1D4-OS-02a-5

分散MaxSATに対する相関均衡点の求め方に関する一検討

Towards Computing a Correlated Equilibrium for Distributed MaxSAT

波多野大督 *1　花田研太 *2
Daisuke HATANO　Kenta HANADA

*1国立情報学研究所　ビッグデータ数理国際研究センター
National Institute of Infomatics, Global Research Center for Big Data Mathematics

*2神戸大学大学院海事科学研究科
Graduate School of Maritime Sciences, Kobe University

Correlated equilibrium, a solution concept in the game theory, is an important notion to securely make a decision in multi-agent systems. Thus, a way of compute the equilibrium in the distributed manner is needed. We tackle this problem through distributed MaxSAT problems, where each variable is separately controlled by an agent. In this paper, to compute a correlated equilibrium in the distributed manner, we propose a new algorithm based on Black box reduction algorithm, which is a kind of no-swap regret algorithms. By using the algorithm, we can produce a solution arbitrary close to the correlated equilibrium.

1. はじめに

近年のソーシャルネットワークの発達や共有型経済の発展により個人が市場に参入することが容易になっており，それに伴い分散環境における最適化の重要性が増している．本稿では，分散 MaxSAT の観点からこの問題に取り組む．SAT（充足可能性判定問題）は判定問題の一種で，様々な問題の理論的解析に用いられるだけでなく人工知能に関連する多くの応用問題を表現できる重要な問題である．MaxSAT（最大充足可能性問題）は，SAT を最適化問題として拡張したもので，充足不可能な問題に対して可能な限り制約を満たす割当を探す問題である．分散 MaxSAT は MaxSAT を分散環境に拡張した問題で，計算資源が分散された状況における計算や 1 つの計算資源では計算できないような巨大な問題を表現できる．一般的に，分散 MaxSAT では，エージェントが存在し互いに情報交換をしあい最適解を求める．

分散 MaxSAT の枠組では，基本的にそれぞれのエージェントが協力しあうことを前提としているが，現実世界の応用例を考慮した場合，必ずしも協力するとは限らない．例えば，ある種の問題では，セキュリティの観点から自身の情報をあまり共有したくないかもしれない．また別の問題では，いくつかのエージェントは全体の利益を鑑みずに自身の利益のみを追求するかもしれない．このような問題設定に対応するために，本稿では，ゲーム理論の解概念の一種である相関均衡点について考える．

相関均衡点 [Aumann 74] は，ゲーム理論の重要な解概念の 1 つで，ナッシュ均衡を一般化したものである．相関均衡点を直観的に説明するために，以下のゲームを考える．今，2 人の運転手がそれぞれ別の方向から交差点に向かい進んでいるとする．運転手は交差点において，進むと止まるの 2 つの戦略を持つ．このゲームのコストを以下のとおりに定めるとする．

- 両ドライバーが共に進むを選択した場合，交差点内で衝突してしまい，5 のコストが発生する．
- 両ドライバーが止まるを選択した場合，進めなくなり，1 のコストが発生する．
- 互いに異なる戦略を選択した場合，衝突が発生しないため，コストが発生しない．

このゲームでは，2 つの純粋ナッシュ均衡（互いに異なる戦略をとる場合）と 1 つの混合ナッシュ均衡（止まると進むを確率 1/2 で選択する場合）を達成する確率分布が存在する．一方で，戦略の組｛進む，止まる｝と｛止まる，進む｝に対してそれぞれ確率 1/2 で選択するような確率分布を考えると，これは純粋ナッシュ均衡でも混合ナッシュ均衡でもないことがわかる．また，この確率分布は上記の例では信号機（信頼のおける第三者）の振る舞いに対応する．つまり，信号機が確率 1/2 でどちらかの戦略の組を選択している限り，ドライバーはその選択された戦略から離反する動機を持たないことを表している．このような均衡点を相関均衡点と呼ぶ．セキュリティの観点において相関均衡点は良い性質をもつ．なぜなら，運転手は信号機さえ見ていれば，他の運転手の戦略を見る必要がないためである．つまり，信頼のおける第三者が存在しその第三者が提示する戦略に従っていれば，情報が洩れることなく問題を解くことができると言える．

これまでに相関均衡点を利用したセキュアに意思決定をするためのプロトコルについては提案されているが [Dodis 00]，相関均衡点を分散環境で効率的に計算する方法についてはこれまで提案されていない．そこで，我々は分散 MaxSAT における相関均衡点を分散環境で求める方法としてブラックボックスリダクションアルゴリズム [Nisan 07] に基づくアルゴリズムを提案し，その理論的解析に関して検討する．

2. 準備

太文字はベクトルとして使用する．二つのベクトル $\boldsymbol{x}$ と $\boldsymbol{y}$ に対して，$\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ は内積を表す．つまり，$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_i x_i \cdot y_i$ である．値域 D 上のある確率分布 $\mathcal{P}$ に対して，$a \sim \mathcal{P}$ は分布 $\mathcal{P}$ から値 $a \in D$ をサンプリングすることを意味する．$\mathbf{E}$ は期待値の計算を表すものとする．

連絡先: 波多野大督，国立情報学研究所，東京都千代田区一ツ橋 2-1-2，03-4212-2664，hatano@nii.ac.jp

2.1 分散 MaxSAT

MaxSAT の問題例は $\phi = (X, C, D, f)$ で定義される．X は命題変数の集合 $X = \{x_1, x_2, \ldots, x_n\}$ で，命題変数 x とその否定 $\neg x$ を総称したものをリテラルと呼ぶ．C は節集合 $C = \{C_1, C_2, \ldots, C_m\}$ を表し，各節 C_i は $y_i \in \{x_i, \neg x_i\}$ とすると，リテラルの論理積 $C_i = y_{i1} \vee y_{i2} \vee \cdots \vee y_{ik}$ で表現される. また，節 C_i に含まれるリテラルの個数を $\alpha(C_i)$ とする．D は値域集合 $D = \{D_1, D_2, \ldots, D_n\}$ を表し，各リテラル x_i は，真を 1，偽を 0 とすると，値域 $D_i = \{0, 1\}$ をとる．f は節にかかるコストを返す関数 $f : C \to \mathbb{R}^+$ である．ある割当を $\boldsymbol{x} \in D := D_1 \times D_2 \times \cdots \times D_n$，割当 $\boldsymbol{x}$ により充足さない節集合を $C_{\text{unsat}}(\boldsymbol{x})$ とすると，MaxSAT の目的はコストの総和 $f(\boldsymbol{x}) := \sum_{C \in C_{\text{unsat}}(\boldsymbol{x})} f(C)$ が最小となる割当 $\boldsymbol{x}$ の探索である．

分散 MaxSAT は MaxSAT を分散環境に拡張した問題で，[Yokoo 98] で提案された分散制約充足問題を MaxSAT に限定した問題とみなすことができる. 分散 MaxSAT では，各命題変数をエージェントとみなす変数集合分割型と各節をエージェントとみなす節集合分割型があるが，本稿では，特に変数集合分割型の分散 MaxSAT を扱う．つまり，各命題変数はエージェントによりそれぞれ独立に意思決定がなされる．分散 MaxSAT では，エージェントはラウンドに基づき同期的に行動するものとする．あるエージェント $i \in \{1, \ldots, n\}$ に対して，節を共有するエージェントの集合を $NB(i)$ とする．あるラウンドにおいて，エージェント i は変数 x_i を管理し，節を共有する他のエージェント $j \in NB(i)$ と情報を送受信することができる．エージェント i がある割当 $\boldsymbol{x}$ を与えられたときのコストを $f_i(\boldsymbol{x}) = \sum_{C \in C_{\text{unsat}}(\boldsymbol{x})} \frac{1}{\alpha(C)} f(C)$ とすると，分散 MaxSAT の目的はコストの総和 $\sum_i f_i(\boldsymbol{x})$ を最小にする割当 $\boldsymbol{x}$ を探索することである．

例 1. *MaxSAT* 問題例

- 変数集合 $X = \{x_1, x_2, x_3\}$，で各変数 $x_i \in X$ の値域 $D_i = \{0, 1\}$,
- 節集合 $C_1 = \neg x_1, C_2 = x_1 \vee x_2, C_3 = \neg x_2 \vee x_3, C_4 = \neg x_3 \vee x_1$ で節 $C_i \in C$ にかかるコスト関数 $f(C_i) = 1$.

2.2 相関均衡点

戦略型ゲームを $G = \{n, D, f\}$ と定義する．ここで，n はプレイヤーの数を表す．つまり，プレイヤーの集合は $\{1, 2, \ldots, n\}$ である．$D = \{D_1, \ldots, D_n\}$ は戦略集合を表し，ある戦略の組 $\boldsymbol{x} \in D_1 \times D_2 \times \cdots \times D_n$ に対してコストを返す関数を $f : D \to \mathbb{R}$ とする．

相関均衡点は第三者から与えられた戦略の組に対して，どのプレイヤーもその戦略の組から離反する動機を持たない状態を指す．これは言い換えると，与えられた戦略の組に対して，どのプレイヤーも戦略を変更することによりコストを小さくできない状態である．

相関均衡点の計算は，以下の制約を満たす確率分布 $\boldsymbol{p} \in [0,1]^{|D_1| \times \cdots \times |D_n|}$ を探すことに対応する．各プレイヤー i に対して，任意の 2 つの戦略 $a, a' \in D_i$ を考える．

$$\sum_{\boldsymbol{x}_{-i} \in \boldsymbol{X}_{-i}} (f_i(a, \boldsymbol{x}_{-i}) - f_i(a', \boldsymbol{x}_{-i})) \cdot p_{a, \boldsymbol{x}_{-i}} \geq 0$$

ここで，$p_{a,x_{-i}}$ はプレイヤー i が戦略 a を他のプレイヤーが戦略 x_{-i} を選択する確率を表し，$\boldsymbol{X}_{-i}$ はプレイヤー i を除く他のプレイヤーの戦略の組の全体集合を表す．

Algorithm 1 乗算型重み更新法

Fix $\eta \leq 1/2$ and set $\boldsymbol{w}_i^1 \leftarrow (1, 1, \ldots, 1)$.
for $t = 1, 2, ..., T$ **do**
  Choose decision a with probability $\boldsymbol{x}_a^t := \frac{\boldsymbol{w}_a^t}{\|\boldsymbol{w}^t\|_1}$.
  Observe the costs of the decisions $\boldsymbol{c}_1^t, \ldots, \boldsymbol{c}_d^t$.
  For every decision a, set $\boldsymbol{w}_a^{t+1} \leftarrow \boldsymbol{w}_a^t(1 - \eta \boldsymbol{c}_a^t)$.

そこで，プレイヤー i に対する任意の 2 つの戦略をスイッチ関数 $\delta_i : D_i \to D_i$ で表現すると，戦略型ゲームの相関均衡点は次のように定義される．

定義 1 (相関均衡点 [Aumann 74])**.** 戦略型ゲームを $G = \{n, D, f\}$ に対して，戦略の組 D 上の確率分布 $\mathcal{P}$ が相関均衡点であるとは，すべてのプレイヤー i と任意のスイッチ関数 δ_i に対して，

$$\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{P}}[f_i(\boldsymbol{x})] \geq \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{P}}[f_i(\delta_i(x_i), \boldsymbol{x}_{-i})]$$

が成立することである．ここで，x_{-i} はプレイヤー i の戦略 x_i を除いた戦略の組 $x_{-i} \in \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$ を表す．

2.3 乗算型重み更新法

d 個の意思決定からなる集合 D が存在し，各ラウンドにおいて，意思決定を一つ選択するものとする．具体的には，ラウンド t において，$\sum_{a \in D} \boldsymbol{x}_a^t = 1$ を満たすベクトル $\boldsymbol{x}^t = (\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_d^t)$ を選択する．$\mathcal{P}^t$ を $\boldsymbol{x}^t$ に関わる確率分布とすると，意思決定 $a \in D$ の確率分布は $\mathcal{P}^t(a) = \boldsymbol{x}_a^t$ と書ける．この確率分布 $\mathcal{P}^t$ から意思決定 a をサンプリングする．意思決定をサンプリング後，すべての意思決定に対するコストがベクトル $\boldsymbol{c}^t = (\boldsymbol{c}_1^t, \ldots, \boldsymbol{c}_d^t)$ の形で表れる．そのため，この解法から得られる期待値は $\mathbf{E}_{a \sim \mathcal{P}^t}[\boldsymbol{c}_a^t] = \langle \boldsymbol{x}^t, \boldsymbol{c}^t \rangle$ と表せる．つまり，T ラウンド後の期待値の総和は $\sum_{i=1}^T \langle \boldsymbol{x}^t, \boldsymbol{c}^t \rangle$ となる．

この期待値の総和が，結果的に最適な意思決定のコストと比較してそれほど悪くならない．つまり，$\min_{a \in D} \sum_{t=1}^T \boldsymbol{c}_a^t$ を達成するアルゴリズムが望まれる．アルゴリズム 1 は乗算型重み更新法（MW 法）と呼ばれ，この性質を満たすものとして知られている．具体的には，以下のとおりである．

定理 2 ([Arora 12])**.** すべてのコストが $\boldsymbol{c}_a^t \in [-1, 1]$ であると仮定する．$T = O(\frac{\log |D|}{\epsilon^2})$，$\eta = \sqrt{\frac{\log d}{T}}$ とすると，*MW* 法は以下が成り立つ．T ラウンド後，任意の意思決定 a に対して，

$$\frac{1}{T}\left(\sum_{t=1}^T \langle \boldsymbol{c}^t, \boldsymbol{x}^t \rangle - \sum_{t=1}^T \boldsymbol{c}_a^t\right) \leq \epsilon.$$

を満たす．

左辺は regret と呼ばれる．もし $T \to \infty$ のときの regret が高々0 であるなら，その解法は *no-regret* アルゴリズムと呼ばれる．MW 法は no-regret アルゴリズムの一種である．

3. ブラックボックスリダクションアルゴリズム

本稿では，分散 MaxSAT の相関均衡点を求めるためのアルゴリズムとして，ブラックボックスリダクションアルゴリズムを導入する．このアルゴリズムを分散 MaxSAT に適応するために，分散 MaxSAT を以下に示すコスト最小化ゲームとしてみなす．

- n 人のプレイヤー $i \in \{1, 2, \ldots, n\}$
- 各プレイヤー i は戦略集合 $D_i \in \{0, 1\}$ をもつ
- 各プレイヤー i はコスト関数 $f_i : D \to [0, 1]$ をもつ．ここで，$D = D_1 \times \cdots \times D_n$.

分散 MaxSAT の各エージェントをプレイヤー，値域を戦略集合と対応付けることにより，分散 MaxSAT をコスト最小化ゲームとみなせる．また，コスト関数はコストの最大値で割ることにより $[0, 1]$ の空間にマッピングできる．

次に，コスト最小化ゲームの相関均衡点を求める方法として，ブラックボックスリダクションアルゴリズムを導入する．相関均衡点の定義 1 を思い出すと，すべてのプレイヤー i に対する任意の 2 つの戦略に対して，コストが増加しない確率分布を探せばよい．定義 1 の左辺から右辺を引いた差は swap-regret と呼ばれ，ブラックボックスリダクションアルゴリズムはこの regret を最小にする no-swap-regret アルゴリズムの一種である．no-swap-regret はコスト最小化ゲームに対して以下の性質をもつ．

定理 3. T ラウンド後の *no-swap-regret* アルゴリズムでは，コスト最小化ゲームの各プレイヤーはどの戦略への変化に対しても高々 ϵ の *swap regret* しかもたない．$\mathcal{P}^t = \prod_{i=1}^n \boldsymbol{p}_i^t$ をラウンド t における確率分布，$\mathcal{P} = \frac{1}{T}\sum_{t=1}^T \mathcal{P}^t$ を T ラウンド後の確率分布の時間平均とすると，各プレイヤー i とすべての戦略 a_i に対して，以下の不等式が成り立つ場合，$\mathcal{P}$ は ϵ-近似相関均衡点と言う．

$$\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{P}}[f_i(\boldsymbol{x})] \geq \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{P}}[f_i(\delta_i(x_i), \boldsymbol{x}_{-i})] + \epsilon.$$

定理 4. $T = O(\frac{\log d_{max}}{\epsilon^2})$ において，アルゴリズムより得られる確率分布 $\mathcal{P}$ は ϵ 近似相関均衡点である．

アルゴリズム 2 にブラックボックスリダクションアルゴリズムの全体像を示す．

ブラックボックスリダクションアルゴリズムでは，各プレイヤーはおおまかに次のような流れで動作する．

Step1: プレイヤー i の no-regret アルゴリズム $M_{i,1}, \ldots M_{i,D_i}$ はそれぞれ確率分布 $\boldsymbol{q}_{i,1}, \ldots, \boldsymbol{q}_{i,D_i}$ を計算し，プレイヤー i に送信する．

Step2: プレイヤー i は $\boldsymbol{q}_i$ をもとに定常分布 $\boldsymbol{p}_i$ を計算し，隣接する他のプレイヤー $j \in NB(i)$ に $\boldsymbol{p}_i$ を送信する．

Step3: 隣接プレイヤーから受け取った確率分布 $\{\boldsymbol{p}_j | j \in NB(i)\}$ をもとに各戦略 $a \in D_i$ の期待コストを計算し，各 no-regret アルゴリズムに送信する．

Step4: no-regret アルゴリズム $M_{i,a}$ は受け取ったコストをもとに重み $\boldsymbol{w}_{i,a}$ を更新する．

以下の節では，各ステップについて詳細に説明する．

3.1 確率分布 $\boldsymbol{q}$ の計算

ブラックボックスリダクションアルゴリズムでは，各プレイヤー i は各戦略 $j \in \{1, \ldots, |D_i|\}$ に対して，$|D_i|$ 個の no-regret アルゴリズム $M_{i,1}, \ldots, M_{i,|D_i|}$ をもつ．また，各アルゴリズム $M_{i,a}$ に対して，重み $\boldsymbol{w}_{i,a} \in \mathbb{R}^{|D_i|}$ をもつ．各 no-regret アルゴリズム $M_{i,a}$ は，ラウンド $t \in \{1, \ldots, T\}$ において，与えられた重み $\boldsymbol{w}_{i,a}$ に基づき確率分布 $\boldsymbol{q}_{i,a} := \{q_{i,a,1}^t, \ldots, q_{i,a,|D_i|}^t\}$ を以下のように求める．

$$q_{i,a,b}^t = \frac{w_{i,a,b}}{\sum_{b \in D_i} w_{i,a,b}}, \forall b \in D_i,$$

Algorithm 2 ブラックボックスリダクションアルゴリズム

procedure Blackbox Reduction Algorithm
for each agent i **do**
  Fix $\eta \leq 1/2$ and set $\boldsymbol{w}_{i,a}^1 \leftarrow (1, 1, \ldots, 1)$ for every action a and $\boldsymbol{p}_i = \boldsymbol{c}_i = \{0, 0, \ldots, 0\}$.
for $t = 1, 2, ..., T$ **do**
  for each agent i **do**
    Receive distributions $\boldsymbol{q}_{i,1}^t, \ldots, \boldsymbol{q}_{i,|D_i|}^t$ from the algorithms $M_{i,1}(\boldsymbol{p}_{i,1}^t \boldsymbol{c}_i^t), \ldots, M_{i,|D_i|}(\boldsymbol{p}_{i,|D_i|}^t \boldsymbol{c}_i^t)$.
    Compute a stationary distribution $\boldsymbol{p}_i^t$ and sends it to their neighbors $j \in NB(i)$.
    Observe the costs of their decisions $\boldsymbol{c}_1^t, \ldots, \boldsymbol{c}_d^t$.
end procedure
procedure $M_{i,a}(\boldsymbol{p}_i \cdot \boldsymbol{c}_i)$
for for each action $b \in D_i$ **do**
  update weights $w_{i,a,b} \leftarrow w_{i,a,b}(1 - \eta q_{i,a,b} p_{i,a} c_{i,a})$
for for each action $b \in D_i$ **do**
  $q_{i,a,b} = \frac{w_{i,a,b}}{\sum_{b \in D_i} w_{i,a,b}}$
return $\boldsymbol{q}_{i,a}$
end procedure
procedure STAT($\boldsymbol{q}_i$)
  Compute $\boldsymbol{p}_i$ such that $\boldsymbol{p}_i = \boldsymbol{p}_i \boldsymbol{q}_i$ and $\sum_{a \in D_i} p_{i,a} = 1$.
return $\boldsymbol{p}_i$
end procedure

ある確率分布 $q_{i,a,b}$ は，戦略 a から $b \in D_i \backslash \{a\}$ へ変更する確率を表す．直観的には，重み $\boldsymbol{w}$ は戦略を変更したときのコストが大きいほど小さくなる値である．つまり，no-regret アルゴリズム $M_{i,a}$ は，なるべくコストの増加が少ない戦略の確率を高くしようとする．

3.2 定常分布 $\boldsymbol{p}$ の計算

プレイヤー i は各アルゴリズム $M_{i,1}, \ldots, M_{i,|D_i|}$ から確率分布 $\{\boldsymbol{q}_{i,1}, \ldots \boldsymbol{q}_{i,|D_i|}\}$ を受け取った後、定常分布 $\boldsymbol{p}_i$ を計算する．プレイヤー i の定常分布 $\boldsymbol{p}_i$ は以下の問題より得られる．

$$\begin{array}{ll} \text{find} & \boldsymbol{p}_i := \{p_{i,1}, \ldots, p_{i,D_i}\} \\ \text{subject to} & p_{i,a} = \sum_{i=1}^n p_{i,a} \cdot q_{i,a}, \forall a \in D_i \\ & \sum_{a \in D_i} p_{i,a} = 1. \end{array}$$

ここで，$\boldsymbol{q}_i \in [0, 1]^{|D_i| \times |D_i|}$，$\boldsymbol{q}_i$ の (a, b) 成分を $\boldsymbol{q}_{i,a,b}$ とする．この問題は固有値計算により簡単に計算できる．直感的に，定常分布 $\boldsymbol{p}_i$ は，プレイヤー i が戦略 $a \in D_i$ を選択する状態である確率を表す．プレイヤーは確率分布 $\boldsymbol{p}_i$ を計算後，他のプレイヤーに $\boldsymbol{p}_i$ を送信する．

3.3 重みの更新

コスト $\boldsymbol{c}_i = \{c_{i,1}, \ldots, c_{i,D_i}\}$ とすると，各プレイヤー i は隣接するプレイヤーに $\boldsymbol{p}_i$ を送受信後に得られる戦略 $a \in D_i$ に対するコスト $c_{i,a}$ を以下のように計算する．

$$c_{i,a}^t = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \boldsymbol{p}}[f_i(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n)].$$

プレイヤー i のマスターアルゴリズムは各戦略 $a \in D_i$ の no-regret アルゴリズム $M_{i,a}$ にコスト $p_{i,a}^t \cdot \boldsymbol{c}_i^t$ を渡す．no-regret アルゴリズム $M_{i,a}$ は受け取ったコストを基に重み $\boldsymbol{w}_{i,a}$ を次のように更新する．

$$w_{i,a,b}^{t+1} \leftarrow w_{i,a,b}^t (1 - \eta q_{i,a,b}^t p_{i,a}^t c_{i,b}^t).$$

例 2. 例 *1* を用いてブラックボックスリダクションアルゴリズムの例を示す．図 *1* はラウンド *1* における各プレイヤーの振舞を示した図である．$\eta = 1/2$ に設定したとする．プレイヤー 1 に注目すると，プレイヤー 1 のアルゴリズム $M_{1,0}, M_{1,1}$ はそれぞれ重み $biw^1_{1,0}, \boldsymbol{w}^1_{1,1}$ を $\{1,1\}$ に初期化し，確率分布を求め，プレイヤー *1* に送信する．つまり，プレイヤー *1* は $\boldsymbol{q}^1_{1,0} = \boldsymbol{q}^1_{1,1} = \{1/2, 1/2\}$ を受け取る．次に，プレイヤー 1 は受け取った確率分布 $\boldsymbol{q}^1 = \{\boldsymbol{q}^1_{1,0}, \boldsymbol{q}^1_{1,1}\}$ を基に定常分布 $\boldsymbol{p}^1_1 = \{\boldsymbol{p}^1_{1,0}, \boldsymbol{p}^1_{1,1}\}$ を計算する．つまり，$\boldsymbol{p}^1_1$ の値は $\boldsymbol{p}^1_{1,0} = \boldsymbol{p}^1_{1,1} = 0.5$ となる．各プレイヤーは節を共有する他のプレイヤーに計算した確率分布 $\boldsymbol{p}^1$ を送信する．この例では，プレイヤー *1* はプレイヤー *2* と *3* に確率分布 $\boldsymbol{p}^1_1$ を送信し，プレイヤー *2* と *3* からそれぞれ確率分布 $\boldsymbol{p}^1_2$ と $\boldsymbol{p}^1_3$ を受け取る．このとき，$\boldsymbol{p}^1_1 = \boldsymbol{p}^1_2 = \boldsymbol{p}^1_3$ である．プレイヤー *1* は受け取った確率分布を基に各戦略に対するコスト $\boldsymbol{c}^1_1 = \{c^1_{1,0}, c^1_{1,1}\}$ を計算する．例えば，戦略 *0* を選択したときのコスト $c^1_{1,0}$ は節 C_1 と C_2, C_4 から計算できる．C_1 は節を違反しないため発生するコストは *0* である．一方，C_2, C_4 に関しては他のプレイヤーが *2* つの戦略をそれぞれ確率 *1/2* で選択するため，コストの期待値はともに *1/2* となる．つまり，$c^1_{1,0} = 0 + 1/2 + 1/2 = 1$ となる．同様に，$c^1_{1,1} = 2$ となる．プレイヤー *1* は，各アルゴリズム $M_{1,0}, M_{1,1}$ にそれぞれコスト $p^1_{1,0}c^1_{1,0}$ と $p^1_{1,1}c^1_{1,1}$ を送信する．アルゴリズム $M_{1,0}$ は受け取ったコスト $p^1_{1,0}c^1_{1,0}$ をもとに重み $\boldsymbol{w}^1_{1,0}$ を次のように更新する．$\boldsymbol{w}^2_{1,0,0} = \boldsymbol{w}^1_{1,0,0}(1 - \eta \cdot q^1_{1,0,0} \cdot p^1_{1,0} \cdot c^1_{1,0})$．つまり，ラウンド *2* における重みは $\boldsymbol{w}^2_{1,0} = \{0.875, 0.75\}$ となる．各アルゴリズムは新たに得られた重みを基に次のラウンドの確率分布 $\boldsymbol{q}^2$ をする．

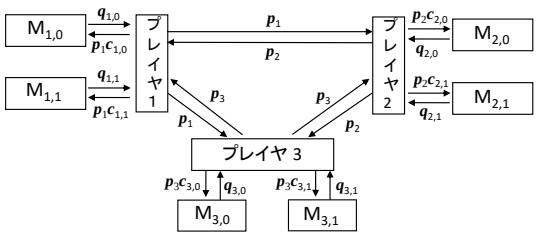

図 1: ブラックボックスリダクションアルゴリズムのラウンド 1 における各プレイヤーの振る舞い

4. 関連研究

得られた相関均衡点を評価する方法として，Price of Anarchy (PoA) がある [Koutsoupias 99]．PoA は，すべての均衡解を考慮したときに，最適値を均衡解の値で割った割合が最も大きくなる値を指す．つまり，均衡解を達成できれば，最悪の場合でも PoA の範囲に収まることを意味する．一般的に，PoA は純粋ナッシュ均衡に対してのみ解析されてきたが，Roughtgarden により提案された smooth game を用いることにより，ある種のゲームに対して純粋ナッシュ均衡解以外の一般的な均衡解に対しても PoA を容易に解析可能になった [Roughgarden 15]．

本稿では，以下に定義する smooth game を用いて，ブラックボックスリダクションアルゴリズムより得られる相関均衡点の解析を行う．

定義 5 (Smooth Game [Roughgarden 15])**.** あるコスト最小化ゲームが (λ, μ)-*smooth* であるとは，任意の *2* つの割当 $\boldsymbol{x}$ と $\boldsymbol{x}*$ に対して，

$$\sum_{i=1}^{n} f_i(x^*_i, \boldsymbol{x}_{-i}) \leq \lambda \cdot f(\boldsymbol{x}^*) + \mu \cdot f(\boldsymbol{x}),$$

が成り立つことを言う．ここで，$f(\boldsymbol{x}) = \sum f_i(\boldsymbol{x}), \lambda > 0, \mu < 1$ である．

もし，あるゲームが (λ, μ)-smooth であれば，どの純粋ナッシュ均衡 $\boldsymbol{x}$ の値も最適解 $\boldsymbol{x}^*$ の値に対して高々$\lambda/(1-\mu)$ 倍の値にしかならないことを簡単に示すことができる．定理 5 の強みは任意の 2 つの割当に対して成り立つ点である．つまり，smooth game であることを用いることで，より一般的な均衡解である相関均衡点に関しても同様に PoA を解析できる．

5. おわりに

本稿では，分散 MaxSAT に対する相関均衡点を求める方法としてブラックボックスリダクションアルゴリズムに基づくアルゴリズムを提案した．今後の課題として，(λ, μ)-smooth を利用して求めた相関均衡点を Price of Anarchy により評価することが挙げられる．そのためには，MaxSAT を変換したコスト最小化ゲームの λ と μ を求める必要がある．また，実験による評価も行う必要がある．

参考文献

[Arora 12] Arora, S., Hazan, E., and Kale, S.: The Multiplicative Weights Update Method: A Meta-Algorithm and Applications, *Theory of Computing*, Vol. 8, No. 1, pp. 121–164 (2012)

[Aumann 74] Aumann, R. J.: Subjectivity and correlation in randomized strategies, *Journal of Mathematical Economics*, Vol. 1(1), pp. 67–96 (1974)

[Dodis 00] Dodis, Y., Halevi, S., and Rabin, T.: A Cryptographic Solution to a Game Theoretic Problem, in *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pp. 112–130 (2000)

[Koutsoupias 99] Koutsoupias, E. and Papadimitriou, C.: Worst-case equilibria, in *In Proceeding of The 16TH Annual Symposium on Theoretical Aspects of Computer Science*, pp. 404–413 (1999)

[Nisan 07] Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V.: *Algorithmic Game Theory*, Cambridge University Press (2007)

[Roughgarden 15] Roughgarden, T.: Intrinsic Robustness of the Price of Anarchy, *Journal of the ACM*, Vol. 62, No. 5, pp. 32:1–32:42 (2015)

[Yokoo 98] Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K.: The Distributed Constraint Satisfaction Problem: Formalization and Algorithms, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, pp. 673–685 (1998)