多人数ゲームにおける報酬分配学習

○柴田 克成, 真崎 勉 （大分大学）

Learning of Reward Distribution in a Multi-Player Game

○Katsunari Shibata and Tsutomu Masaki (Oita University) shibata@cc.oita-u.ac.jp

Abstract: In this paper, autonomous learning of reward distribution in multi-agent reinforcement learning was applied to a 4-player game named "not100". In this game, more shrewd strategy to cooperate with the other agents is required for each agent than the other tasks that the learning was applied previously. The reward distribution ratio after learning was varied among simulation runs. However, the validity of the average reward distribution ratio was examined in some ways. The three agents with higher probability of win after learning were stronger when the distribution ratio was learned than some cases when the agents learned with fixed distribution ratio.

１．まえがき

マルチエージェントシステムでは、あらかじめタスク解決の方策を知ることが難しいため、強化学習のような自律的な学習が有効であると考えられる。しかし、利害の衝突回避や協調行動に大きく影響をおよぼす、「複数のエージェント間での報酬配分」は難しい問題である。従来、タスク達成に最終的に貢献したエージェントのみに報酬が与えられたり、エージェント全員に均等に分配したり[1]、その中間をとる[2]などして配分割合が事前に決定されていた。しかし、適切な配分は、タスクに依存するところが大きいため[2]、それを決定するためには、タスクに関する十分な知識が必要である。したがって、タスクに関する知識が十分でない時に威力を発揮する強化学習の有効性を損なう可能性が大きい。

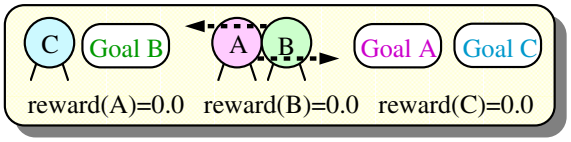
そこで、筆者らの一部は、報酬を得たエージェントが、その報酬を他のエージェントにどれぐらい分配するかを、「報酬を分配することで、その後他のエージェントが協力してくれて自らのためになる」という学習指針のもとで、行動の学習と並行して行っていく方法を提案し[3]、簡単な３エージェントの競合問題に適用した[4][5]。そして、(1)コンフリクトが続いて誰も報酬が得られない状態は回避でき、外部から観測して妥当と考えられる解が得られること、(2)他の協力を必要としない場合は、報酬配分割合が小さくなること、(3)３エージェントの場合は、もらう報酬がもう一つのエージェントより小さいと、ゴールをじゃまするなどの駆け引きも観察された。また、(4)報酬配分の割合だけでなく、その変化量も行動の学習に影響することもわかった。

本稿では、(1)より複雑な駆け引きが必要、(2)コンフリクトによって全エージェントが報酬を得られなくなることがない、(3)多くの強化学習問題のように、目的を達成する時間が早いほど良いということがない、(4) 複数のエージェントが報酬をもらえるという点で、従来適用してきた問題と異なるタイプの問題に本学習を適用し、本手法の広い有効性を検証する。具体的には、not100 (実際には not30)という４人で行うゲームを取り上げる。このゲームは、３人が協力すれば、他の一人は必ず負けるというゲームであり、巧妙な駆け引きが要求される。そして、本学習を通して協調関係が生まれるか、また、学習によって得られた報酬分配割合が妥当かといった点に関する検証結果を報告する。

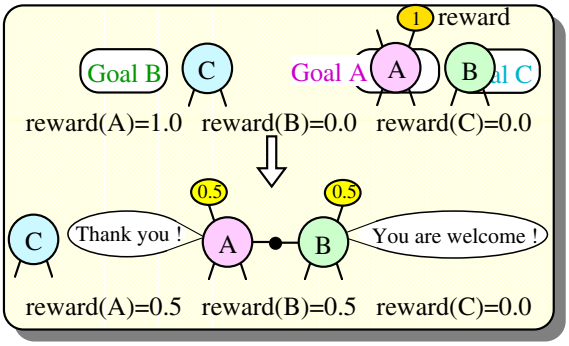
２．報酬分配の学習

始めに、本稿で用いる報酬分配の学習について説明する。図１(a)のように、A, B, C の３つのエージェントがそれぞれのゴールを目指し、A と B はコンフリクト状態にあるとする。ここで、図１(b)のように B がたまたま道を譲り、A がゴールし、報酬１を獲得したとしても、B は報酬をもらえないため、A が道を譲るのを待った方が得となり、道を譲るという行動は学習されない。したがって、学習が進むにつれ、お互いに道を譲らないことを学習し、ますますデッドロック状態から抜け出せなくなる。ところが、A が、もらった報酬の半分を B にあげたとすれば、デッドロック状態で報酬が得られないことと比較すれば、A, B ともに得になる。つまり、A にとっては、B に報酬を分配すると短期的には損になるが、長期的には得になる。しかし、A が C に報酬を分配しても、見返りはなく、自分の報酬が減少するだけであるので、C に報酬を分配することは学習しない。これが基本的な学習原理であり、あくまでも自らの利益のために学習を行う。

また、３エージェント以上の場合は、Fig. 2 のように、分け与えてもらえる報酬の大小によって協力するエージェントを選択するという働きも持つ。いずれにしても、報酬を相手に与え過ぎると自分が獲得する報酬が減少するし、与えないと協力してもらえず、結果として自分がゴールして報酬を得る確率が減少する。したがって、その中間のちょうど良いところに報酬分配率が落ち着くと期待される。



(a) confilict state



(b) when the reward is distributed

Fig. 1 Principle of reward distribution learning.

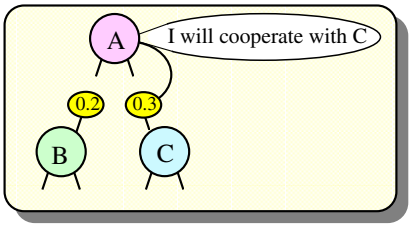


Fig. 2 Selection of the agent to cooperate with depending on the reward distribution ratio.

前述のような学習を、行動の学習と並行して行うが、２つの学習はその時間スパンが異なる。そこで、行動の学習は常に行うが、報酬分配率は、変化させた後の何試行か固定し、その間の自ら得られる報酬をもとに、より報酬が得られる方に学習を行う。具体的には、エージェント i から j への報酬分配率 $dist_{ji}$ を乱数を用いて変化させ、その後の N 試行の間固定させる。そして、過渡状態を除くため、その後半 $N/2$ 試行の間に得られる強化信号 r とステップ数 $step$ の総和を

$$total\_rein_j = \sum_{n=N/2+1}^{N} \sum_{i=1}^{A} dist_{ji} r_i(n) \quad (1)$$

$$total\_step = \sum_{n=N/2+1}^{N} step(n) \quad (2)$$

ただし、A: エージェントの数

と計算する。本稿では、N=400 とした。筆者の一部が以前提案した方法では、エージェント j に対し

$$R_j = total\_rein_j \, \gamma^{total\_step} \quad (3)$$

という長期的な評価を導入し、各エージェントごとに、分配率を変化させる前後でこの値を比較した。しかし、ここでは、試行が無期限に続くことはなく、時間を評価に入れない方が良いタスク上の性格より、

$$R_i = total\_rein_i \quad (4)$$

として評価した。そして、この値が増加した場合は加えた値を新たな分配率とし、そうでない場合は加える前の値に戻す。この時、分配率は、常に

$$\sum_{j=1}^{A} dist_{ji} = 1.0 \quad (5)$$

を満たす必要があるので、分配値の変化量は、

$$\Delta dist_{ji} = rnd_{ji} - rnd_{(j+1)\%A,\, i} \quad (6)$$

とした。分配値 $dist$ が 1 より大きくなった場合には 1 とし、0 より小さくなった場合は 0 として、その分を他のエージェントへの分配値に均等に分配した。

ここでは、各エージェントは、エージェント 0,1,2.. の順に１ステップ１エージェントが行動するものとし、状態遷移は決定論的とした。学習は、基本的には Q-learning に基づく。前述のように、今回は時間を評価に入れる必要がないため、割引率 γ は１とした。例としてエージェント j=0 の学習則を示す。まず、時刻 t でエージェント j の行動後の状態評価値 $Vj(sj(t+1))$を、次の自分の順番までにゲームが終了する確率 P_{fin_j}、その場合の期待報酬 $\tilde{r}_{fin_j}$、および、他のエージェントが行動後の最大 Q 値の期待値 $maxQ_j$ を用いて、

$$V_j(s_j(t+1)) = P_{fin_j}(s_j(t+1))\, \tilde{r}_{fin_j}(s_j(t+1)) + (1 - P_{fin_j}(s_j(t+1)))\, maxQ_j(s_j(t+1)) \quad (7)$$

と計算する。ここで、αを学習係数として

$$P_{fin_j}(s_j(t+1)) \leftarrow (1-\alpha)\, P_{fin_j}(s_j(t+1)) + \alpha \quad \text{if the game finishes by its next tern}$$
$$\leftarrow (1-\alpha)\, P_{fin_j}(s_j(t+1)) \quad \text{otherwise} \quad (8)$$

$$\tilde{r}_{fin_j}(s_j(t+1)) \leftarrow (1-\alpha)\, \tilde{r}_{fin_j}(s_j(t+1)) + \alpha \sum_{i=0}^{A} dist_{ji}\, r_i(t+k) \quad \text{if the game finishes at } t+k \quad (9)$$

$$maxQ_j(s_j(t+1)) \leftarrow (1-\alpha)\, maxQ_j(s_j(t+1)) + \alpha \max_k(Q_j(s_j(t+A), a_k) \quad (10)$$

とし、この V を用いて以下のように Q 値を学習する。

$$Q_j(s_j(t),a(t)) \leftarrow (1-\alpha)\, Q_j(s_j(t),a(t)) + \alpha\{\sum_{i=0}^{A} dist_{ji}\, r_i(t+1) + V_j(s_j(t+1))\} \quad (11)$$

３. NOT100 ゲーム

本章では、本稿で用いた NOT100 というゲームについて説明する。通常、４人がテーブルを囲み、１から順番に数をカウントしていき、100 をカウントした人が負けとなり、罰ゲームを行う。ゲームのルールを Fig. 4 に示す。

このゲームでは、一人でいくら頑張っても、他の３人が協力すると勝つことができないため、勝つためには、いかに他の人の協力を得るかが問題となる。したがって、実際のゲームの際にも、人間関係や性格を垣間見るという意味でのおもしろさもある。Fig. 5 に例を示す。たとえば、自分が D で、他の A, B, C の３人が協力していたとする。D が 97, 98, 99 のいずれかをカウントすれば、必ず他の誰かが 100 をカウントすることになり、勝つことができる。しかし、90 から 96 までのいずれかの数をカウントすると、A, B, C の３人で、最大９個、最小３個のカウントができるため、D が必ず 100 をカウントすることになってしまう。Fig. 5 の上段は、D が 90 をカウントした場合である。ところが、下の段のように、D が 89 をカウントしても、A, B, C が１ずつカウントすると、D が３個カウントしても 95 にまでしか届かず、結局、D が 97, 98, 99 のいずれかをカウントすることはできず、どう頑張っても 100 をカウントすることになって負けてしまう。また、例えば、Fig. 3 の場合、C にとっては、最後に、98,99 とカウントしても、98 だけカウントしても自分が負けないという意味では同じであるが、その選択結果によって、D が負けるか、A が負けるかが決まる。

本稿では、計算時間の関係で、NOT100 ゲームを NOT30 ゲームとした。そして、30 がカウントされた際に、カウントしたエージェント以外のエージェントにに 1.0 の報酬を与えた。罰を与える方が実際のゲームの設定に近いが、報酬の分配ということで、負けなかったエージェントに報酬を与える方法を採用した。

今まで提案学習法を適用してきた問題は、自己の利益だけを追求するとコンフリクト状態に陥り、すべてのエージェントが報酬を得られなくなる可能性があり、少なくとも、全エージェントに報酬を分配した方が得であることが明らかであった。しかし、今回は、常に３つのエージェントが報酬をもらうことができ、どう配分すれば良いかが明らかでないという点で、結果が注目される。

４．シミュレーション

１回のシミュレーションは、400 回の試行を１サイクルとし、１サイクルごとに報酬配分割合を変化させ、10000 サイクル行った。したがって、全部で 4000000 という多大な試行を行った。報酬分配率の初期値は、総和が 1.0、負にならない範囲でランダムに決定した。報酬分配率の１回の変化量の範囲を、最初の 9000 サイクルの間に、0.1 から 0.01 に、対数をとったものに対して直線的に減少させた。また、毎試行ごとに、最初にカウントするエージェントを４つのエージェントからランダムに決めた。行動の選択は、当初、単純なボルツマン選択を用いていたが、他のエージェントからの小さな報酬にも反応するように、その状態におけるＱ値の値の最大値が１になるように正規化した後にボルツマン選択で行動を選択した。そして、温度は、最初の 8000 サイクルの間に、1.0 から 0.02 または 0.01 まで、対数をとったものに対して直線的に減少させた。Fig. 6（A）に、サイクルが進むにしたがって、各エー

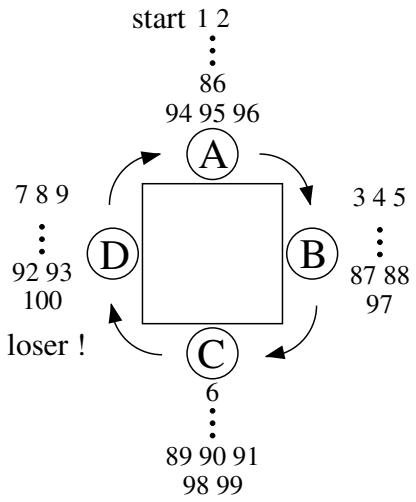


Fig. 3　Not 100 game.

１．最初の数を０とし、４人が順番にカウントしていく
２．１回のカウントは、１，２，３から選択し、現在の数に足していく
（現在の数が 90 で２を選択した場合は 92 となる）
３．足した数が 100 になった人が負け
ただし、足した数が 100 以上になってはいけない（例えば、現在の数が 98 の場合に選択できるカウント数は１か２、現在の数が 99 の場合は、１しか選択できず、負けるしかない）

Fig. 4　The rule of Not 100 game.

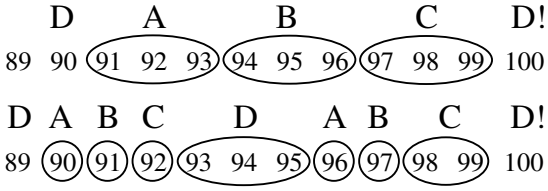


Fig.5　The reason why an agent always loses the game when the other 3 agents cooperate with the others.

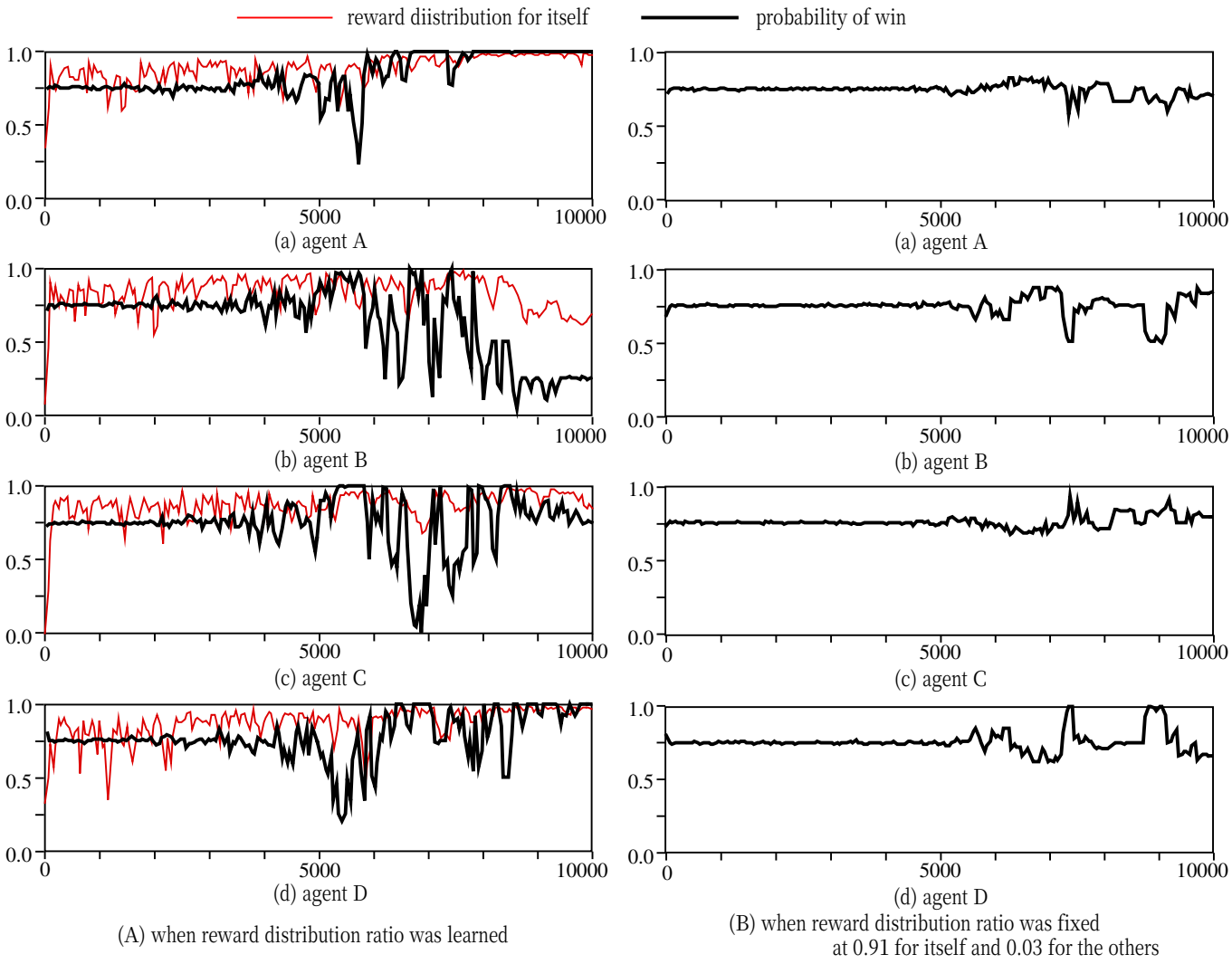


Fig. 6　Change of each agent's win-probability and reward-distribution-ratio to itself.

ジェントの勝利確率と自分自身に残す報酬配分割合がどのように変化したかの例を示す。図は、５０サイクル分を平均したものをプロットしているため、実際には、もっと細かい変動も見られる。

当初は、ボルツマン選択の温度が高いため、各エージェントがほぼランダムに行動しており、勝利確率もほぼ 0.75 のあたりを推移している。一方、報酬分配率は、初期値を乱数で決めているにもかかわらず、いずれのエージェントも、多少の変動はあるものの、すぐに 0.8 から 0.9 程度になっていることがわかる。そして、5000 サイクルに近づく頃から、勝利確率が大きく変動し始め、最終的に落ち着きつつある。ただし、勝利確率の変化の様子は、シミュレーションごとにかなり異なっており、7000 サイクル程度で常に特定の３エージェントが勝ち続けるという場合や、最後まで変動が続く場合もあった。また、報酬分配率は、勝利確率が高いエージェントは大きくなり、低いエージェントは小さくなる傾向が見られた。

比較のため、Fig. 6(B) に、報酬分配率を自分自身に 0.91、他のエージェントに 0.03 で固定した場合の勝利確率の変化を示す。シミュレーションによっては、あるエージェントの最終的な勝利確率が 1.0 になるなど、もう少し差が出ることもあったが、いずれにしても、Fig. 6(A)と比べて、勝利確率の変動が小さかった。これは、報酬分配率を学習している場合、その変動によっても、勝利確率が大きく変動していると考えられる。また、定量的には分析していないが、Fig. 6(A)のエージェント C の 7000 サイクルの手前とその後のように、分配率を学習した場合には、自分への分配率が大きくなると勝利確率が減少し、勝利確率が小さくなると分配率が小さくなる傾向が見られたが、定量的には評価していない。

次に、学習後の報酬分配率の妥当性について検証してみた。Table 1 に学習後の報酬分配率を、ボルツマン選択の最終温度を 0.01 と 0.02 の場合について、さらに、学習後のすべてのエージェントについて平均した

場合と勝利確率が 0.9 以上のエージェントについて平均した場合について、自分自身、次のエージェント、向かいのエージェント、前のエージェントへの報酬分配率を示した。いずれも、乱数系列を変えて、100 回のシミュレーションの結果を平均したものである。したがって、全エージェントの場合は、のべ 400 エージェントの平均、勝利確率が 0.9 以上のものは、最終温度が 0.01 の場合が 248、0.02 の場合が 180 エージェントの平均となっている。この結果から、勝利確率の高いエージェントの方が自分自身への報酬分配率が大きく、最終温度が小さい方がその傾向が強いこと、次のエージェントへの報酬分配率が、わずかではあるが、他のエージェントへの分配率より小さいことがわかる。直感的にも、次のエージェントよりも、向かいまたは直前のエージェントに協力してもらった方がゴール確率が上がると考えられる。ただし、この値の変動は大きいため、平均的には分配率に差が出る傾向があるが、そうでない場合も多々あった。

さらに、１エージェントの報酬分配率だけ固定して他を学習させた。そして、固定した値を変化させた時の、そのエージェントの勝利確率、獲得報酬、および他のエージェントの自分自身への報酬分配率の平均値を最終温度が 0.01 と 0.02 の場合について Fig. 7 に示す。この図は、固定した報酬分配率ごとに、それぞれ 10 回行ったシミュレーション結果の平均である。この図からわかるように、自分自身への報酬分配率が 0.8 や 0.9 周辺より大きい時、および 0.0 のすぐ近くで急激に勝利確率が小さくなるが、それ以外の場合は勝利確率がほぼ 1.0 であることがわかる。また、分配率が大きくなるにしたがって、獲得報酬も大きくなるが、報酬分配率が 0.8 や 0.9 あたりより大きくなって勝利確率が減ると大きく減少する。また、報酬分配率が 0.8 や 0.9 あたりより大きくなると、学習している他のエージェントの自分への報酬分配率が少し大きくなっている。これは、相手の出方を見て自分の報酬分配率を変化させていることを表しており、非常に柔軟であることがわかる。また、報酬分配率が 0.8 や 0.9 というのは、Table 1 で見た学習後の自分自身への報酬分配率に近い。また、温度が小さいほど、その値が大きくなることも、Fig. 7 と一致しており、本学習の妥当性を表している。

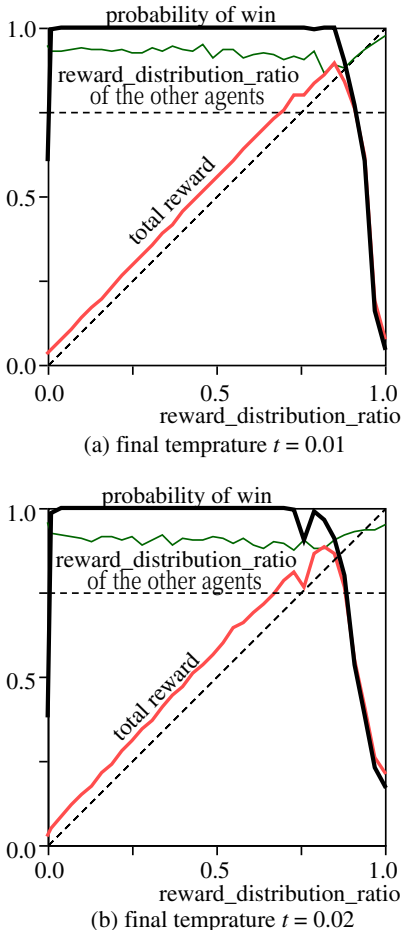


Fig. 7 The probability of win, obtained reward and the reward distribution ratio of the other agents to themselves when the reward distribution ration of one agent is fixed. x-axis indicates the fixed reward distribution ratio to itself. The ratio to the other agents is uniform.

Table 1 The reward distribution ratio after learning

		the agent whom the reward is distributed to				
			itself	next agent	opposite agent	previous agent
final temprature	t=0.01	the agents with *win_prob*>0.9	0.961	0.011	0.013	0.014
		all agents	0.895	0.037	0.033	0.035
	t=0.02	the agents with *win_prob*>0.9	0.922	0.022	0.030	0.026
		all agents	0.905	0.029	0.035	0.031

さらに、前述のように、Table 1 で他の３つのエージェントに対する分配率が、次のエージェントには少ないなど、わずかではあるが均一でないことが意味のあることかどうかを検証した。すべてのエージェントの報酬分配率を固定し、一つのエージェントのみ、Table 1 の分配率と同じにし、その他のエージェントは、自分自身への分配率は Table 1 にそろえたが、他のエージェントには均等に配分した。学習後の各エージェントの勝利確率を Fig. 8 (a) に示す。また、逆に、３つのエージェントを Table 1 とそろえ、残りのエージェントだけ、自分以外に均等配分とした場合を Fig. 8(b) に示す。いずれの場合も、最終温度 0.01、勝利確率が 0.9 以上のエージェントの平均の値を用いた。この図

よりわかるように、自分自身への分配率が同じでも、他のエージェントへの配分を Table 1 のようにした方が、勝利確率があがることがわかる。

最後に、報酬分配率を学習させた場合と、固定して学習させた場合について、その中で一番勝利確率が低かったエージェントを、別の場合で勝利確率が一番高かったエージェントと入れ替えて、その勝利確率を観察した。その結果を Table 2 に示す。固定した分配率の値は、すべて自分に分配(利己的)、全エージェント均等(0.25 ずつ)、さらに、自分に 0.91 他に 0.03 の３通りの場合について行った。それぞれ１０回のシミュレーションを行ったので、入れ替えの組み合わせは 100 通りあり、その平均を表に示した。

まず、入れ替える前の最小勝利確率を見ると、学習した場合は 0.238 と小さく、それ以外は、0.5 以上と大きい。分配率を固定した場合は、いずれも自分以外の報酬分配率は等しいが、分配率を学習した場合は、エージェントごとに自分にくれる報酬が異なるため、Fig. 2 で示したように、特定の３エージェント間で協調関係が形成されやすいためと考えられる。

報酬分配率を学習したエージェントの中に、分配率を固定して行動だけ学習したエージェントを入れた場合は、いずれも、勝利確率が 0.2 以下となった。これは、特定の３エージェント間での協調関係ができあがっているため、外部から入ってもたちうちできないためと考えられる。一方、逆に学習したエージェントの一つを他のエージェントの中に入れた場合も、勝利確率が下がっている。しかし、いずれも 0.5 より大きい。これは、分配率を学習したエージェントが協調関係を得られないため、個人で行動を学習してきた中ではあまり強くないと考えられる。これは、利己的と 0.91-0.03 の分配率で固定した場合間で入れ替えた方が、分配率が大きいことから推測される。また、筆者自身が入っても、分配率を学習した場合、ほとんど勝てなかった。

５．結論と今後の課題

報酬分配学習を not30 ゲームに適用した。そして、エージェント間に協調関係が得られること、および、学習によって得られた報酬分配率が妥当であり、学習が柔軟に行われていることを確認した。

謝辞

本研究の一部は、文部省科研費(#13780295)の補助の下で行われた。ここに謝意を表する。

参考文献

[1] Ono, N. and Fukumoto, K.: Multi-agent Reinforcement Learning: A Modular Approach, Proc. of ICMAS-96, pp.252-258 (1996)

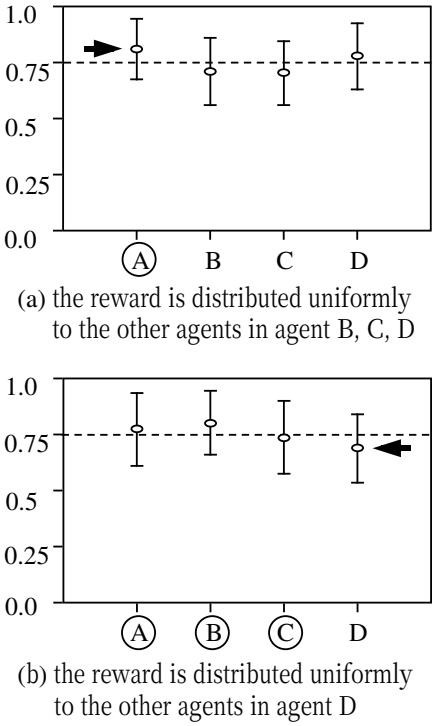


Fig. 8 Effect of non-uniform reward distribution. The alphabet with circle indicates the agent with non-uniform reward distribution.

Table 2 Change of win-probability by the replace of one agent between two groups.

		the agent with the minimum win-probability was replaced				
		max	learning	selfish 1.0-0.0	uniform 0.25-0.25	0.91 -0.03
	min		0.238	0.578	0.711	0.592
the agent with the maximum win-probability is chosen	learning	1.000		0.504	0.898	0.511
	selfish	0.921	0.158		0.932	0.568
	uniform	0.784	0.096	0.302		0.345
	0.91 -0.03	0.889	0.175	0.554	0.920	

[2] 白川英隆, 木村元, 小林重信: 強化学習による協調的行動の創発に関する実験的考察, 第 25 回知能システムシンポジウム予稿集, pp. 119-124 (1998)

[3] 柴田克成、伊藤宏司: ２エージェント強化学習における報酬分配の自律学習, ロボティクス・メカトロニクス講演会(ROBOMEC) '99, 1A1-28-036, (2pages) (1999)

[4] Katsunari Shibata and Koji Ito: Autonomous Learning of Reward Distribution for Each Agent in Multi-Agent Reinforcement Learning, Intelligent Autonomous Systems, Vol. 6, pp. 495-502 (2000)

[5] 柴田克成, 杉坂政典, 伊藤宏司: マルチエージェント強化学習における報酬配分の自律的学習, 第 19 回計測自動制御学会九州支部大会学術講演会予稿集, pp.483-486 (2000)