まない純粋戦略の形で最適戦略が存在するかどうかは各モデルによって異なり，自明ではない．しかしここでは，k-server問題には確率を使わない純粋戦略の形で最適戦略が存在することを示す．この際，k-server問題を変換する先として，最適戦略に関して以下の定理が知られているMean Payoff gameを利用する．

補題 4.1 ([EM79]) Mean Payoff Gameには純粋最適戦略が存在する．

定理 4.2 k-server問題には純粋最適戦略が存在する．

証明: 与えられたk-server問題の距離空間$M=(V,d)$,$V=\{1,2,\cdots,n\}$をMPG $(G=(\widetilde{V_1},\widetilde{V_2},\widetilde{E}),w)$に変換する方法は次の通りである．なお，MPGのグラフGは二部グラフとなる．要求を入れる側をプレイヤI，serverを動かす側をプレイヤIIとしてMPGを構成する．

$m=\left(\begin{array}{c} n \\ k \end{array}\right)$とする．MPGの頂点集合を$\widetilde{V_1}=\{\tilde{1},\tilde{2},\cdots,\tilde{m}\}$, $\widetilde{V_2}=\{(\tilde{x},y)|\tilde{x}\in\widetilde{V_1},y\in V,\tilde{1}\leq\tilde{x}\leq\tilde{m},1\leq y\leq n\}$とする．ここで，$\widetilde{V_1}$の各頂点は，$M$における$k$個のserverの配置をそれぞれ表しているものとする．$\widetilde{V_1}$の頂点$\tilde{x}$が表すk個のserverの配置を$S_{\tilde{x}}$とする．次に枝を考える．$\widetilde{V_1}$から$\widetilde{V_2}$へ出ている枝は，すべての$\widetilde{V_1}$頂点$\tilde{1}\leq\tilde{x}\leq\tilde{m}$と$V$頂点$1\leq y\leq n$に対して$\tilde{x}\rightarrow(\tilde{x},y)$とする．この枝のコストは全て0とする．この枝は$k$-serverの配置が$S_{\tilde{x}}$の時に要求$y$を受けとったことを示している．逆に，$\widetilde{V_2}$から$\widetilde{V_1}$へ出ている枝は，各$\widetilde{V_2}$頂点$(\tilde{x},y)$から配置$S_{\tilde{z}}$において頂点$y$上にserverがのっているような全ての頂点$\tilde{z}\in\widetilde{V_1}$に$(\tilde{x},y)\rightarrow\tilde{z}$となるようにつくる．この枝には$D(S_{\tilde{x}},S_{\tilde{z}})$のコストをつける．以上の変換を厳密に書き下すと次のようになる．

Input: $M=(V,d),V=\{1,2,\cdots,n\}$

Output: $G=(\widetilde{V_1},\widetilde{V_2},\widetilde{E})$, w: 重み関数($w(x,y)$は枝(x,y)の重み)

1: $m = \binom{n}{k}$

2: $\widetilde{V_1} = \{\tilde{1}, \tilde{2}, \cdots, \tilde{m}\}$,
$\widetilde{V_2} = \{(\tilde{x}, y) | \tilde{1} \leq \tilde{x} \leq \tilde{m}, 1 \leq y \leq n\}$,
$\widetilde{E} = \emptyset$

3: for $\tilde{1} \leq \tilde{x} \leq \tilde{m}$ do
  for $1 \leq y \leq n$ do
    $\widetilde{E} = \widetilde{E} \cup \{(\tilde{x}, (\tilde{x}, y))\}$
    $w(\tilde{x}, (\tilde{x}, y)) = 0$

4: for $1 \leq y \leq n$ do
  for $\tilde{1} \leq \tilde{z} \leq \tilde{m}$ do
    if $y \in S_{\tilde{z}}$ then
      $\widetilde{E} = \widetilde{E} \cup \{((\tilde{x}, y), \tilde{z})\}$
      $w((\tilde{x}, y), \tilde{z}) = D(S(\tilde{x}), S(\tilde{z}))$

この変換で構成されるMPGにおいて，プレイヤIIの最適戦略がserver側の最適戦略，プレイヤIの最適戦略が要求側の最適戦略と対応していることは明らかである．補題4.1より，MPGの最適戦略は純粋戦略の形で存在している．つまり，MPGでの最適戦略は各頂点から出ている枝を一本選んだ枝集合になっている．よって，プレイヤIIの戦略は，$\widetilde{V_2}$ の全ての頂点 $(\tilde{x}, y)$ に対して $\widetilde{V_2}$ から出ている枝 $(\tilde{x}, y) \to \tilde{z}$ の集合として表される．これを k-server 問題に置き換えて考えてみると，配置 $S_{\tilde{x}}$ で要求 y を処理するために $S_{\tilde{z}}$ に配置を変えることを全ての配置と要求の組に対して示した戦略となる．また，プレイヤIの戦略は，$\widetilde{V_1}$ の全ての頂点 $\tilde{x}$ に対して $\widetilde{V_1}$ から出ている枝 $\tilde{x} \to (\tilde{x}, y)$ の集合として表される．これは k-server 問題においては，配置 $S_{\tilde{x}}$ のときは常に要求 y を入力するという k-server の全ての配置に対して入力する要求を示した戦略となる． □

4.1 k-server 問題の例

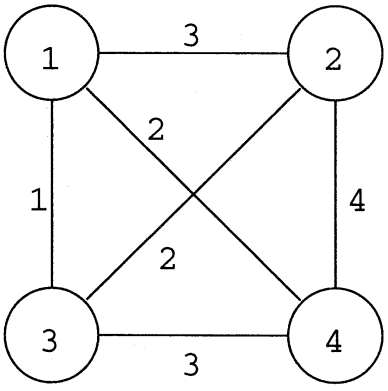

図 1: 2-server 問題 (頂点数 $n = 4$)

ここでは，k-server 問題の例を使って，MPGへの変換と最適戦略を具体的に示す．図1の例におけるserverの配置は次の6種類である (各配置 $S_{\tilde{i}}$ はserverの存在する頂点の集合である)．

$$S_{\tilde{1}} = \{1,2\}, S_{\tilde{2}} = \{1,3\}, S_{\tilde{3}} = \{1,4\},$$
$$S_{\tilde{4}} = \{2,3\}, S_{\tilde{5}} = \{2,4\}, S_{\tilde{6}} = \{3,4\}.$$

よって，この k-server 問題をMPGに変換すると，頂点集合 V_1 は $\{\tilde{1}, \tilde{2}, \tilde{3}, \tilde{4}, \tilde{5}, \tilde{6}\}$，$V_2$ は $\{(\tilde{1}, 1), (\tilde{1}, 2), (\tilde{1}, 3), (\tilde{1}, 4), (\tilde{2}, 1), \cdots, (\tilde{6}, 3), (\tilde{6}, 4)\}$ となる．

V_1 から V_2 への枝は，各 V_1 頂点 $\tilde{i}$ $(\tilde{1} \leq \tilde{i} \leq \tilde{6})$ から $(\tilde{i}, x)$，$1 \leq x \leq 4$ へ図2のように出す．この枝の重みはすべて0である．

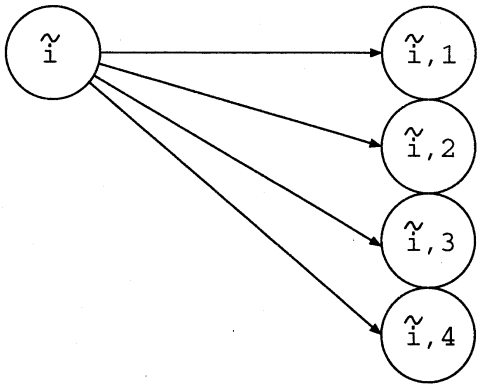

図 2: V_1 から V_2 への枝

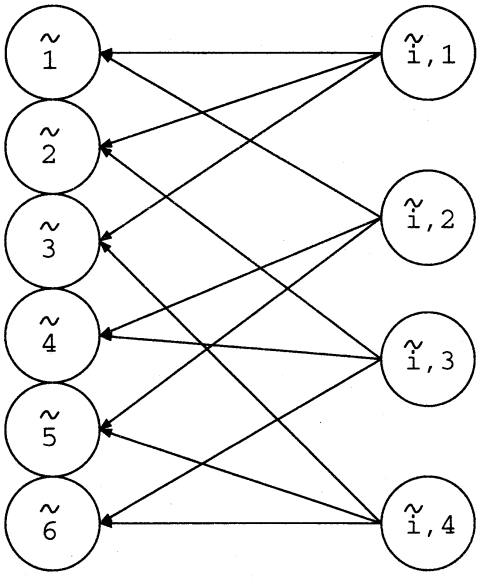

図 3: V_2 から V_1 への枝

V_2 から V_1 への枝は，各 V_2 頂点 $(\tilde{i}, x)$ から $S_{\tilde{j}}$ に x を含むような頂点 $\tilde{j}$ に出す．この例では，図3のように，すべての $\tilde{i}$ について，$(\tilde{i}, 1)$ からは $\tilde{1}, \tilde{2}, \tilde{3}$ へ，$(\tilde{i}, 2)$ からは

$\tilde{1},\tilde{4},\tilde{5}$ へ，$(\tilde{i},3)$ からは $\tilde{2},\tilde{4},\tilde{6}$ へ，$(\tilde{i},4)$ からは $\tilde{3},\tilde{5},\tilde{6}$ へ出す．このとき，各枝 $(\tilde{i},x)\to\tilde{j}$ には $D(S_{\tilde{i}},S_{\tilde{j}})$ の重みが付けられる．

こうして構成されたMPGにおけるプレイヤIIの最適戦略は次のような枝の集合として記述される．

$$(\tilde{1},1)\to\tilde{1}\quad(\tilde{1},2)\to\tilde{1}\quad(\tilde{1},3)\to\tilde{2}\quad(\tilde{1},4)\to\tilde{5}$$

$$(\tilde{2},1)\to\tilde{2}\quad(\tilde{2},2)\to\tilde{1}\quad(\tilde{2},3)\to\tilde{2}\quad(\tilde{2},4)\to\tilde{6}$$

$$(\tilde{3},1)\to\tilde{3}\quad(\tilde{3},2)\to\tilde{5}\quad(\tilde{3},3)\to\tilde{6}\quad(\tilde{3},4)\to\tilde{3}$$

$$(\tilde{4},1)\to\tilde{1}\quad(\tilde{4},2)\to\tilde{4}\quad(\tilde{4},3)\to\tilde{4}\quad(\tilde{4},4)\to\tilde{5}$$

$$(\tilde{5},1)\to\tilde{1}\quad(\tilde{5},2)\to\tilde{5}\quad(\tilde{5},3)\to\tilde{6}\quad(\tilde{5},4)\to\tilde{5}$$

$$(\tilde{6},1)\to\tilde{2}\quad(\tilde{6},2)\to\tilde{5}\quad(\tilde{6},3)\to\tilde{6}\quad(\tilde{6},4)\to\tilde{6}$$

これは，もとの k-server 問題においてはserver側の最適戦略を表しており，$(\tilde{i},x)\to\tilde{j}$ は，配置が $S_{\tilde{i}}$ のときに要求 x を受けとったら $S_{\tilde{j}}$ に配置を変えることを意味している．

また，プレイヤIの最適戦略は次にようになる．

$$\tilde{1}\to 3\quad\tilde{2}\to 4\quad\tilde{3}\to 2$$
$$\tilde{4}\to 4\quad\tilde{5}\to 1\quad\tilde{6}\to 2$$

これは，k-server 問題において要求側の最適戦略を表しており，$\tilde{i}\to x$ はserverの配置が $S_{\tilde{i}}$ のときに x を要求頂点とすることを示している．

この最適戦略を両者が使った場合，serverの配置は $S_{\tilde{1}}\to S_{\tilde{2}}\to S_{\tilde{6}}\to S_{\tilde{5}}\to S_{\tilde{1}}\to\cdots$ と繰り返すことになり，配置の遷移はそれぞれコストが2かかっている．よって，1要求あたりの平均コストは2となる．

4.2 k-server 問題の最適戦略の計算複雑性

今回提案した k-server 問題をMPGに変換する方法では，一般の k-server 問題の最適戦略を計算する複雑性に関してMPGと同じ結果を導くことはできない．なぜなら，定理2.1, 定理2.2での多項式時間はMPGの頂点数に関しての多項式であり，我々の方法で構成されたMPGの頂点数は $(n+1)\binom{n}{k}$ とかなり増えてしまうためである．

また，この最適戦略は枝の集合として表されるが，その枝の本数はserver側は $n\binom{n}{k}$ 本，要求側は $\binom{n}{k}$ 本となり，このままでは最適戦略を記述する記憶容量までも指数サイズ必要となる．データ構造をうまくとって最適戦略の記述領域を小さくすることができないか考えてみることは興味深い問題であると思われる．

ただ，server数が定数 c の場合および $n-c$ の場合にはMPGの頂点数は $O(n^c)$ となり n の多項式となるので，MPGについての結果[ZP95]と結びつけると系4.3が，ランダムプレイヤを含めた結果[YS95]と結びつけると系4.4がいえる．

系 4.3 server数が定数 c または $n-c$ の場合には，k-server 問題の最適戦略は非決定性多項式時間で計算可能である．

系 4.4 server数が定数 c または $n-c$ の場合には，ランダム要求に対する k-server 問題の最適戦略は多項式時間で計算可能である．

ここで，k-server 判定問題を次のように定義する．

入力： 距離空間 M，初期配置 S_0，定数 $R\geq 0$.

問い： server側と要求側がともに最適戦略を使った時，1要求あたりの平均コストは R より大きいか．

server数が定数 c または $n-c$ の場合には，系4.3より非決定性多項式時間で最適戦略と平均コストが計算できるので，次の系が成り立つ．

系 4.5 server数が定数 c または $n-c$ の場合には，k-server 判定問題はNP∩co-NPである．

k-server 問題の実際の例として，プリンタやビットマップディスプレイにおけるフォントのキャッシングが考えられる．プリンタまたはディスプレイのフォント用メモリ(キャッシュ)には，ある一定数の文字しか置くことができず，また，フォント用メモリに入っている文字しか出力できないものとする．フォント用メモリにない文字を出力しようとすると，いったんその文字をフォント用メモリに読み込んでから出力することになる．キャッシュへ一文字読み込む代わりに，フォント用メモリにある文字を一文字除かなければならない．このときに，読み込み文字と除去文字との反転ビット数をコストと考え，このコストを少なくすることが目的である．k-server 上では，距離空間の各頂点は各文字フォントを表し，serverの数 k はキャッシュの大きさ(読み込める文字数)を表していることになる．我々の結果をこのキャッシュ問題に適用することを考えてみると，キャッシュがハードで作ってあってその大きさが固定である時，これは定数とみなすことができ，系4.3，系4.4がそのまま適用できることになる．

5 頂点数 $n=k+1$ のときの最適戦略

k-server 問題の最適戦略をMPGに変換して考えることは，k-server 問題の個々の問題例に対して最適なserverの配置の遷移を与えるものである．しかし，次のような限定版の k-server 問題に対しては，多項式時間で計算可能な共通の最適戦略を考えられることがわかった．

ここでは，距離空間 M の頂点数 n が $k+1$ であるときに限定して考える．これは server がのっていない頂点が常に一箇所だけであるような場合である．この場合に，server の配置の遷移を決定するような多項式時間アルゴリズム P を考える．

- アルゴリズム P
  $\min_{i\neq j} d(i,j)$ を満たす頂点 i,j を求める．i,j 以外の頂点にまず $k-1$ 個の server を割り当て，これらの server は動かさない．残りのひとつの server が，i と j のうち要求として選ばれた方に移動して処理を行なう．

定理 5.1 距離空間 M の頂点の数 $n=k+1$ であるとき，アルゴリズム P は server 側の最適戦略である．

証明: まず，要求側が常に server の存在しない頂点に要求を出す場合，どんなアルゴリズムでも 1 要求あたり少なくとも $\min_{i\neq j} d(i,j)$ のコストがかかる．また，server 側がアルゴリズム P を使う場合，どんな要求列に対しても 1 要求あたり多くて $\min_{i\neq j} d(i,j)$ のコストしかかからない．よって，定理が成り立ち，このときの 1 要求あたりのコストの最適値は $\min_{i\neq j} d(i,j)$ となる． □

系 5.2 距離空間 M の頂点の数 $n=k+1$ であるとき，常に server の存在しない頂点に要求を出すのが要求側の最適戦略である．

系 5.3 距離空間 M の頂点の数 $n=k+1$ であるとき，最適戦略は多項式時間で計算可能である．

このアルゴリズム P は，競合比が有限ではないことは簡単に分かる (アルゴリズム P が常に server を動かさなければならないような要求列を考えればよい)．よって，$n=k+1$ の場合の k-server 問題には，最適戦略であり，かつ競合アルゴリズムであるようなものは存在しないと思われる．ゆえに，一般の k-server 問題に対しても最適戦略であり，かつ競合アルゴリズムであるようなオンラインアルゴリズムは存在しないと考えられる．

6 おわりに

$n=k+1$ の場合の k-server 問題に対する競合比 k の最適競合アルゴリズム [MMS90] は，バランスアルゴリズムと呼ばれるもので，以下のような動作をする．

- バランスアルゴリズム (BAL) [MMS90]
  BAL は server それぞれについての合計移動コストを扱う．ある時点で頂点 i にある server の合計移動コストを Z_i と表す．ここで，頂点 j へのリクエストを考える．もし頂点 j に server があれば BAL はなにもしない．もし頂点 j に server がなければ，BAL は $Z_i+d(i,j)$ を最小にする頂点 i 上の server を j に移す．言いかえれば，BAL は各 server の合計移動コストを同じにするように server を動かすものである．

我々は，$n=k+1$ の場合の最適戦略は多項式時間で計算できることを示したが，この最適戦略はバランスアルゴリズムとは明らかに異なるものである．よって，個々の例題に対してはどちらが優れているのか調べてみたいと思うのは自然なことである．このとき，要求列 ρ に対する最適戦略のコストを $\mathrm{cost}_{opt}(\rho)$，最適競合アルゴリズムでのコストを $\mathrm{cost}_{on}(\rho)$ として，

$$\frac{\underset{\rho}{\mathrm{AVE}}\ \mathrm{cost}_{opt}(\rho)}{\underset{\rho}{\mathrm{AVE}}\ \mathrm{cost}_{on}(\rho)}$$

のようなランダム要求に対するコストの期待値の比を個々の例題に対して求めることは，はかり方の一つとして有効であると思われる．

参考文献

[BLS92] Borodin, A., Linial, N., and Saks, M.E., "An optimal on-line algorithm for metrical task system," *J. ACM* **39** (1992) 745–763.

[BS95] Burley, W.R. and Irani, S., "On algorithm design for metrical task systems," *Proc. of the 6th annual ACM-SIAM symposium on Discrete Algorithms* (1995) 420–429.

[CKPV91] Chrobak, M., Karloff, H.J., Payne, T., and Vishwanathan, S., "New results on server problems," *SIAM J. Disc. Math.* **4** (1991) 172–181.

[CL91] Chrobak, M. and Larmore, L.L., "An optimal on-line algorithm for k-servers on trees," *SIAM J. Comput.***20** (1991) 144–148.

[Con92] Condon, A., "The Complexity of Stochastic Games," *Information and Computation* **96** (1992) 203–224.

[EM79] Ehrenfeucht, A. and Mycielski, J., "Positional strategies for mean payoff games," *International Journal of Game Theory* **8** (1979) 109–113.

[FRR90] Fiat, A., Rabani, Y., and Ravid, Y., "Competitive k-server algorithms," *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science* (1990) 454–463.

[Gro91] Grove, E., "The harmonic online k-server algorithm is competitive," *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing* (1991) 260–266.

[KP95] Koutsoupias, E. and Papadimitriou, C.H., "On the k-server conjecture," *J. ACM* **42** (1995) 971–983; also in: *Proc. 25th Ann. ACM Symp. on Theory of Computing* (1994) 507–511.

[MMS88] Manasse, M.S., McGeoch, L.A., and Sleator, D.D., "Competitive algorithms for on-line problems," *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (1988) 322-333.

[MMS90] Manasse, M.S., McGeoch, L.A., and Sleator, D.D., "Competitive algorithms for server problems," *J. Algorithms* **11** (1990) 208–230.

[MR95] Motwani, R. and Raghavan, P., *Randomized Algorithms*, Cambridge University Press (1995)

[Suzuki] 鈴木光男, 「ゲーム理論入門」, 共立出版.

[YS95] 山家 明男, 櫻井 幸一, "二人確率ゲームの最適戦略に関する計算複雑性," 京大数理解析研究所研究集会 [数理モデルにおける最適化理論] (平成 7 年 11 月).

[YS96] Yanbe, A. and Sakurai, K., "On the Complexity of Computational Problems Associated with Simple Stochastic Games," *Second Annual International Computing and Combinatorics Conference (COCOON'96)* **LNCS1090** 310–322.

[ZP95] Zwick, U. and Paterson, M., "The complexity of mean payoff games on graphs," *ECCC Reports Series 1995* **TR95-040**; also in: *COCOON'95* **LNCS959** 1–10.