多目的意思決定のα-domination戦略に基づく分散強化学習と分散データベースへの応用

○青木 圭，木村 元，小林 重信
東京工業大学　大学院総合理工学研究科

Distributed Reinforcement Learning based on α-domination Strategy for Multi-criteria Decision Making and its Application to Distributed Database Systems

Kei Aoki, Hajime Kimura, Shigenobu Kobayashi
Interdisciplinary Graduate School of Science and Eng., Tokyo Institute of Technology

***Abstract*— In the distributed systems in which information cannot be exchanged directly among agents, we deal with problems of deciding how each agent holds the shared resource. To achieve a lot of tasks greedily, agents tend to attempt to hold the resources for a long term. However the system performance decreases consequentially because it competes with the processing of other agents' tasks. To acquire cooperative policies that avoid above competition, we formulate the resource sharing problems to multi-criteria decision making problems with the priority level by using the domain knowledge into the reward. We propose distributed reinforcement learning that narrows the choice of action space by using the α-domination strategy based on value functions for the object and the cooperation. The proposed method is applied to the distributed database systems, and simulation results shows that our method acquires cooperative policies and improves the throughput performance of the system.**

1　はじめに

近年では様々な分散システムの自動化や自律エージェント群による協調システムなどの研究が盛んに行われている．特にネットワーク環境では不特定多数のタスクが存在し，その効率的な自動化や自律化が望まれている．しかし，いくつかのマルチエージェントシステム (MAS) ではシステム全体の効率的な運用が重要であるにもかかわらず，一般にエージェント間の情報交換を行うことは簡便性やコストの点から困難であるため，各エージェントが独自に性能を追求すると競合が発生し，結果的にシステムの性能が低下するという問題点がある．我々はMASが扱うタスクを分類し，**競合する協調タスク**の特徴を明示した後，典型的な問題クラスとして資源共有問題をモデル化する．

また実システムには不確実性や環境の変化といった特徴があり，MASにおいては分散強化学習 (Distributed Reinforcement Learning, DRL) による接近法が注目されている [10] が，上記の競合を解決する方法は知られていない．そこで我々は競合回避のために知識を報酬設計に導入し，副目的として**譲歩タスク**を定義することにより，資源共有問題を優先順位のある多目的意志決定問題として扱う．性能追求のための主目的と競合回避のための副目的の２つの価値関数を基に，多目的最適化の分野で弱いトレードオフによって導かれるpareto集合を効率的に求めるα-domination戦略 [4] を用いて行動空間全体から有望な行動を実行可能行動集合 (Feasible Action Set, FAS) として選別する．これにより行動空間が大きい場合でも探索を有望なFASに絞って行うことができる．我々は競合回避のための副目的導入による多目的化とα-domination戦略による探索の効率化により資源共有問題のシステム性能の向上を目指す．

提案手法の有効性を確認するために資源共有問題の一例である分散データベースシステム (Distributed Database Systems, DDBs) のトランザクション処理をモデル化したベンチマーク問題に適用し，いくつかの環境でシミュレーション実験により獲得した性能を示す．

以下では2章でMASが扱うタスクについて分類し，資源共有問題をモデル化する．3章では競合回避のための副目的を導入し，多目的の価値関数を基にα-domination戦略でFASを絞り込むDRLアルゴリズムを提案する．4章ではDDBsのトランザクション処理のベンチマーク問題に適用してその有効性を確認する．5章はまとめである．

2　問題設定

本章ではMASが対象とするタスクについて，その目的の観点から分類する．そして従来のDRLにおいて困難な問題の具体例を挙げ，資源共有問題をモデル化する．

2.1　競合を伴う協調タスク

我々はMASを環境内のエージェントが相互作用することにより何らかのタスクを達成することを目的とするシス

テムと定義する．まず，システム自体に目的がない場合とある場合に分類する．

システムに目的がない場合：これらの問題には，売買エージェントを扱う市場問題やオークションシステム，チェスや 1 対 1 の対戦ゲームなどが挙げられる．エージェントは効果的な売買や勝利といったそれぞれの目的があるが，システムは単に環境である．また各エージェントの目的を達成しようという試みは，たいてい他のエージェントと競合するため，本論文ではこれを**競合タスク**と呼ぶ．

システムに目的がある場合：これらの問題には，追跡問題や箱押し問題といった協同作業問題や次節で述べる資源共有問題などが挙げられる．システムには各エージェントの協調動作によって達成されるタスク（**システムタスク**と呼ぶ）が最も重要だが，分散環境であるために各エージェントは直接システムタスクを制御することはできず，それぞれ部分的なタスク（**部分タスク**と呼ぶ）を制御することになる．そこでこれを**協調タスク**と呼ぶ．

我々は協調タスク問題に興味がある．そこでさらに協調タスク問題をシステムタスクのために部分タスクが競合しない場合と競合する場合に分類する．

部分タスクが競合しない場合：これらの問題では一般に各エージェントによる部分タスクの達成はシステムタスクの達成につながる．そのため各エージェントはシステムや他のエージェントのタスクについて特別考慮せずに部分タスクの達成を目指せばよい．

部分タスクが競合する場合：これらの問題では各エージェントが利己的に部分タスクの性能を追求すると，競合によってシステムタスクの性能が低下する．そのため各エージェントは部分タスクの達成だけでは不十分で，システムタスクのために他のエージェントと協調的に相互作用する方法が必要になる．これを**競合する協調タスク**と呼ぶ．

このような競合する協調タスクは一般に多くの実問題に見られ，部分タスクの達成と協調することによるシステムタスクへの貢献のバランスを獲得しなければならない困難な問題である．困難さを説明するために，繰り返し囚人のジレンマを拡張した例題を用いる．囚人のジレンマ問題では，行動 (C or D) を提示したエージェント A_i はその組合せによって表 1 に示す利得 r_i を獲得する．これを繰り返すとき，利得表と相手の出した行動を基に合理的な行動選択を獲得する問題で，ゲーム理論によれば合理的なエージェントは，互いに行動 D を出し合う Nash 均衡政策を選択することが知られている [8].

ここで双方の利得の総和の最大化 $\max_{action}\sum_i r_i$ をシステムタスクと定義すれば，各エージェントは利得表より最大の $\sum_i r_i = 8$ を得る行動 C を選択できるかもしれない．

		A_1	
		C	D
A_2	C	(4,4)	(5,1)
	D	(1,5)	(2,2)

Table1: 利得表 1,(r_1,r_2).

		A_1	
		C	D
A_2	-	(4,-)	(5,-)
	-	(1,-)	(2,-)

Table2: 利得表 2,(r_1,-).

しかし我々は分散環境を考慮し，表 2 に示すように完全な利得表も相手の行動も観測することができず，結果的に得られる利得のみを観測できる問題を考える．得られる利得を基に r_i の最大化を部分タスクとする場合，相手の行動が一意に定まればいずれの場合も行動 D が行動 C よりも高い利得を得るので，双方のエージェントは互いに行動 D を選択し，システムは最低の利得 $\sum_i r_i = 4$ を得る．このときシステムタスクを向上させる協調行動は存在するが，情報が足りないために発見することは難しい．また行動 C を選択するとしたとき，同じ行動 C を選択しても相手次第で利得が 4 になったり 1 になったりするため，その行動の価値を決めることは容易ではない．

本研究では競合する協調タスクである資源共有問題に着目し，システムタスクを達成する協調的な行動政策を獲得できる手法を提案することを目的とする．

2.2 資源共有問題のモデル化

資源共有問題とは互いを観測できない複数のエージェントが必要な共有資源を排他的に確保してジョブを処理し，環境中の総処理量を最大化する問題である．

本問題において我々は各エージェントが相互作用の結果として得るそれぞれの利得以外，互いのエージェントの情報を全く得ることができないと仮定する．これはインターネットやイントラネット環境を通じて共有資源を扱うことが多くなりつつある近年の状況に合致する．

環境 $E = \{N, S\}$ はエージェント集合 N と資源集合 S からなる．各エージェント $n \in N$ はジョブ j_n を保持する．$s_i^n \in S$ は j_n の実行に必要な資源，m はその数である．

$$j_n = \{s_1^n, s_2^n, s_3^n, \cdots, s_m^n\} \tag{1}$$

ジョブに必要な資源 s_i^n には順序があるので，図 1 に示すようにエージェント n は s_i^n に対して $i = 0, 1, 2, \cdots$ の順に図 2 の要領で確保を試みる．

- s_i^n が他のエージェントによって確保されていなければ s_i^n を排他的に確保し，次に s_{i+1}^n の確保を試みる．
- s_i^n がすでに確保されている場合，解放されるまで利用できない．そこで期間 $timeout_n(s_i^n)$ の間待機し，解放されなければジョブを一度あきらめてこれまで確保した資源を全て解放し，s_1 から再度開始する．

必要な資源を全て確保したらジョブ j_n を処理して処理数 $commit_n$ を増やし，新たなジョブ j_n' を開始する．

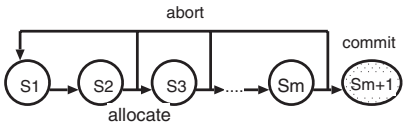

Fig.1: 資源確保の遷移図

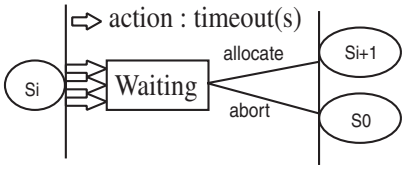

Fig.2: 資源確保の模式図

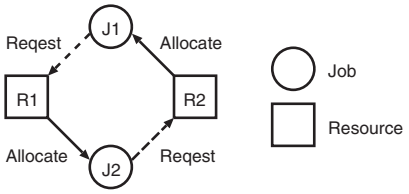

Fig.3: デッドロック発生の模式図

制御変数は各タイムアウト政策 $\pi_n \in \Pi$ である．

$$\pi_n = \{timeout_n(s_1), \cdots, timeout_n(s_m)\} \tag{2}$$

部分タスクは各エージェントが処理完了したジョブの総数の最大化であり，それらを基にシステムの目的である全エージェントが処理完了したジョブの総数の最大化を行う．

$$\begin{array}{ccc} \text{部分タスク} & & \text{システムタスク} \\ \max_{\pi_n} commit_n & \rightarrow & \max_{\Pi} \sum_n commit_n \end{array} \tag{3}$$

ジョブを確実に処理完了するためにはなるべく長いタイムアウト政策を選択し，必要資源を確保している他のエージェントの処理が完了するかあきらめて資源が解放されるまで待つことが有効である．しかし，長いタイムアウト政策は図 3 に示すような既に確保されている資源を待ち合うデッドロックを引き起こしやすく，他のエージェントの処理を妨げてしまう．このように資源共有問題は各エージェントの利己的な性能追求が競合を生み，システムの性能低下を招く競合する協調タスクの典型といえる．

3 接近法

資源共有問題では各エージェントが部分タスクの利得を追求すると競合が発生し，システムタスクの性能を低下させる．従って利得の追求と共に競合を回避する方法が必要であるが，環境中の他のエージェントと情報交換できないために，競合するエージェント同士が明示的に相互作用して競合を回避するような制御を行うことは困難である．そこで我々は競合の回避を目的とするタスクを導入し，多目的の観点で効率的に意志決定を行う方法を提案する．

3.1 競合回避のための譲歩タスク

我々は競合回避を目的とした副次的なタスクを導入し，これを**譲歩タスク**と呼ぶ．一般に譲歩タスクは問題依存であるため，ここでは資源共有問題を対象として議論する．

資源共有問題において競合は一時的なデッドロックが原因で生じる．デッドロックに陥るといずれかのエージェントがジョブを放棄して資源を解放しない限り，どのジョブも処理を完了できない．従って，競合を回避するためにはデッドロック状態をなるべく早く解消する必要があるが，エージェント間で情報を交換できないため，デッドロックを検出することは容易ではない．

そこで我々はデッドロックの検出は行わず，資源 s_i の解放を待つ時間 $timeout(s_i)$ が短いほどデッドロックになりにくいことを利用して譲歩タスクを定義する．

$$min_{\pi_n} \sum_i timeout_n(s_i^n) \tag{4}$$

ただしこれはシステムタスクに直結するものではないため副目的として扱う必要がある．譲歩タスクは競合回避を促進するものの，必要以上の譲歩はエージェントの利得の減少を招き，結果的にシステムタスクの性能も低下させる可能性があるからである．

主目的である部分タスクによる利得の追求は競合を生じ，副目的である譲歩タスクによる競合回避の追求は利得減少となることを考慮して，資源共有問題を双方のバランスをとる優先順位付の多目的意志決定問題として扱う．

$$\left.\begin{array}{l} \max_{\pi_n} commit_n \\ \min_{\pi_n} \sum_i timeout_n(s_i) \end{array}\right\} \rightarrow \max_{\Pi} \sum_n commit_n \tag{5}$$

3.2 α-domination 戦略に基づく分散強化学習

競合のために処理タスク数には不確実なノイズが含まれることは避けられない．このような場合，政策の長期的な評価を行う強化学習を用いることは有望である [10].

我々は各エージェントに主目的と副目的に関する価値関数をそれぞれ定義し，Q-learning[10] を用いて更新する．

$$\mathbf{Q}^n(x,a) = \{Q_c^n(x,a), Q_t^n(x,a)\} \tag{6}$$

$$\mathbf{Q}^n(x,a) = (1-\lambda)\mathbf{Q}^n(x,a) + \lambda(\mathbf{r_n} + \gamma \max \mathbf{Q}^n(x',a')) \tag{7}$$

ここで Q_c は主目的である処理数の価値関数，Q_t は副目的である政策の待ち時間の価値関数で，λ は学習率，γ は割引率である．報酬 $\mathbf{r}$ は双方の目的に関して与えられる．

$$\mathbf{r_n} = \begin{cases} commit_n & \text{for } Q_c \\ \sum_i timeout_n(s_i) & \text{for } Q_t \end{cases} \tag{8}$$

我々は 2 つの価値関数に基づく多目的の意志決定において，まず行動空間全体から有望な行動の集合 (FAS) を選択する．そして FAS を探索行動空間としてその中から主目的を基準に行動を一意に決定する方法を用いる．

多目的最適化の pareto 集合の観点から，図 4(左) に示すようにある程度有望な行動集合を選別することができる．

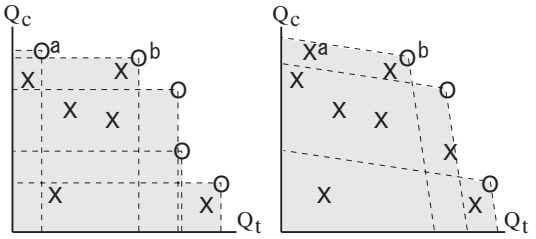

Fig.4: pareto 行動集合 (左) と α-pareto 行動集合 (右)．o は x をドミネイトする集合の要素で FAS に登録される．灰色部はドミネイト領域を表す．

しかし，”a”と”b”のついた行動に着目すると，行動 b に対してわずかに主目的の価値 Q_c が高いけれども副目的の価値 Q_t がかなり低い行動 a も pareto 集合に含まれてしまうことがわかる．行動 b に対して利得の向上は少ししか期待できないにもかかわらず，競合を引き起こすリスクが大きい行動 a は望ましくない．そこで我々は α-domination 戦略 [4] を用いることにする．α-domination 戦略は多目的な要素に対して，dominance 判定に弱いトレードオフを導入する手法で，dominance 抵抗解を pareto 集合から効率よく排除できる有効性が示されている．各価値関数間のトレードオフ比が厳密に設定できないことはしばしば言われることだが、その上下限を設定することは比較的容易である．

$$\alpha_{tc} \leq \frac{\Delta Q_c}{\Delta Q_t} \leq \frac{1}{\alpha_{ct}} \tag{9}$$

ここで ΔQ_c と ΔQ_t は Q_c と Q_t の変化量である

この弱トレードオフ関係を用いて図 4(右) に示すように「Q_c の観点で b が a に少々劣っていても、Q_t の観点で b が a におおいに勝る」場合は a を集合から排除する．この場合を b が a を α-dominate するといい，この集合を α-pareto 集合という．我々は α-pareto 集合を実行可能な行動集合 (FAS) として利用することにする．図 5 に意志決定アルゴリズムを略記する．

また α-domination 戦略は探索効率を向上させる効果も期待できる．ε-greedy 戦略などのランダム性を用いた探索戦略は単純さなどから良く用いられるが，行動空間が大きい場合，空間全域をランダムに探索するのは効果的ではない．また，他のエージェントとの競合を考慮すると影響の大きい行動をランダムに選択することは望ましくない．そこであるトレードオフの下で選別された FAS を探索空間として利用することにより，探索による競合の発生と無駄な探索を低減し，効率よく学習することが期待できる．

3.3 関連研究

環境を共有する意志決定エージェントで構成する MAS はゲーム理論の分野で古くから研究が行われ，特に Nash 均衡という観点から多くの有益な分析や知見が得られている [8]．しかしゲーム理論の観点では環境の完全観測性が前提にあり，資源共有問題のように情報が限定された環境

1．価値関数 **Q** を初期化．
2．状態 x を観測．
3．$\mathbf{Q}(x,a),\forall a$ に関して α-pareto 集合を FAS に登録．
4．FAS から $Q_c(x,a)$ に関して ε-greedy 選択．
5．行動 a を実行して報酬 **r** を観測し，式 (7) で価値関数を更新．
6．2に戻る

Fig.5: 意志決定アルゴリズム

を扱うことはできない．そこで協調行動を学習する政策勾配法 [9] や分散強化学習 [3] を Nash 均衡の観点で捉える研究がある．これらは分散環境に適用できるが各エージェントが競合する場合は扱うことができない．我々の知る限りでは分散環境で競合する協調タスクを直接扱うことができる効果的な方法は知られていない．

また MAS ではないが多目的性の取り扱いについては辞書的順序を利用した手法 [2] や目標領域に平均報酬ベクトルを指向する強化学習 [5] がある。これらの手法は意志決定そのものを多目的の観点から行うものであるが，我々は優先順位付の多目的性を利用して行動選択の候補を効率的に絞り込むことを考えている．

4 分散データベースシステムへの応用

本章では資源共有問題のベンチマークとして分散データベースシステム (DDBs) のトランザクション処理をモデル化した問題に提案手法を適用し，有効性と有用性を示す．

4.1 トランザクション処理とデッドロック

トランザクション処理とは，並列処理において個別に行われる動作の集合からなる計算処理をいい，統一概念として原子性，一貫性，分離性，持続性という ACID 特性が定義されている．各トランザクションはジョブが与えられると処理を開始し，処理の終了をコミット，途中で中止することをアボートという．アボートした処理は全て元通りに戻さなければならないなどデータの一貫性を維持するために参照や書換えが行われるデータをロックする方法が広く使用されている [1]．あるトランザクションがデータをロックすると，他のトランザクションはそのデータにアクセスできなくなりロックが外されるまで待機状態に入る．

このとき資源を一定時間待ったトランザクションをデッドロック状態にあるとみなし，強制的にアボートさせる方法をタイムアウトといい，単純かつ実用的なため広く用いられている．しかし，動的で不確実な環境において良い性能を引き出すタイムアウトの設定は非常に難しい [11].

4.2 実験設定

トランザクション自体の処理は詳細を観察することはできない．そこで本研究では図 6 に示す DDBs の各サイトをエージェントとし，そこから生成されるトランザクショ

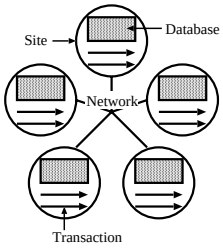

Fig.6: 実験モデル

Table3: 実験パラメータ

パラメータ	値
データベースサイト数	5
コミットまでに必要な資源数	8
並列動作トランザクション数	10〜200
CPU 処理時間	15 ms
I/O 処理時間	35 ms
他サイトへのリクエスト率	40 %
自サイトへのメッセージ遅延	1 ms
多サイトへのメッセージ遅延	100 ms

ンは同じタイムアウト政策に従うものとする．

トランザクションはコミットまでに複数の様々なデータの確保を必要とするため，式 (2) をそのまま政策とすることはできない．そこでトランザクション処理の特性を利用して確保データ数に関する関数としてタイムアウト政策を定義する [7]．a, b は政策パラメータである．

$$timeout(s_i) = a * b^{i-4}, \qquad i = 1, \cdots, m. \tag{10}$$

各サイトは政策パラメータ (a,b) を決定し，一定期間環境にトランザクションを生成してスループットを受け取る．

実験では表 3 に示す DDBs において各サイトの保持データ数が 100 の均一環境と，ひとつのサイトが 100 で他の 4 サイトが 300 の不均一環境に対して適用した．これは組合せ最良解を計算するために単純な設定を用いている．また並列実行数 (MPL:Multi Programming Level) を固定した定常環境で 10 から 200 まで実験した．MPL が大きいほど環境が混雑しデッドロックが起こりやすくなる．

実験期間は全体で 1.5×10^9ms，評価は最終 1.0×10^7ms 間の性能を示す．予備実験に基づき強化学習パラメータは学習率 0.1，割引率 0.9，α-domination 戦略の α は均一環境では MPL が 10 から 100 で 0.005，110 から 200 で 0.001 を用いた．不均一環境では全 MPL で 0.0001 を用いた．政策パラメータは a は 500 間隔で (500,7500) の範囲を 15 段階，b は 0.05 間隔で (1.05,1.5) の範囲を 10 段階に離散化した．従って各サイトは行動数 150 から意志決定し，システム全体ではサイト数 N に対して 150^N の組合せとなる．図 7 から図 10 は 10 試行の平均値をプロットしている．

従来運用に用いられる固定値のタイムアウト政策 (Fix Best) とスループット最大化を行う Q-learning 政策 (Q) と比較を行う．Fix Best は確保データ数に関係なく固定値のタイムアウト期間をとるが，図 7 では (500,20000) の範囲で離散的に探索して得られた最良値を示した．Q は各エージェントが部分タスクである処理数最大化 $max_{\pi_n} commit_n$ を行い，競合は考慮しない．また限界性能を示すために，全サイトを一括管理の下で政策パラメータを離散的に全探索して得られた組合せ最良解も示す．これは分散環境では実行不可能な方法であるが獲得性能の理論的な限界 (Theoretical Limit) として示す．

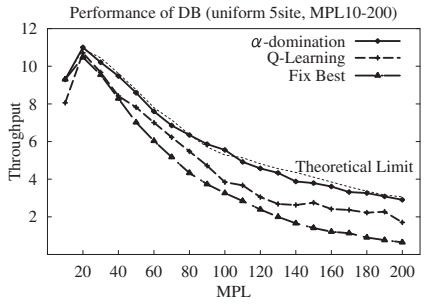

Fig.7: MPL10 から 200 の均一環境におけるスループット性能

α-domination 戦略の有効性を示すための実験では下式のように提案する副目的の報酬を主目的の報酬と α で重み付け線形和することにより単目的化する Q-learning(Linearly Weighted Sum Q, LWS-Q) との比較を行う．

$$r_n = (1-\alpha) commit_n + \alpha \sum_i timeout_n(s_i) \tag{11}$$

4.3 実験結果と考察

均一環境において各 MPL の時のスループット性能を図 7 に示す．従来運用で良く用いられている固定タイムアウト政策は最良の値を選択しても十分な性能が得られていないことがわかる．提案手法はほぼ全ての MPL においてスループット最大化を行う Q-learning を上回り，限界性能である組合せ最良解と同等の性能を獲得したことから，副目的の導入と α-domination 戦略による行動空間の絞り込みの有効性が確認された．

均一環境で MPL50 の時の学習曲線を図 8 に示す．学習序盤における性能差は，α-domination 戦略により早い段階から多くの価値の低い行動がドミネイトされるためと考えられる．報酬は他のサイトの政策により大きく変動するため，学習序盤の探索的な行動によって競合が起きやすいが，α-pareto 集合である FAS に有望な行動が登録されることでこの競合が早期に解消できている．また学習速度の違いは探索の効率化によるものと考えられる．

不均一環境において各 MPL の時のスループット性能を図 9 に示す．極端に混雑するサイトと他の 4 サイトとの性能はトレードオフ関係にある．組み合わせ最良解は混雑するサイトをある程度犠牲にすることで全体の性能を向上させる政策群であり，スループットは他のサイトの約 $\frac{1}{6}$

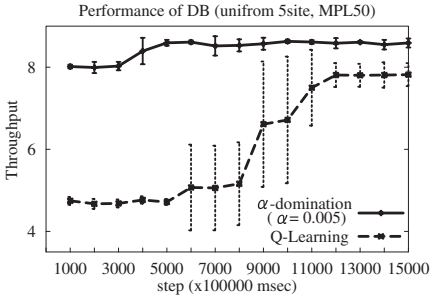

Fig.8: MPL50 の均一環境における学習曲線．エラーバーは標準偏差を表す．

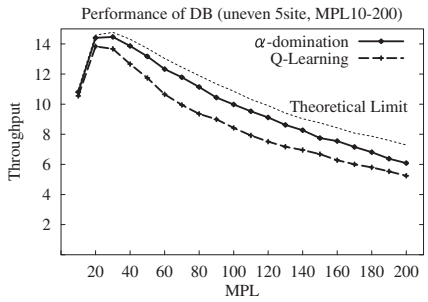

Fig.9: MPL10 から 200 の不均一環境におけるスループット性能

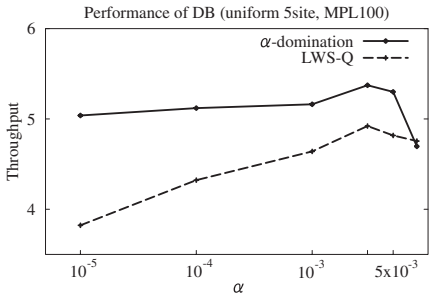

Fig.10: α とスループット性能との関係 (均一環境，MPL100)

程度となる．提案手法ではそのような政策群の他に，ある程度揃った政策群が学習される試行が確認された．このときスループットは他のサイトの約 $\frac{1}{3}$ 程度となる．これはスループットの総計を観測できないことと実験パラメータなどの設定が同一ためであると考えられる．ただしスループットを最大化する Q-learning よりも性能は良い．

4.4 α に関する考察と実験

α-domination 戦略において α は主目的と副目的の間の弱いトレードオフ関係を表すパラメータで，獲得性能に影響する．本問題において MPL100 以下の場合は経験的にスループット性能を 1 向上させるために 200ms から 300ms 程度までの待ち時間 $\sum_s timeout(s)$ の増加は許容できるが，それ以上は費用対効果が悪いことがわかっている [6, 7]. また待ち時間が増加するならスループット性能もかなり向上しないと許容しないといったように副目的を重視しすぎれば，性能に直結する主目的がないがしろになってしまうと考えられる．従って，α が大きすぎると主目的が達成されずに性能が急速に悪化し，3.0×10^{-3} から 5.0×10^{-3} 程度で適切なトレードオフ関係が得られて性能が良くなり，小さくなるにつれて性能が悪化すると予想される．

均一環境において MPL100 の時，α を変化させたときのスループット性能を図 10 に示す．予想通り適切なトレードオフ比である 3.0×10^{-3} から 5.0×10^{-3} で良い性能を示した．副目的を重視しすぎる α が大きい設定は本来用いないが，実験では全サイトが短いタイムアウト政策を選択し，急激な性能悪化が見られた．α を小さくしていくにつれて α-pareto 集合は増加していくが，相互作用を介して競合しづらい行動が結果的に学習されるために，性能の低下は徐々に起こることが確認された．

α を適切に調節することは適切なトレードオフを求めていることと同義である．そこで α をトレードオフ比とした LWS-Q において同様の性能が得られるはずである．しかし図 8 の学習曲線について述べたように，大きな行動空間での探索の困難さなどから全体的に性能が低いことと，特に α が小さいときは部分タスクが重視されて競合が発生し，性能が低下していく様子が確認されたという点で提案手法の結果と異なることがわかった．

以上のことから提案手法は LWS-Q と同様にトレードオフを用いるものの，α を適切に選択すれば LWS-Q よりも良い性能をロバストに獲得できることが確認された．

5 おわりに

我々は資源共有問題を対象に，競合回避のための副目的の導入と多目的意志決定の α-domination 戦略に基づく分散強化学習を提案した．資源共有問題は競合する協調タスクに分類され，分散環境ではシステムタスクの達成が困難であるが，優先順位付多目的意志決定問題にモデル化して提案手法を適用することでシステムタスクに関して良い性能を獲得できることを DDBs の課題で実験的に示した．

本論文では問題の特性上，状態空間について議論しなかった．今後は状況の変化に対する対応などを考慮し，競合する協調タスクの様々な課題に適用を試みる．

REFERENCES

[1] Bernstein, P., Newcomer, E. (1997) *Principles of Transaction Processing*. Morgan Kaufmann Publishers.

[2] Gabor Z., Kalmar, Z., Szepesvari, C. (1998) Multi-criteria Reinforcement Learning, *Proc. of the 15th International Conference on Machine Learning*, 197–205.

[3] Hu, J., Wellman, M.P. (2000) Experimental Results on Q-learning for General-Sum Stochastic Games, *Proc. of the 17th International Conference on Machine Learning*, 407–414.

[4] Ikeda, K., Kita, H., Kobayashi, S. (2001) Failure of Pareto-Based MOEAS: Does Non-Dominated Really Mean Near to Optimal?, *Proc. of the 2001 Congress on Evolutionary Computation*, 957–962.

[5] Mannor, S., Shimkin, N. (2002) The Steering Approach for Multi-Criteria Reinforcement Learning, *Advances in Neural Information Processing Systems 14*, 1563—1570.

[6] 大金義規 (2003) 分散データベースにおける協調機構の実現, 東京工業大学知能システム科学専攻修士学位論文.

[7] 大金義規，木村元，小林重信 (2003): 分散データベースにおける協調機構の実現，30th SICE Symposium on Intelligent Systems, 73–78.

[8] Osborne, M.J., Rubinstein, A. (1994) *A Cource in Game Theory.*, The MIT Press.

[9] Peshkin, L., Kim, K., Meuleau, N., Kaelbling, L. (2000) Learning to Cooperative via Policy Search, *Proc. of 16th Conference on Uncertainty in Artificial Intelligence*, 489–496.

[10] Sutton, R.S., Barto, A. (1998) *Reinforcement Learning: An Introduction*. A Bradford Book, The MIT Press.

[11] 魚田勝臣, 小碇暉雄.(1993): データベース, 日科技連.