数理解析研究所講究録 第2108巻 2019年 95-114

最小費用全域木ゲームのShapley値に対する近似アルゴリズム

高瀬光一[†]　安藤和敏[‡,*]

[†] 静岡大学大学院総合科学技術研究科
[‡] 静岡大学工学部

Koichi Takase[†] and Kazutoshi Ando[‡]

[†]Graduate School of Integrated Science and Technology, Shizuoka University

[‡]Faculty of Engineering, Shizuoka University

概 要

最小費用全域木ゲームはそれを定義するネットワークの費用関数が木距離である場合には木距離最小費用全域木ゲームと呼ばれる. 一般の最小費用全域木ゲームのShapley値の計算は#P-困難であるが, 木距離最小費用全域木ゲームのShapley値は多項式時間で計算できることが知られている. 本研究では, 木距離最小費用全域木ゲームに対する多項式時間アルゴリズムのアイデアに基づいて, 一般の最小費用全域木ゲームのShapley値に対する多項式時間近似アルゴリズムを導入した. このアルゴリズムの近似精度を評価するためにランダムに生成した費用関数を入力として数値実験を行った結果, 与えられた費用関数が2次元ユークリッド距離の場合には最大でも14%程度の相対誤差を持つということが観察された.

1　はじめに

協力ゲーム理論における中心的課題は, プレイヤー全体として負担する費用をプレイヤー間でどのように配分するかということである. 各プレイヤーへの費用配分を指定する方法はゲームの解と呼ばれ, Shapley値 [8] はもっとも重要な解の一つである. 本論文で扱う最小費用全域木ゲームは Bird [2] によって導入された協力ゲームの一つのクラスであり, 通信網構築などの状況において, ユーザ間での適切な費用配分法を分析するためのゲーム理論的モデルとしても考えることができる. $N = \{1, \ldots, n\}$ をプレーヤー（あるいは，クライアント）の集合とし, $N' = N \cup \{r\}$ とする. r はソースと呼ばれる. さらに, 各 $u, v \in N'$ に対して非負の実数 $c(u, v)$ を割り当てる関数 c が与えられている. これを費用関数と呼ぶ. 最小費用全域木ゲームは, プレーヤー集合 N と以下で定義される特性関数 $\tilde{c}\colon 2^N \to \mathbb{R}$ の組 $(N, \tilde{c})$ によって定義される. ここで, 各 $S \subseteq N$ に対して, $\tilde{c}(S)$ は完全グラフ $\mathrm{K}_{S\cup\{r\}}$ の最小費用全域木の費用である.

最小費用全域木ゲームを上述したような現実的な問題へと応用する際にはゲームの解の効率的な計算が重要であるが, 最小費用全域木ゲームのShapley値の計算は#P-困難であることが示されている [1]. したがって, 効率的に計算が可能でかつ精度の高い近似値の計算が次善の策となる.

*Corresponding author. Email: ando.kazutoshi@shizuoka.ac.jp

最小費用全域木ゲームの Shapley 値の近似アルゴリズムとして一般的な協力ゲームの Shapley 値に対する近似アルゴリズムであるサンプリング・アルゴリズム [3, 9] が知られている. しかし, 要求される近似精度を達成するためのサンプル数が非常に大きいため, その計算のためには膨大な時間を要するという欠点がある.

最小費用全域木ゲームは, それを定義するネットワークの費用関数が木距離である場合には木距離最小費用全域木ゲームと呼ばれる. 木距離最小費用全域木ゲームの Shapley 値は, 多項式時間で計算できることが知られている [1]. このアルゴリズムの概略は以下の通りである. 任意の費用関数 c に対して, ある $\{0,1\}$-費用関数 c_i $(i=0,\ldots,l)$ が存在して, c は c_i の非負結合によって表される.

$$c=\sum_{i=0}^{l}\delta_i c_i. \tag{1}$$

各 $i=0,\ldots,l$ に対して, $c_i(u,v)=0$ のときに (u,v) を枝として持つようなグラフを $G(c_i)$ とする. もし c が木距離であれば, すべての $i=0,\ldots,l$ に対して $G(c_i)$ はコーダルグラフとなる. グラフ $G(c_i)$ がコーダルグラフのときには, c_i に関連する最小費用全域木ゲームの Shapley 値 $\mathrm{Sh}(\tilde{c_i})$ は多項式時間で計算できる [1] ため, Shapley 値の線形性により c に関連する最小費用全域木ゲームの Shapley 値 $\mathrm{Sh}(\tilde{c})$ を得る.

$$\mathrm{Sh}(\tilde{c})=\sum_{i=0}^{l}\delta_i\mathrm{Sh}(\widetilde{c_i}). \tag{2}$$

木距離最小費用全域木ゲームに対する多項式時間アルゴリズムのアイデアに基づいて, 本論文では一般の最小費用全域木ゲームの Shapley 値に対する近似アルゴリズムを導入する. 以下にこのアルゴリズムの概略を述べる. c を任意の費用関数とし, c は式 (1) のように c_i の非負結合によって表されるとする. 各 $i=0,1,\ldots,l$ に対して $G(c_i)$ のコーダルグラフによる近似 G_i' を求め, $G(c_i')=G_i'$ であるような $\{0,1\}$-費用関数を c_i' とする. そして,

$$\mathrm{Sh}(\tilde{c'})=\sum_{i=0}^{l}\delta_i\mathrm{Sh}(\widetilde{c_i'}) \tag{3}$$

を Shapley 値 $\mathrm{Sh}(\tilde{c})$ の近似とする. G_i' はコーダルであるから $\mathrm{Sh}(\widetilde{c_i'})$ は多項式時間で求められるため, $\mathrm{Sh}(\tilde{c'})$ も多項式時間で求めることができる. 本論文ではさらに, 導入した近似アルゴリズムの近似精度を評価するために, ランダムに生成した問題例を入力として近似アルゴリズムの出力の相対誤差を求める数値実験を行った. その結果, 与えられた費用関数が 2 次元ユークリッド距離の場合には最大でも 14%以内の相対誤差を持つということが観察された.

本論文の残りは以下のように構成される. 第 2 節では最小費用全域木ゲームとその Shapley 値についての基本的な事柄について述べる. 第 3 節では, 最初に木距離最小費用全域木ゲームの Shapley 値に対する多項式時間アルゴリズムについての既存の結果 [1] について述べた後, 一般の最小費用全域木ゲームの Shapley 値に対する近似アルゴリズムを導入する. 第 4 節では, 近似アルゴリズムの近似精度を評価する数値実験の結果を示す. 最後に第 5 節では結論を述べる.

2 最小費用全域木ゲームと Shapley 値

この節では, 最小費用全域木ゲームとその Shapley 値についての基本的な事柄について述べる.

2.1 協力ゲームと Shapley 値

協力ゲームとは集合 $N = \{1, \ldots, n\}$ と関数 $C : 2^N \to \mathbb{R}$ $(C(\emptyset) = 0)$ の対 (N, C) である．N を**プレイヤーの集合**, C を**特性関数**と呼ぶ．

$\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ を N の置換とする．協力ゲーム (N, C) の π に関する**限界費用ベクトル** $x(\pi) = (x(\pi)_1, \ldots, x(\pi)_n)$ は

$$x(\pi)_{\pi(v)} = C(\{\pi(1), \ldots, \pi(v)\}) - C(\{\pi(1), \ldots, \pi(v-1)\}) \quad (v = 1, \ldots, n)$$

によって定義される．$\Pi(N)$ を N のすべての置換から成る集合とする．協力ゲーム (N, C) の **Shapley 値** $\mathrm{Sh}(C) : N \to \mathbb{R}$ は,

$$\mathrm{Sh}(C)_v = \frac{1}{n!} \sum_{\pi \in \Pi(N)} x(\pi)_v \quad (v = 1, \ldots, n) \tag{4}$$

によって定義される．

$$\sum_{v=1}^{n} \mathrm{Sh}(C)_v = C(N)$$

が成り立つことは容易にわかる．また, Shapley 値の重要な性質として線形性がある．任意の実数 $\alpha, \beta \in \mathbb{R}$ と協力ゲーム $(N, C), (N, C')$ に対して, 次式が成り立つ．

$$\mathrm{Sh}(\alpha C + \beta C') = \alpha \mathrm{Sh}(C) + \beta \mathrm{Sh}(C').$$

ここで, 協力ゲーム $(N, \alpha C + \beta C')$ は $(\alpha C + \beta C')(S) = \alpha C(S) + \beta C'(S)$ $(S \subseteq N)$ で定義される．

2.2 最小費用全域木ゲームの Shapley 値

本論文において考えるすべてのグラフ $G = (V, E)$ は, 単純な (自己閉路と並列枝の無い) 無向グラフとする．グラフ $G = (V, E)$ の枝 $e \in E$ は相異なる点の非順序対であるが, $e = \{u, v\}$ の代わりに $e = (u, v)$ と表記する．グラフ $G = (V, E)$ は, $E = \{(u, v) \mid u, v \in V, u \neq v\}$ であるとき**完全グラフ**と呼ばれ, K_V によって表される．

$N' = N \cup \{r\}$ とする．r は**ソース**と呼ばれる．また, $c : N' \times N' \to \mathbb{R}_+$ をすべての $u, v \in N'$ に対して $c(u, u) = 0, c(u, v) = c(v, u)$ を満たす関数とする．このような関数を N' 上の**費用関数**と呼ぶ．このとき, 完全グラフ $\mathrm{K}_{N'}$ と c の対 $(\mathrm{K}_{N'}, c)$ を**ネットワーク**と呼ぶ．$(\mathrm{K}_{N'}, c)$ に対して, 点集合が N' であるような木 $T = (N', \Gamma)$ は**全域木**と呼ばれる．$T = (N', \Gamma)$ が全域木であるとき, Γ もまた**全域木**と呼ばれる．この Γ に対し, Γ の費用 $c(\Gamma)$ を

$$c(\Gamma) = \sum_{(u,v) \in \Gamma} c(u, v)$$

と定義する．ネットワークにおいて, 費用が最小になる全域木を**最小費用全域木**と呼び, 最小費用全域木を見つける問題を**最小費用全域木問題**と呼ぶ．最小費用全域木問題に対するアルゴリズムとして, Prim のアルゴリズム [6] などが存在する．図 1 は $N' = \{r, 1, 2, 3\}$ であるようなネットワーク $(\mathrm{K}_{N'}, c)$ の一例であり, $(\mathrm{K}_{N'}, c)$ の最小費用全域木は $\Gamma = \{(2, 3), (r, 3), (1, 2)\}$ である．

任意の $S \subseteq N$ に対して, $S' = S \cup \{r\}$ とおく．ネットワーク $(\mathrm{K}_{N'}, c)$ に関連する**最小費用全域木ゲーム**とは, N と次式で定義される特性関数 $\tilde{c} : 2^N \to \mathbb{R}$ の対 $(N, \tilde{c})$ である．

$$\tilde{c}(S) = \min\{c(\Gamma) \mid \Gamma \text{は } \mathrm{K}_{S'} \text{の全域木 }\} \qquad (S \subseteq N).$$

ここで, $\mathrm{K}_{S'}$ は S' によって誘導される $\mathrm{K}_{N'}$ の部分グラフである．

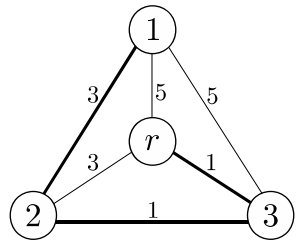

図 1: ネットワーク $(\mathrm{K}_{N'}, c)$ と最小費用全域木.

例 2.1 図 1 のネットワーク $(\mathrm{K}_{N'}, c)$ に関連する最小費用全域木ゲーム $(N, \tilde{c})$ を考える. ここで, $N = \{1, 2, 3\}$ である. $S' = \{r, 1, 2\}$ によって誘導される $\mathrm{K}_{N'}$ の部分グラフ $\mathrm{K}_{S'}$ は図 2 の破線で囲んだ部分であり, $\mathrm{K}_{S'}$ の最小費用全域木は図 2 の太線で描かれた枝から成る木である. したがって, $\tilde{c}(\{1, 2\}) = 6$ となる.

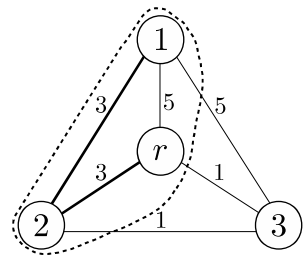

図 2: 部分グラフ $\mathrm{K}_{S'}$ と $(\mathrm{K}_{S'}, c)$ の最小費用全域木.

例 2.2 図 1 のネットワーク $(\mathrm{K}_{N'}, c)$ に関連する最小費用全域木ゲーム $(N, \tilde{c})$ を考える. 置換 $\pi = (1, 3, 2)$ に対する $(N, \tilde{c})$ の限界費用ベクトルは以下のように計算される.

$$\begin{array}{rcll} x(\pi)_1 & = & \tilde{c}(\{1\}) - \tilde{c}(\emptyset) & = & 5, \\ x(\pi)_3 & = & \tilde{c}(\{1, 3\}) - \tilde{c}(\{1\}) & = & 1, \\ x(\pi)_2 & = & \tilde{c}(\{1, 3, 2\}) - \tilde{c}(\{1, 3\}) & = & -1. \end{array}$$

同様にして, すべての $N = \{1, 2, 3\}$ の置換に対する $(N, \tilde{c})$ の限界費用ベクトルを表 1 に示す. 式 (4) より, 例 2.1 の最小費用全域木ゲーム $(N, \tilde{c})$ に対する Shapley 値は次のように計算される.

$$\mathrm{Sh}(\tilde{c}) = (\mathrm{Sh}(\tilde{c})_1, \mathrm{Sh}(\tilde{c})_2, \mathrm{Sh}(\tilde{c})_3) = (4, 1, 0).$$

最小費用全域木ゲームの Shapley 値の計算複雑度に関しては以下の結果が知られている.

命題 2.1 (Ando [1]) 最小費用全域木ゲームの Shapley 値の計算は#P-困難である.

表 1: 例 2.1 の最小費用全域木ゲーム $(N, \tilde{c})$ に対する限界費用ベクトル.

π	$x(\pi)_1$	$x(\pi)_2$	$x(\pi)_3$
$(1,2,3)$	5	1	-1
$(1,3,2)$	5	-1	1
$(2,1,3)$	3	3	-1
$(2,3,1)$	3	3	-1
$(3,1,2)$	5	-1	1
$(3,2,1)$	3	1	1

以下では, 任意の最小費用全域木ゲームが $\{0,1\}$-費用関数に関連する最小費用全域木ゲームの非負結合によって表されることを示す. この分解と Shapley 値の線形性によって任意の最小費用全域木ゲームの Shapley 値は $\{0,1\}$-費用関数に関連する最小費用全域木ゲームの Shapley 値の非負結合によって表される.

ネットワーク $(\mathrm{K}_{N'}, c)$ に対して, 相異なる正の $c(u,v)$ の値を

$$(0 <)\alpha_1 < \cdots < \alpha_l$$

と表し, $\alpha_0 = 0$ とする. 各 $i = 0, \ldots, l$ に対して, $c_i : N' \times N' \to \{0,1\}$ を

$$c_i(u,v) = \begin{cases} 1 & \text{if } \alpha_i < c(u,v), \\ 0 & \text{otherwise} \end{cases} \qquad (u, v \in N') \tag{5}$$

で定義する. このとき $\delta_i = \alpha_{i+1} - \alpha_i$ $(i = 0, \ldots, l-1), \delta_l = 0$ とおくと費用関数は以下のように分解できる.

$$c = \sum_{i=0}^{l} \delta_i c_i. \tag{6}$$

例 2.3 $(\mathrm{K}_{N'}, c)$ を図 3 で示されるようなネットワークとする. このとき, $l = 5, \delta_0 = 1, \delta_1 = 2, \delta_2 = 1, \delta_3 = 2, \delta_4 = 1, \delta_5 = 0$ であり, 式 (5) の $c_0, \ldots, c_5$ によって定義されるネットワークは図 4 の (a), . . . , (f) である.

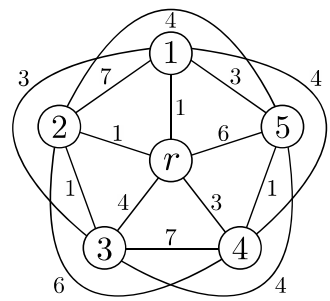

図 3: ネットワーク $(\mathrm{K}_{N'}, c)$.

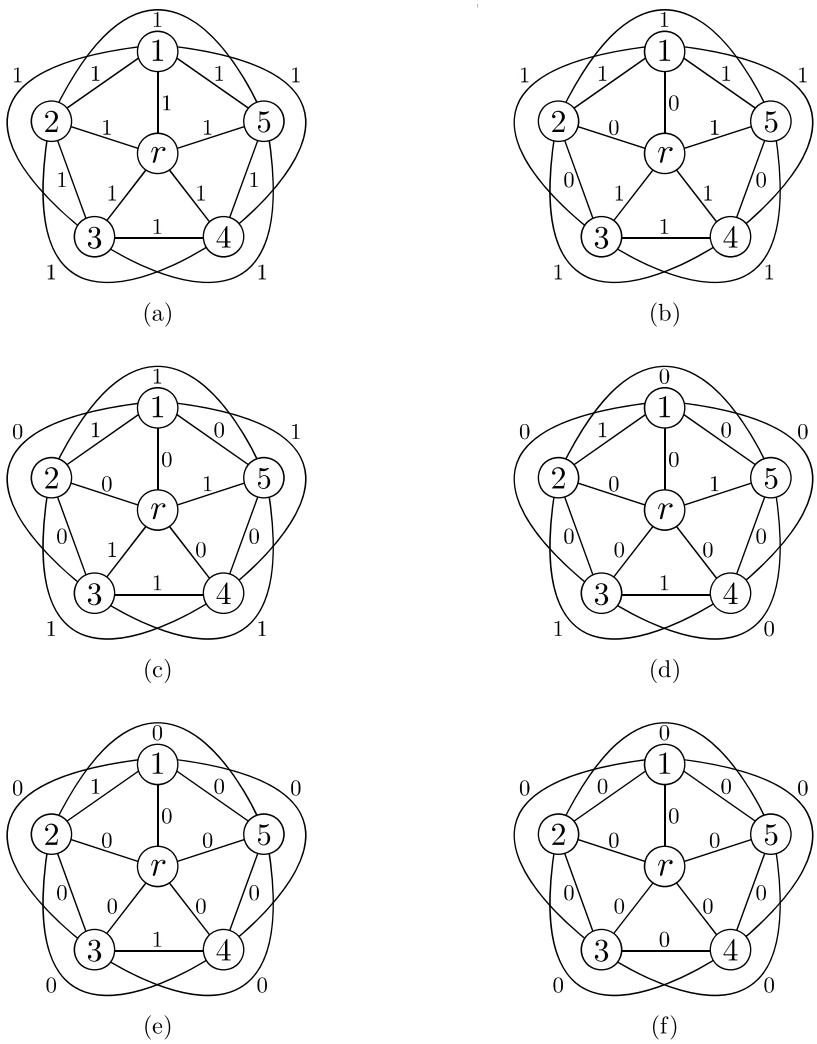

図 4: (a) $(\mathrm{K}_{N'}, c_0)$; (b) $(\mathrm{K}_{N'}, c_1)$; (c) $(\mathrm{K}_{N'}, c_2)$; (d) $(\mathrm{K}_{N'}, c_3)$; (e) $(\mathrm{K}_{N'}, c_4)$; (f) $(\mathrm{K}_{N'}, c_5)$.

補題 2.2 (Norde, Moretti and Tijs [5]) $(\mathrm{K}_{N'}, c)$ をネットワークとする．c が式 (6) のように分解されるとき，$\tilde{c} : 2^N \to \mathbb{R}$ は

$$\tilde{c} = \sum_{i=0}^{l} \delta_i \tilde{c}_i \tag{7}$$

と分解できる．

補題 2.2 と Shapley 値の線形性より以下が成り立つ．

命題 2.3 $(\mathrm{K}_{N'}, c)$ をネットワークとする．c が式 (6) のように分解されるとき，

$$\mathrm{Sh}(\tilde{c}) = \sum_{i=0}^{l} \delta_i \mathrm{Sh}(\tilde{c}_i)$$

が成り立つ．

3　近似アルゴリズム

費用関数が木距離である場合には関連する最小費用全域木ゲームは木距離最小費用全域木ゲームと呼ばれる．一般の最小費用全域木ゲームの Shapley 値の計算は#P-困難であるが，木距離最小費用全域木ゲームの Shapley 値は多項式時間で計算できることが知られている [1]．本節では，木距離最小費用全域木ゲームに対する多項式時間アルゴリズムのアイデアに基づいて，一般の最小費用全域木ゲームの Shapley 値に対する多項式時間近似アルゴリズムを導入する．

3.1　木距離最小費用全域木ゲーム

費用関数 $c : N' \times N' \to \mathbb{R}_+$ は，もし点集合が N' を含み非負の枝重みを持つ木 U が存在して，任意の $u, v \in N'$ に対して

$$c(u, v) = d_U(u, v)$$

を満たすとき，**木距離**と呼ばれる．ここで，$d_U(u, v)$ は U 中の一意的な u-v 道に含まれる枝の重みの合計である．また，このとき U は c を**表現する**と言う．図 5 に c が木距離であるネットワーク $(\mathrm{K}_{N'}, c)$ とそれを表現する木の例を示す．

各 $(u, v) \in N' \times N'$ に対して $c(u, v) \in \{0, 1\}$ であるようなネットワーク $(\mathrm{K}_{N'}, c)$ に対して，グラフ $G(c) = (N', E(c))$ を

$$E(c) = \{(u, v) \mid u, v \in N', u \neq v, c(u, v) = 0\} \tag{8}$$

によって定義する．

単純無向グラフの任意の閉路に対して，閉路内の連続しない 2 点を結ぶ枝をその閉路の**弦**と呼ぶ．全ての長さ 4 以上の閉路が弦を持つようなグラフは**コーダルグラフ**と呼ばれる．図 6 は図 4 に示した各 $\{0, 1\}$-費用関数 c_i に対する $G(c_i)$ である．$G(c_0), G(c_1), G(c_4), G(c_5)$ はコーダルであり，$G(c_2), G(c_3)$ はコーダルでない．

補題 3.1 (folklore) $(\mathrm{K}_{N'}, c)$ を c が木距離であるようなネットワークとする．c が式 (6) のように分解されるとき全ての i に対して $G(c_i)$ はコーダルグラフである．

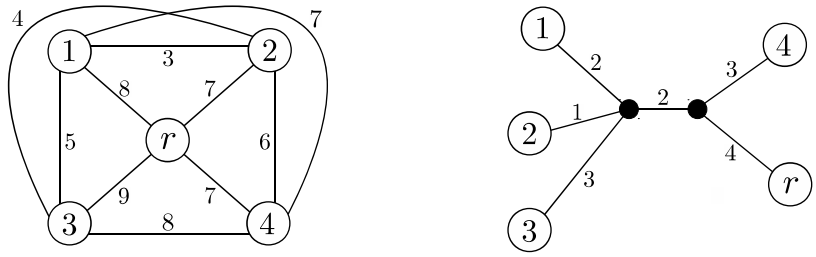

図 5: (a) c が木距離であるネットワーク $(\mathrm{K}_{N'}, c)$; (b) c を表現する木.

補題 3.2 (Ando [1]) $(\mathrm{K}_{N'}, c)$ を, すべての $(u, v) \in N' \times N'$ に対して $c(u, v) \in \{0, 1\}$ であるようなネットワークとする. $G(c)$ がコーダルならば, 最小費用全域木ゲーム $(N, \tilde{c})$ の Shapley 値は $\mathrm{O}(n^2)$ 時間で計算できる.

費用関数 c が木距離であるとき, ネットワーク $(\mathrm{K}_{N'}, c)$ に関連する最小費用全域木ゲーム $(N, \tilde{c})$ を**木距離最小費用全域木ゲーム**と呼ぶ. 命題 2.3, 補題 3.1, 補題 3.2 より以下が成り立つ.

定理 3.3 (Ando [1]) 木距離最小費用全域木ゲームの Shapley 値は $\mathrm{O}(n^4)$ 時間で計算できる.

3.2 近似アルゴリズム

この節では, 一般的な最小費用全域木ゲームの Shapley 値に対する新しいアルゴリズムを導入する. このアルゴリズムは, 補題 3.1 と補題 3.2 に基づいて, 以下のように Shapley 値の近似解を求める. まず, 費用関数 c を式 (6) のように $c = \delta_0 c_0 + \cdots + \delta_l c_l$ と分解する. 各 $i = 0, \ldots, l$ に対して, $G(c_i)$ のコーダル近似を $G'_i = (N', E'_i)$ とし, c'_i を $G'_i = G(c'_i)$ であるような $\{0, 1\}$-費用関数とする. 最後に, $\mathrm{Sh}(\widetilde{c'}) = \delta_0 \mathrm{Sh}(\widetilde{c'_0}) + \cdots + \delta_l \mathrm{Sh}(\widetilde{c'_l})$ を Shapley 値 $\mathrm{Sh}(\tilde{c})$ の近似解として出力する. アルゴリズム 1 はこのアルゴリズムの形式的な記述である.

入力: ネットワーク $(\mathrm{K}_{N'}, c)$.
出力: $(N, \tilde{c})$ の Shapley 値 $\mathrm{Sh}(\tilde{c})$ の近似解.
1 費用関数 c の分解 (6) を求める;
2 **for** すべての $i = 0, \ldots, l$ **do**
3 $\quad$ $G(c_i)$ のコーダル近似 $G'_i = (N', E'_i)$ を求める;
4 $\quad$ c'_i を $G(c'_i) = G'_i$ であるような $\{0, 1\}$-費用関数とする;
5 $\quad$ 最小費用全域木ゲーム $(N, \widetilde{c'_i})$ の Shapley 値 $\mathrm{Sh}(\widetilde{c'_i})$ を計算する;
6 **end**
7 $\mathrm{Sh}(\widetilde{c'}) \leftarrow \delta_0 \mathrm{Sh}(\widetilde{c'_0}) + \cdots + \delta_l \mathrm{Sh}(\widetilde{c'_l})$

アルゴリズム 1: Shapley 値の近似アルゴリズム.

アルゴリズム 1 の 3 行目において $G(c_i)$ のコーダル近似 G'_i を計算する 2 つの方法を以下に示す. それぞれコーダル近似アルゴリズム A, コーダル近似アルゴリズム B と呼ぶ.

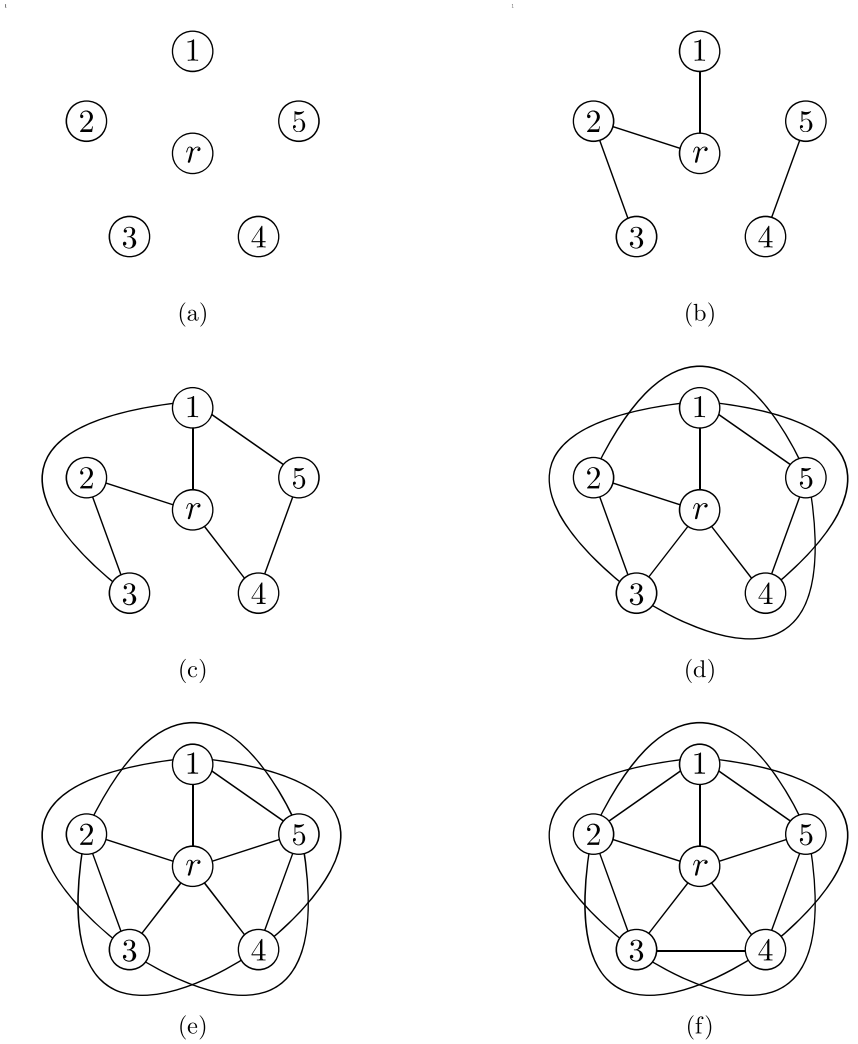

図 6: (a) $G(c_0)$; (b) $G(c_1)$; (c) $G(c_2)$; (d) $G(c_3)$; (e) $G(c_4)$; (f) $G(c_5)$. $G(c_0), G(c_1), G(c_4), G(c_5)$ はコーダルであり, $G(c_2), G(c_3)$ はコーダルでない.

コーダル近似アルゴリズム A によって求められる G'_i は, 各 $i = 0, \ldots, l$ に対して以下の条件を満たす.

(A1) $G'_{i-1} \subseteq G'_i$.

(A2) G'_i は $G'_i \subseteq G(c_i)$ なる極大コーダルグラフ.

ここで, $G'_{-1} = (N', \emptyset)$ である. もし $G(c_i)$ がコーダルならば, (A2) より $G'_i = G(c_i)$ である. アルゴリズム 2 にコーダル近似アルゴリズム A を示す. コーダル近似アルゴリズム A での G'_i の計算は $G(c_i)$ と G'_{i-1} を入力としているため, コーダル近似アルゴリズム A を用いるアルゴリズム 1 における for ループの i は 0 から l へと動く.

```
   入力: G(c_i) = (N', E(c_i)), G'_{i-1} = (N', E'_{i-1}).
   出力: (A1) と (A2) を満たす G'_i = (N', E'_i).
 1 G'_i ← G'_{i-1};
 2 do
 3 |   flag ← 0;
 4 |   for e ∈ E(c_i) − E'_i do
 5 |   |   if G'_i + e がコーダルグラフ then
 6 |   |   |   G'_i ← G'_i + e;
 7 |   |   |   flag ← 1;
 8 |   |   end
 9 |   end
10 while flag = 1;
```

アルゴリズム 2: コーダル近似アルゴリズム A.

例 3.1 $(\mathrm{K}_{N'}, c)$ を図 3 で示されるようなネットワークとする. このとき, $G(c_0), \ldots, G(c_5)$ は図 6 のようになる. 図 7 の $G'_0, \ldots, G'_5$ はコーダル近似アルゴリズム A で計算した $G(c_0), \ldots, G(c_5)$ のコーダル近似である. $i = 0, 1, 4, 5$ のときは $G(c_i)$ はコーダルであるから $G'_i = G(c_i)$ である. $i = 2, 3$ のときは $G(c_i)$ はコーダルでないから $G'_i \subset G(c_i)$ である.

命題 3.4 コーダル近似アルゴリズム A を用いたアルゴリズム 1 の実行時間は $\mathrm{O}(n^5)$ である.

(証明) 最初にコーダル近似アルゴリズム A (アルゴリズム 2) の計算時間について考える. 5 行目の $G'_i + e$ がコーダルグラフかどうかの判定は $\mathrm{O}(n)$ 時間で実行できる [4]. do-while ループの反復は高々 $(|E'_i| - |E'_{i-1}|)$ 回行われ, for ループの反復は高々 n^2 回であることから, コーダル近似アルゴリズム A は G'_i を $\mathrm{O}(n^3)(|E'_i| - |E'_{i-1}|)$ 時間で計算する.

補題 3.2 より $\mathrm{Sh}(\widetilde{c'_i})$ は $\mathrm{O}(n^2)$ 時間で計算できるため, アルゴリズム 1 の各反復は $\mathrm{O}(n^3)(|E'_i| - |E'_{i-1}|)$ 時間で実行できる. $|E'_l| = \frac{n(n+1)}{2}, |E'_{-1}| = 0$ であることから, アルゴリズム 1 の実行時間は $\sum_{i=0}^{l} \mathrm{O}(n^3)(|E'_i| - |E'_{i-1}|) = \mathrm{O}(n^5)$ である. □

コーダル近似アルゴリズム B によって求められる G'_i は, 各 $i = 0, \ldots, l$ に対して以下の条件を満たす.

(B1) $G'_i \subseteq G'_{i+1}$.

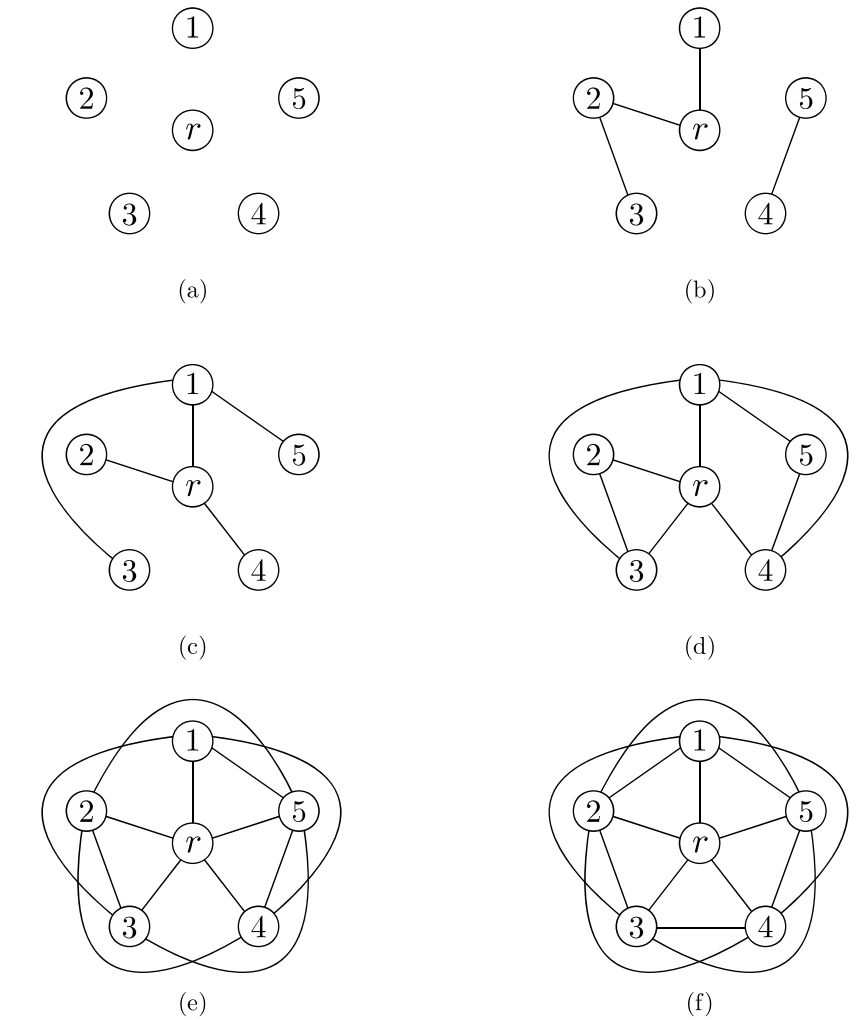

図 7: (a) G'_0; (b) G'_1; (c) G'_2; (d) G'_3; (e) G'_4; (f) G'_5.

(B2) G'_i は $G(c_i) \subseteq G'_i$ なる極小コーダルグラフ.

ここで, $G'_{l+1} = \mathrm{K}_{N'}$ である. もし $G(c_i)$ がコーダルならば, (B2) より $G'_i = G(c_i)$ である. アルゴリズム 3 にコーダル近似アルゴリズム B を示す. コーダル近似アルゴリズム B での G'_i の計算は $G(c_i)$ と G'_{i+1} を入力としているため, コーダル近似アルゴリズム B を用いるアルゴリズム 1 における for ループの i は l から 0 へと動く.

```
   入力: G(c_i) = (N', E(c_i)), G'_{i+1} = (N', E'_{i+1}).
   出力: (B1) と (B2) を満たす G'_i = (N', E'_i).
 1 G'_i ← G'_{i+1};
 2 do
 3 |   flag ← 0;
 4 |   for e ∈ E'_i − E(c_i) do
 5 |   |   if G'_i − e がコーダルグラフ then
 6 |   |   |   G'_i ← G'_i − e;
 7 |   |   |   flag ← 1;
 8 |   |   end
 9 |   end
10 while flag = 1;
```

アルゴリズム 3: コーダル近似アルゴリズム B.

例 3.2 $(\mathrm{K}_{N'}, c)$ を図 3 で示されるようなネットワークとする. このとき, $G(c_0), \ldots, G(c_5)$ は図 6 のようになる. 図 8 の $G'_0, \ldots, G'_5$ はコーダル近似アルゴリズム B で計算した $G(c_0), \ldots, G(c_5)$ のコーダル近似である. $i = 0, 1, 4, 5$ のときは $G(c_i)$ はコーダルであるから $G'_i = G(c_i)$ である. $i = 2, 3$ のときは $G(c_i)$ はコーダルでないから $G(c_i) \subset G'_i$ である.

$G'_i - e$ がコーダルグラフかどうかの判定を $\mathrm{O}(n)$ 時間で計算するアルゴリズムが存在する [4] ため, 命題 3.4 と同様にして以下の命題を得る.

命題 3.5 コーダル近似アルゴリズム B を用いたアルゴリズム 1 の実行時間は $\mathrm{O}(n^5)$ である.

4 数値実験

この節では, 第 3 節で導入した Shapley 値の近似アルゴリズム (アルゴリズム 1) を実装し, このアルゴリズムの近似精度を数値実験によって検証する. アルゴリズム 1 の実装は C 言語で行い, 実験は以下の環境で行った.

- コンパイラ: gcc version 4.8.4
- OS: Ubuntu 14.04.5
- CPU: Intel(R) Core(TM) i7-4770
- Memory: 7.5GB

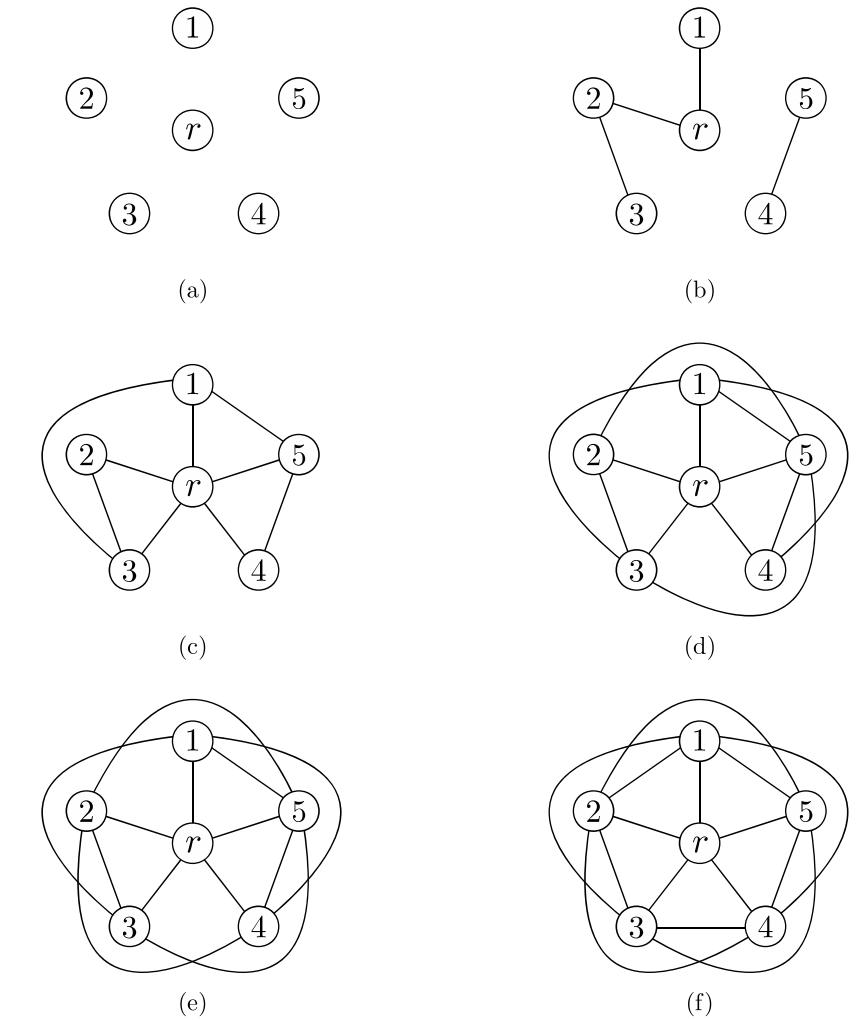

図 8: (a) G'_0; (b) G'_1; (c) G'_2; (d) G'_3; (e) G'_4; (f) G'_5.

実験の内容は, 第 3 節で導入した 2 種類のコーダル近似アルゴリズムを用いたアルゴリズム 1 による Shapley 値の近似解 $\mathrm{Sh}(\widetilde{c'})$ の相対誤差の比較を行うというものである. 相対誤差は,

$$\frac{\|\mathrm{Sh}(\tilde{c}) - \mathrm{Sh}(\widetilde{c'})\|_2}{\|\mathrm{Sh}(\tilde{c})\|_2}$$

で計算される. ここで, $\mathrm{Sh}(\tilde{c})$ は真の Shapley 値である. ただし, 真の Shapley 値を求めることは現実的に困難であるため, サンプリング [3] を用いて求めた値を真の Shapley 値 $\mathrm{Sh}(\tilde{c})$ とみなして相対誤差を計算した. コーダル近似アルゴリズム A を用いたアルゴリズム 1 をアルゴリズム A と呼び, コーダル近似アルゴリズム B を用いたアルゴリズム 1 をアルゴリズム B と呼ぶ. また, 本論文で導入した近似アルゴリズムの近似精度を相対的に評価するために, 与えられた費用関数を木距離によって近似することで Shapley 値の近似解を得る手法 [9] による結果も示す. このとき, 与えられた費用関数を木距離によって近似する方法として Neighbor Joining (NJ) 法 [7] を採用した. この Shapley 値近似アルゴリズムを NJ アルゴリズムと呼ぶ.

4.1 実験で用いた入力

実験では, 近似アルゴリズムに対する入力として以下の 3 種類の費用関数を用いた.

一つ目はランダムに生成された 2 次元ユークリッド距離である. 各 $v \in N'$ に対して x_v, y_v を $[0, \lfloor \frac{n(n+1)}{2\sqrt{2}} \rfloor]$ 上の一様整数乱数とする. ランダムな 2 次元ユークリッド距離 c は以下の式で定義される.

$$c(u,v) = \|(x_u, y_u) - (x_v, y_v)\|_2 \quad (u, v \in N').$$

二つ目は木距離を摂動させた費用関数である. この費用関数は, まず N' を葉集合として持つ木 U をランダムに構築し, 枝の費用を $[0, \frac{n(n+1)}{2}]$ 上の一様整数乱数で与える. これによって得た木距離 d_U を以下の式でスケーリングして c^* を得る.

$$c^*(u,v) = \frac{\frac{n(n+1)}{2}}{\max\limits_{p,q \in N'} d_U(p,q)} d_U(u,v) \quad (u, v \in N').$$

次に, $\mathrm{K}_{N'}$ のすべての枝の中からランダムに選んだ半数の枝を E^* として, $c\colon N' \times N' \to \mathbb{R}$ を

$$c(u,v) = \begin{cases} c^*(u,v) \times \delta(u,v) & \text{if } (u,v) \in E^*, \\ c^*(u,v) & \text{otherwise} \end{cases} \quad (u, v \in N') \tag{9}$$

によって定義する. ここで, $\delta(u,v)$ は $[0.95, 1.05]$ 上の一様実数乱数である.

三つ目は一様ランダムな費用関数である. この費用関数 c は, 各 $u, v \in N'$ $(u \neq v)$ に対して, $c(u,v)$ に $[0, \frac{n(n+1)}{2}]$ 上の一様整数乱数を割り当てたものである.

4.2 結果と考察

3 種類の各入力に対して, $n = 20, 40, 60, 80, 100$ とした場合についてそれぞれ 20 個の問題を用意した. その 20 個の問題に対して, 各アルゴリズムが出力した解の相対誤差の平均値を比較する.

最初に, 2 次元ユークリッド距離を入力としたときの Shapley 値の近似に関する結果を図 9 に示す. アルゴリズム B, アルゴリズム A, NJ アルゴリズムの順で精度が高いことがわかる. プレイ

ヤー数の増加に対して, 各アルゴリズムの出力の相対誤差の増加率に大きな差は無かった. アルゴリズム B に関しては, プレイヤー数が 100 のとき, 相対誤差の平均値は 14% 程度であった.

次に, 木距離を摂動させた費用関数を入力としたときの Shapley 値の近似に関する結果を図 10 に示す. アルゴリズム A とアルゴリズム B の精度は, ほぼ同じであり NJ アルゴリズムよりも高いことがわかる. プレイヤー数の増加に対して, 各アルゴリズムの出力の相対誤差の平均値はほぼ一定であった.

最後に, 一様ランダムな費用関数を入力としたときの Shapley 値の近似に関する結果を図 11 に示す. アルゴリズム B, NJ アルゴリズム, アルゴリズム A の順で精度が高いことがわかる. アルゴリズム A とアルゴリズム B を比較すると, プレイヤー数の増加に対するアルゴリズム A の出力の相対誤差の平均値の増加率は高く, アルゴリズム B の出力の相対誤差の平均値の増加率は低かった. アルゴリズム B に関しては, プレイヤー数が 20 のとき相対誤差の平均値は 36% 程度であり, プレイヤー数が 100 のとき相対誤差の平均値は 58% 程度であった.

実験の結果から他の手法と比較すると, アルゴリズム B の出力はほぼ全ての入力に対して精度が高いことがわかる. また, 入力が木距離に近ければ相対誤差は小さく, 一様ランダムに近ければ相対誤差は大きくなることが観察された. これらの観察から筆者らは, 近似アルゴリズムによって得られる近似関数 c' の c に対する近似精度が高ければ, Shapley 値の近似精度も高くなるのではないかと考えた. この予想を検証するために, 入力の費用関数 c に対する 3 種類の近似アルゴリズムによって得られる近似関数 c' の相対誤差の平均値を比較した. c' の相対誤差は,

$$\frac{\|c - c'\|_2}{\|c\|_2}$$

で計算される. 入力 c は Shapley 値の近似精度を検証するために生成した費用関数を用いた.

最初に, 2 次元ユークリッド距離に関する結果を図 12 に示す. Shapley 値の近似精度はアルゴリズム B, アルゴリズム A, NJ アルゴリズムの順で高かったのに対して, 近似関数の近似精度はアルゴリズム A, アルゴリズム B, NJ アルゴリズムの順で高かった. プレイヤー数の増加に対して, 各アルゴリズムによって得られる近似関数の相対誤差の増加率に大きな差は無かった.

次に, 木距離を摂動させた費用関数 c に関する結果を図 13 に示す. Shapley 値の近似精度はアルゴリズム A とアルゴリズム B の精度はほぼ同じであり NJ アルゴリズムよりも高かったのに対して, 近似関数の近似精度はアルゴリズム A, アルゴリズム B, NJ アルゴリズムの順で高かった. また, プレイヤー数の増加に対してアルゴリズム A と NJ アルゴリズムによって得られる近似関数の相対誤差の平均値はほぼ一定であるのに対して, アルゴリズム B によって得られる近似関数の相対誤差の平均値は単調増加している.

最後に, 一様ランダムな費用関数 c に関する結果を図 14 に示す. Shapley 値の近似精度はアルゴリズム B, NJ アルゴリズム, アルゴリズム A の順で高かったのに対して, 近似関数の近似精度はプレイヤー数が 20 のときを除いて NJ アルゴリズム, アルゴリズム A, アルゴリズム B の順で高かった. プレイヤー数の増加に対して, 各アルゴリズムによって得られる近似関数の相対誤差の増加率に大きな差は無かった.

Shapley 値の近似精度に関する実験結果と近似関数の近似精度に関する実験結果を比較すると, アルゴリズムの近似精度の優劣が異なっていた. さらに, 木距離を摂動させた費用関数や一様ランダムな費用関数を入力としたとき, プレイヤー数の増加に対して, 近似関数 c' と Shapley 値の近似解 $\mathrm{Sh}(\widetilde{c'})$ の相対誤差の平均値の増加傾向に違いが見られた. よって, 近似アルゴリズムによって得られる近似関数の近似精度と Shapley 値の近似精度との間に相関は存在しないと結論付けられる.

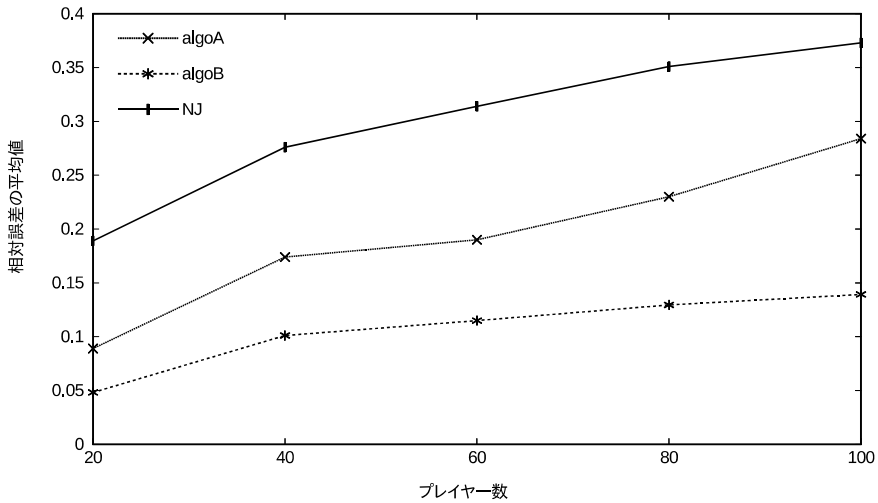

図 9: 2 次元ユークリッド距離 c に対する $\frac{\|\mathrm{Sh}(\tilde{c})-\mathrm{Sh}(\tilde{c'})\|_2}{\|\mathrm{Sh}(\tilde{c})\|_2}$ の平均値.

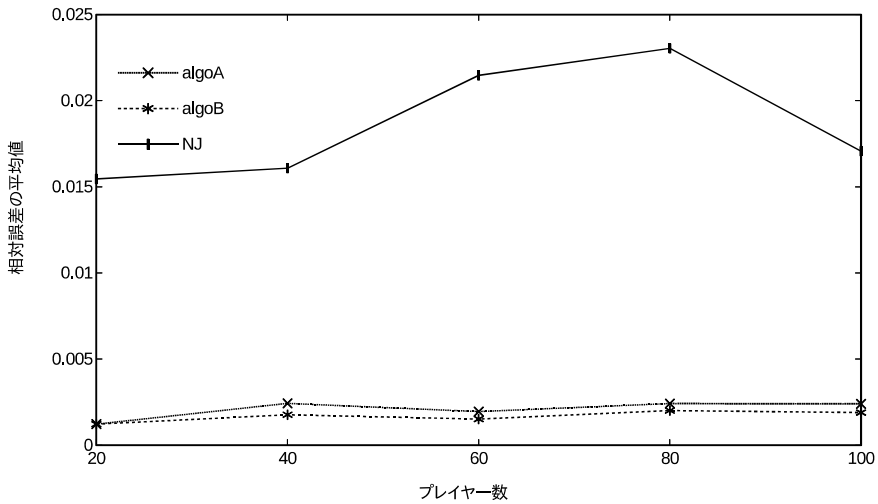

図 10: 木距離を摂動させた費用関数 c に対する $\frac{\|\mathrm{Sh}(\tilde{c})-\mathrm{Sh}(\tilde{c'})\|_2}{\|\mathrm{Sh}(\tilde{c})\|_2}$ の平均値.

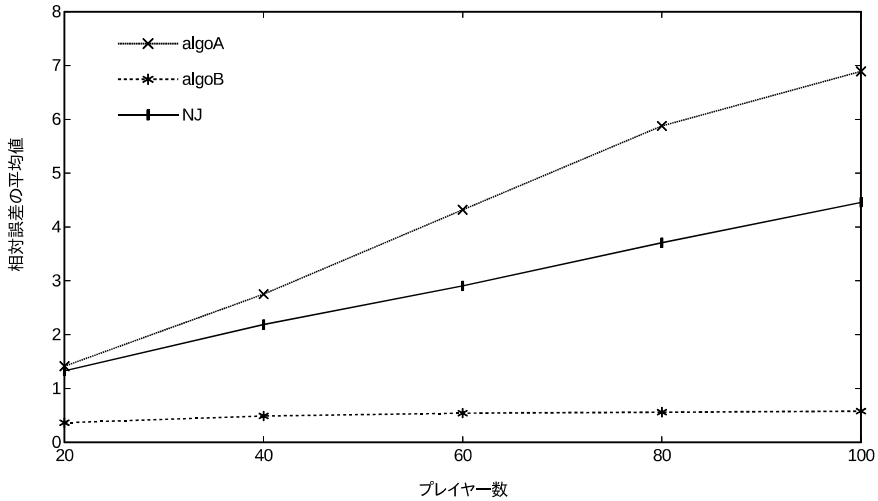

図 11: 一様ランダムな費用関数 c に対する $\frac{\|\mathrm{Sh}(\tilde{c})-\mathrm{Sh}(\widetilde{c'})\|_2}{\|\mathrm{Sh}(\tilde{c})\|_2}$ の平均値.

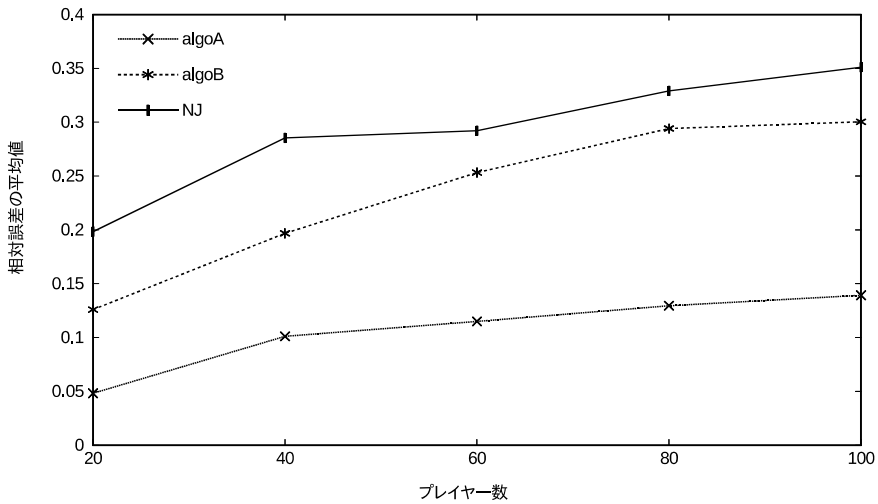

図 12: 2 次元ユークリッド距離 c に対する $\frac{\|c-c'\|_2}{\|c\|_2}$ の平均値.

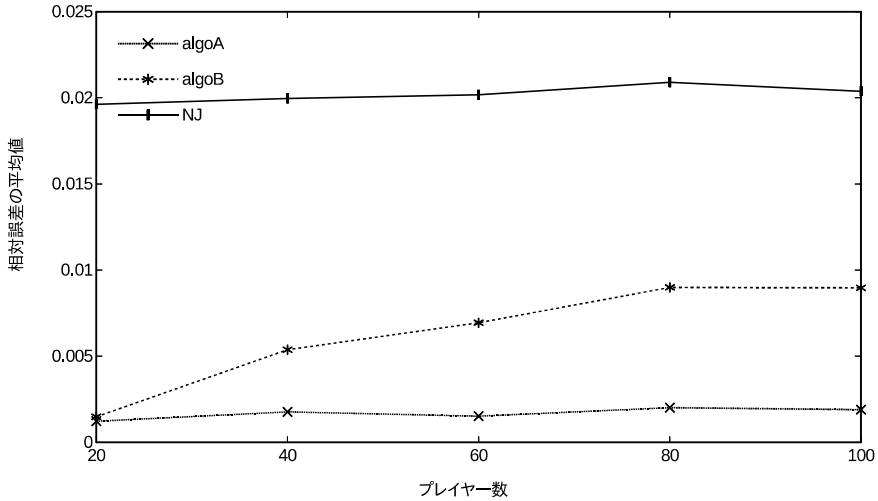

図 13: 木距離を摂動させた費用関数 c に対する $\frac{\|c-c'\|_2}{\|c\|_2}$ の平均値.

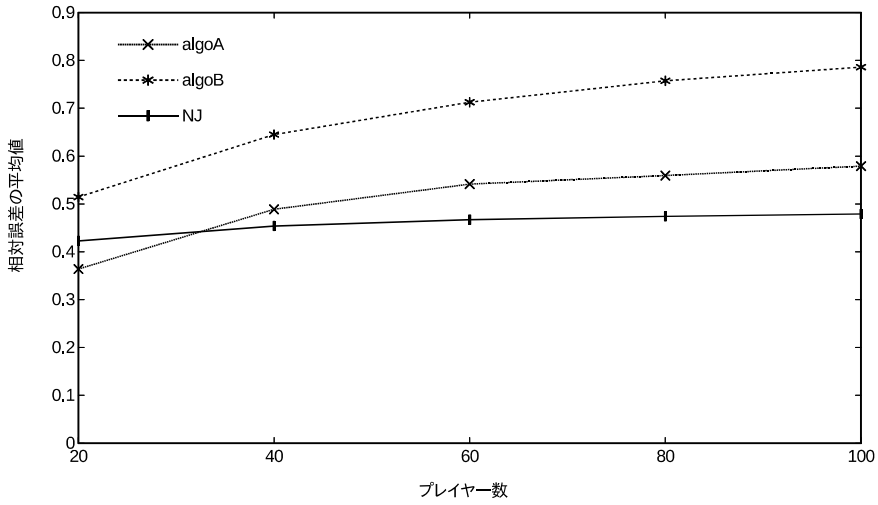

図 14: 一様ランダムな費用関数 c に対する $\frac{\|c-c'\|_2}{\|c\|_2}$ の平均値.

5 おわりに

最小費用全域木ゲームの Shapley 値の計算は#P-困難である [1]. 最小費用全域木ゲームの Shapley 値の近似アルゴリズムとして, 一般的な協力ゲームの Shapley 値に対する近似アルゴリズムであるサンプリング・アルゴリズム [3, 9] が知られているが, 要求される近似精度を達成するためのサンプル数が非常に大きいため近似解の計算のためには膨大な時間を要する.

最小費用全域木ゲームは, それを定義するネットワークの費用関数が木距離である場合には木距離最小費用全域木ゲームと呼ばれ, 木距離最小費用全域木ゲームの Shapley 値は $\mathrm{O}(n^4)$ 時間で計算できることが知られている [1]. ここで, n はプレーヤーの数である. 木距離最小費用全域木ゲームに対する多項式時間アルゴリズムのアイデアに基づいて, 本研究では一般の最小費用全域木ゲームの Shapley 値に対する近似アルゴリズムを導入した. 費用関数 c が与えられたときに c に関連する最小費用全域木ゲームの Shapley 値 $\mathrm{Sh}(\tilde{c})$ に対して, このアルゴリズムによる近似値は, c を近似する別の費用関数 c' に関連する最小費用全域木ゲームの Shapley 値 $\mathrm{Sh}(\widetilde{c'})$ である. ここで, c' は以下のように求められる. まず c を $\{0,1\}$-費用関数 c_i $(i=0,\ldots,l)$ の非負結合に分解する. $c=\sum_{i=0}^{l}\delta_i c_i$. そして, 各 $i=0,\ldots,l$ に対して c_i を c'_i によって近似する. ここで, c'_i は $G(c'_i)$ がコーダルグラフであるように選ばれる. c' は $c'=\sum_{i=0}^{l}\delta_i c'_i$ によって与えられる.

本論文ではさらに, 導入した近似アルゴリズムの近似精度を評価するために, ランダムに生成した問題例を入力として近似アルゴリズムの出力の相対誤差を求める数値実験を行った. その結果, 各 i に対して $G(c'_i)$ が $G(c_i)$ を含む極小なコーダルグラフであるように c'_i を選ぶ近似アルゴリズムが全ての入力に対して最も高い近似精度を達成した. 例えば与えられた費用関数が 2 次元ユークリッド距離の場合の相対誤差は最大でも 14%程度であった.

本研究で導入した近似アルゴリズムの欠点は理論的な精度保証を持たないことである. つまり, アルゴリズムが出力する Shapley 値の近似値がどれだけの誤差を持つのかということが分からないことである. そのような精度保証の可能性を探るために, 入力費用関数 c に対する近似関数 c' の誤差と近似値 $\mathrm{Sh}(\widetilde{c'})$ の誤差との相関を数値実験によって検証したが, そのような相関は観察されなかった. 入力として与えられた費用関数が特殊な場合, 例えば三角不等式を満たす場合, 本研究で導入した近似アルゴリズムの精度保証を導出することは今後の課題である.

謝辞

本研究は JSPS 科研費 15K00033 の助成を受けたものである.

参考文献

[1] K. Ando: Computation of the Shapley value of minimum cost spanning tree games: #P-hardness and polynomial cases. *Japan Journal of Industrial and Applied Mathematics* **29** (2012) 385–400.

[2] C. G. Bird: On cost allocation for a spanning tree. A game theoretic approach. *Networks* **6** (1976) 335–350.

[3] J. Castro, D. Gómez and J. Tejada: Polynomial calculation of the Shapley value based on sampling. *Computer and Operations Research* **36** (2009) 1726–1730.

[4] L. Ibarra: Fully dynamic algorithms for chordal graphs and split graphs. *ACM Transactions on Algorithms* **4** (2008) 1–20.

[5] H. Norde, S. Moretti and S. Tijs: Minimum cost spanning tree games and population monotonic allocation schemes. *European Journal of Operational Research* **154** (2004) 84–97.

[6] R. C. Prim: Shortest connection networks and some generalizations. *Bell System Technical Journal* **36** (1957) 1389–1401.

[7] N. Saito and M. Nei: The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* **4** (1987) 406–425.

[8] L. S. Shapley: A value for n-person games. In: H. W. Kuhn and A. W. Tucker (eds.): *Contributions to the Theory of Games, volume II* (Princeton University Press, 1953), pp. 307–317.

[9] 徳武忠俊: 最小費用全域木ゲームに対する Shapley 値の近似. 修士論文. 静岡大学工学研究科システム工学専攻, 2013.