分散データベースにおける協調機構の実現

○大金 義規，木村 元，小林 重信
東京工業大学　大学院総合理工学研究科

A Coordinaiton Mechanism for Distributed Database System

○ Yoshinori Ogane, Hajime Kimura, Shigenobu Kobayashi
Interdisciplinary Graduate School of Science and Eng., Tokyo Institute of Technology

Abstract: To resolve deadlocks on the transaction processing with distributed databases, the so-called timeout is used widely for convenience. However, it is a very difficult problem how to control the length of timeout period. This paper discusses coordination mechanisms for distributed database systems. We present a local search method that controls appropriate timeout periods adaptively. Its effectiveness is shown by several simulations under single agent environment. On the other hand, under multi-agent environments, if each agent behaves selfishly, the system performance will decrease rapidly. To avoid such a situation, it is necessary to introduce some constraint for coordination. We introduce a mechanism called the coordination bias. It suppresses selfish behaviors that move down the performance and promotes altruistic ones that move up it. Several simulations show that the local search method augmented by the coordination bias can improve the system performance stably under multi-agent environment.

1　はじめに

分散データベース上で実行されるトランザクション処理は，データの正当性を保証するためにロックを用いることが多い．しかし，ロックの概念は非常に単純であるにも拘らず，性能に与える影響，効率性，予測性などが不透明であるといわれている [1]．さらに，ロックを用いるとデッドロックという問題が発生する．デッドロックに対処する手法として検出による方法が知られているが，通信，計算コスト等の点において実用的ではない．この理由から，現実的方策として簡便でコストの低いタイムアウト手法が用いられることが多い．

しかし，タイムアウト間隔の設定は非常に難しく，安易に設定すると，システムのスループット性能を低下を招く恐れがある．これまでタイムアウト間隔をどのように設定すれば，妥当であるかを検討する研究は少なく，適切に設定する理論や方法は確立されていなかった．唯一，強化学習を用いたタイムアウト間隔の適応的な制御方法が後藤によって提案されているに過ぎない [2].

後藤による先行研究では，トランザクションのロック処理の流れを逐次決定過程として定式化し，近似的にマルコフ決定過程として扱っている．後藤は，Q-learning による最適タイムアウト間隔の適応制御の獲得を目指したが，デッドロックが頻発する環境では，Q-learning による学習は成功するに至らなかった．これは，対象問題が複雑なマルチエージェント系であり，非マルコフ性を有していたためと考えられる．

本研究では，マルコフ性の仮定を必要としない Local Search(LS) を用いて，デッドロックが頻発する環境においても適応的な学習を可能とする手法の確立を目指す．ところで，各サイト (エージェント) がそれぞれ利己的な政策ばかりを選択すると，エージェント間に競合関係が生じて，系全体として良好な性能を出せないことが予想され，実際に確認されている．本研究では，そのような問題に対処するために，まず，各エージェントの利得関係がマルチエージェント問題としては特殊であることを確認した．そこで，その利得関係をうまく利用したヒューリスティクスを提案し，LS に組み込む．

シミュレーションにより，環境に応じて適切な政策の獲得とスループット性能向上を確認することによって，提案手法の有効性を示す．

2　トランザクション処理

トランザクションとロック

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う．全ての操作は完遂されるか全く行われないかのどちらかであり，並列動作するトランザクションから中間的な状態を観測することはできない．トランザクション処理の統一概念として，原子性 (処理が完遂されるか，されないか)，一貫性，分離性 (他の影響を受けない)，持続性 (処理結果が失われない) という ACID 特性が定義されている．

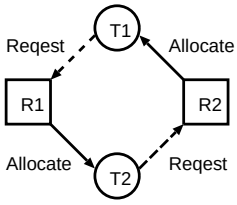

Fig.1: 資源割り当てグラフによるデッドロック発生の模式図

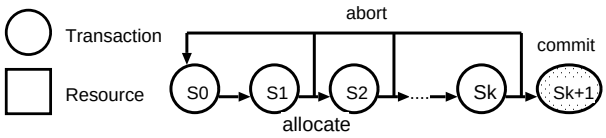

Fig.2: トランザクションの状態遷移図.

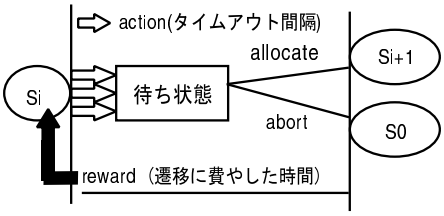

Fig.3: 逐次決定過程の模式図.

トランザクション処理は予め数種類用意されており，各トランザクションは該当するタスクが与えられると処理を開始する．そして，全てのタスクが終了すると処理完了となる．これをコミット (commit) という．反対に，途中で処理を中止することをアボート (abort) という．アボートした処理は全て元通りに戻されることが保証されなければならない．この元に戻す処理をロールバックという．

トランザクション処理では，データの一貫性を保持するために参照や書き換えが行われているデータをロック (lock) する方法が広く使用されている．ロックとは，そのトランザクションが使用するデータのアクセスを予約する機構をいう．あるトランザクションがデータにロックを掛けると，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．

デッドロックとタイムアウト

デッドロック (deadlock) は，Fig 1 のように，すでに保持されている資源に対して，互いのトランザクションがアクセスすることによって発生する現象である．デッドロックの問題点は，デッドロック状態に陥ったトランザクションが，次の処理を中断をできないことにある．これにより，トランザクション処理の機能不全，性能劣化を引き起こす危険性がある．この問題の対処法として，デッドロック検出によるものとタイムアウトによるものがある．

デッドロック検出について，Fig 1 で示されるようなループ構造をしている箇所を特定する手法，CMH アルゴリズム [3] によるトランザクション間のメッセージ交換による手法などが存在する．しかし，分散データベースシステム上では，計算コスト，通信コストが極めて高いという問題点がある．

デッドロックの発生が稀であるという仮定の下で，一定時間資源に対して待ち状態に入ったトランザクションは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がある．これをタイムアウトと呼び，単純かつ実用的であることから現実のシステムで広く用いられている．

しかし，良い性能を引き出すタイムアウト間隔の設定を行うことは非常に難しい [4]．なぜならば，時間の経過とともに変化していくシステムに応じて，適宜チューニングしなければならないためである．

そこで，本研究では，環境変化に追従して最適に近いスループット性能を達成できるようなタイムアウト間隔を適応的に設定する学習手法を提案する．スループットとは，単位時間あたりのコミット数を示しており，この値を最大化することによってシステム全体の処理性能の向上を目指す．

3 学習によるタイムアウト間隔の制御

3.1 トランザクション状態遷移の定式化

先行研究 [後藤 01] において，トランザクションの保持資源数に対してそれぞれ異なるタイムアウト間隔を設定する手法が提案されている．トランザクションに優先度を付けることによって，タイムアウト間隔を一定とする既存手法よりもスループット性能を向上することが期待できる．そのため，本研究でもこの枠組を用いる．

1 つのトランザクションが処理を開始し，終了するまでの流れは図 2 に示される．各待ち状態 S_i において，資源が利用可能かどうかによって状態遷移が発生する．このトランザクションの流れは逐次決定過程として捉えることができる．逐次決定過程は，(状態表現，行動表現，遷移確率表現，報酬表現) で表現される．以下では，各表現の設定を述べ，逐次決定過程の模式図を Fig 3 に示す．

状態表現 状態は，トランザクションが現在確保している資源数とする．

行動表現 行動は，各状態において，どのくらいの期間資源を待ち続けるかを表すタイムアウト間隔とする．

遷移確率表現 トランザクションが，確保している資源数による状態 s_i で，あるタイムアウト間隔行動 a をとった後，次状態 s'(アボートした場合 $s' = s_0$，資源確保した場合 $s' = s_{i+1}$．) に遷移する確率とする．

報酬表現 次状態に遷移するまでにかかった時間を負の報酬とする．

以上のような表現により，各状態行動対における Value は，トランザクションが各状態から終了するまでの期待平均生存期間の負値を表すことになる．よって，状態 0 における Value は，トランザクションが生成されてから終了するまでの期待平均生存期間の負値となる．この value を最大化することは，処理時間の最小化，すなわち，スルー

プット最大化を行うことと近似的に等価である．

3.2 先行研究による接近の問題点

先行研究では，この逐次決定過程がマルコフ決定過程に従っているとみなし，強化学習の Q-learning 手法を用いてタイムアウト間隔制御を行った．しかし，各サイトにおいて，トランザクションがデッドロックを頻発する環境ではうまく学習を行うことができなかった．これは，各サイトが意志決定を行うマルチエージェントであり，さらに，サイト内部を走査する複数のトランザクションも相互作用するという複雑な問題である．これにより，強い非マルコフ性を有していたと考えられる．

1 初期政策 π_1 を生成し，長期間，π_1 でシステムを動作させ，推定平均待ち時間 $V^{\pi_1}(0)$ を得る．

2 政策 π_1 における複数の近傍の中から，確率的に政策 π_2 を選択する．

3 前の評価で得られた推定値 $V^{\pi_1}(0)$ を近傍政策 π_2 の政策評価における初期値として利用することにより，短期間，π_2 でシステムを動作させ，推定平均待ち時間 $V^{\pi_2}(0)$ を得る．

4 推定平均待ち時間 $V^{\pi_2}(0)$ が，推定平均待ち時間 $V^{\pi_1}(0)$ を上回るならば，試行政策 π_2 を現在の最適政策 π_1 に代入し，同様に，value 推定値も代入する．そして，2 に戻る．そうでなければ，現在の最適政策を更新せずに 2 に戻る．

Fig.4: TD 法による政策評価のアルゴリズム．

3.3 エージェントの学習

非マルコフ性の環境下において，政策探索を行う手法として Direct Policy Search という枠組がある．Direct Policy Search は，マルコフ性などの強い仮定なしで直接的に政策空間を探索する手法の総称概念である [5]．

逐次決定過程で表現された各状態におけるタイムアウト間隔を探索すべき 1 次元と設定することにより，数次元の探索問題に帰着させる．探索手法として，ローカルサーチ (LS) を用いる．この理由は，高々数次元程度の探索問題であること，また，オンライン学習で環境に追従させることを考えると，GA などの集団を用いた探索手法は適していないなどが挙げられる．

LS による政策の評価値として，政策を固定した Temporal Difference(TD) 法で得られる推定値 $V(0)$ を用いる．この方法は，評価値としてスループットを用いる方法より，政策評価期間を短縮できると考えられる．なぜならば，各状態には，政策によって得られる value 推定値が保存され，その value 推定値は，次の政策評価のための初期値として利用できるからである．このアルゴリズムを Fig 4 に示す．

3.4 協調機構の提案

各サイトがそれぞれ利己的な政策ばかりを選択すると，競合関係が生じて，系全体として良好な性能を出せないことが確認されている．この問題を回避するために，一般的なマルチエージェントの利得関係に対して本マルチエージェントが特殊な利得構造であることを定性的に考察し，そのことを確認する．ついで，その特殊な利得構造を利用したヒューリスティクスとして協調バイアスを提案し，実験によりその有効性を確認する．

本マルチエージェント利得関係の考察

本問題における自エージェントの利益は，他エージェントの影響だけではなく，自サイト内部におけるトランザクションの流れがスムーズであるかによって決定される．それは，自エージェントが利己的すぎる政策を保持しているとき，顕著に現れる．ここで，更に利己的近傍を選択する場合，トランザクションの流れが悪くなり，性能向上を妨害する恐れがある．また，利他的近傍を選択する場合，自サイト内部のトランザクションの流れがスムーズになるため，ある程度自分の損益を打ち消すことが可能である．

以上の考察から，トランザクションの流れの影響で利得構造が特殊となっていることが考えられる．この特殊な利得関係を有していることを実験により示す．実験のために，以下の事項を定義する．各サイトにおいて，近傍探索の評価値から現政策の評価値を引いた差分を gain(利得) という指標で表現する．gain がプラスならば性能は向上し，マイナスならば性能は低下したといえる．gain 分布図を Fig 5 に示す．サイト 1 を自エージェントとし，サイト 2〜5 をまとめて相手エージェントと設定する．解析を簡単化するため，相手エージェントの政策を固定し，自エージェントが近傍を選択した時の各エージェントの利得関係を観測する．

利己的な近傍選択

自エージェントが利己的政策を保持している状況 (groupA) では，先の考察と同様，fig 6 の結果から実験でも得られることを確認した．それに対して，両者が利他的政策を保持している状況 (groupB) では，自エージェントが大きな利益を得て，系全体としてプラスとなる近傍がある．利己的な近傍を選択したとき，Fig 8 右のように，ある程度の gain が得られない場合は受諾しない閾値を設けると，この 2 つの group を区別することが可能である．

利他的な近傍選択

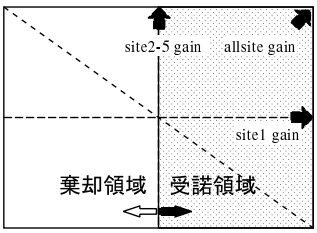

Fig.5: gain 分布図.

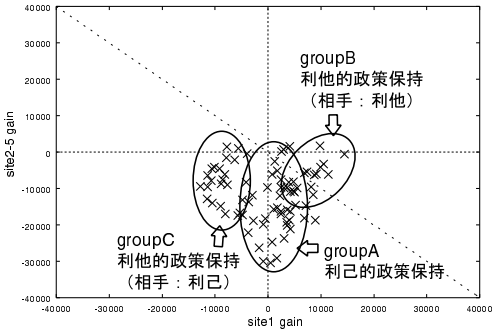

Fig.6: 利己的な近傍選択時の利得関係.

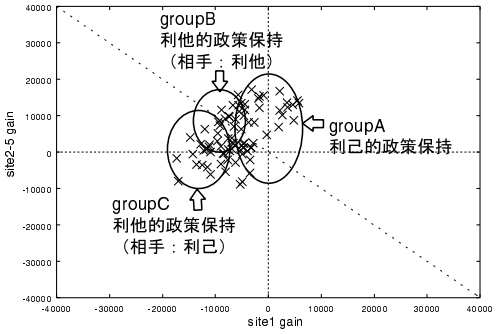

Fig.7: 利他的な近傍選択時の利得関係.

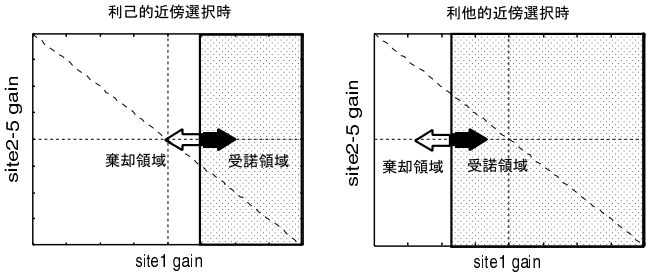

Fig.8: 協調バイアスの例.

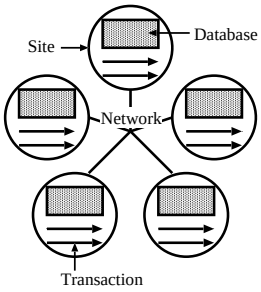

Fig.9: 実験モデル

Table. 1: 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

自エージェントが利己的政策を保持している状況 (groupA) では，先の考察と同様，fig 8 の結果から実験でも得られることを確認した．それに対して，利他的政策を保持している状況 (groupB，C) では，更に利他的な近傍を選択しても，自分が大きな損失を被るだけで，相手の利益にはならない．利他的な近傍を選択したとき，Fig 8 右のように，gain のロスが許容範囲内であれば受諾する閾値を設けると，groupA と groupB，C を区別することが可能である．

以上をまとめると，利己的過ぎるエージェントに対して，系全体をマイナスにする利己的な近傍の受諾を棄却し，系全体をプラスにする利他的な近傍の受諾を促進することができる．このようなメカニズムを協調バイアスと呼ぶことにする．協調バイアスの閾値設定は，ある程度妥当であれば確実に利他的な近傍受諾が促進されるのでパラメータチューニングにより設定する．

4 実験モデルによる性能評価

4.1 問題設定

シミュレーション実験を行うにあたり，複数のトランザクションが走行する分散データシステムを想定した問題設定を行う．これは，先行研究の設定を引き継ぐものとする．実験モデルやパラメータを，Fig 9 ，Table 1 に示す．これは，基本的に [6],[7] にしたがって設定した．実験モデルは，Fig 9 のように，5 サイトは密に繋がっており，各トランザクションはサイト間を渡って資源を獲得していく．

本研究では，各サイトの資源数が一定である均質 (homogeneous) な環境，ばらばらである不均質 (hetrogeneous) な環境という 2 種類の分散データベースモデルを設計する．均質な環境は，先行研究と提案手法を比較するとともに，均質という全体の挙動が理解しやすい設定から結果が分析しやすいと考えられる．しかし，この環境では，マルチエージェントの能力は発揮されない．それに対して，不均質な環境では，各サイトの規模に応じた適応的なマルチエージェント学習が行えることが期待できる．また，実環境を考えた場合，このような不均質な環境を想定する方が自然である．

本研究では，政策を一定時間システムに投入して評価を得る．その政策評価期間として，1.0×10^5step を設定した．5.0×10^8step 分実験を行い，最終 1.0×10^7step をスループットによる性能評価期間とした．また，システム全体で実行中のトランザクション数 (MPL:Multi Programming Level) を一定に保った状態で実験を行う．また，政策評価にノイズが生ずる環境といえる．そのノイズの影響で局所解に陥る可能性があるが，低い確率で政策の再評価を行うことで対処する．

4.2 LS の近傍設計

本研究では，各保持資源数に対する 7 つのタイムアウト間隔において，図 10 で示されるように，いづれかのタ

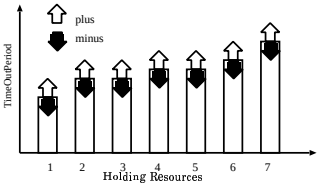

Fig.10: LS の近傍表現

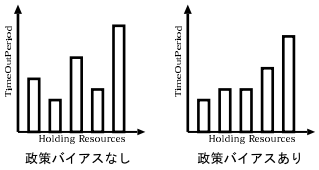

Fig.11: 政策バイアス設定

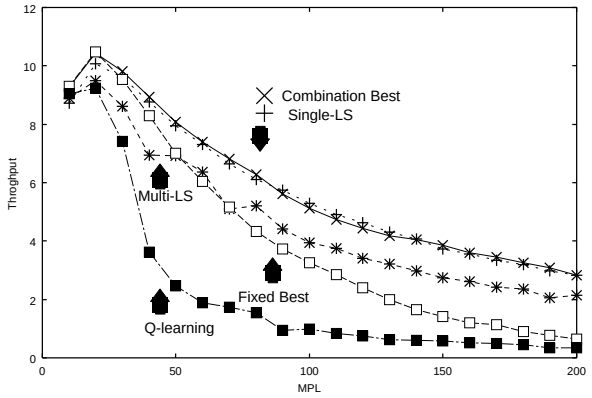

Fig.12: 均質環境下でのスループット性能

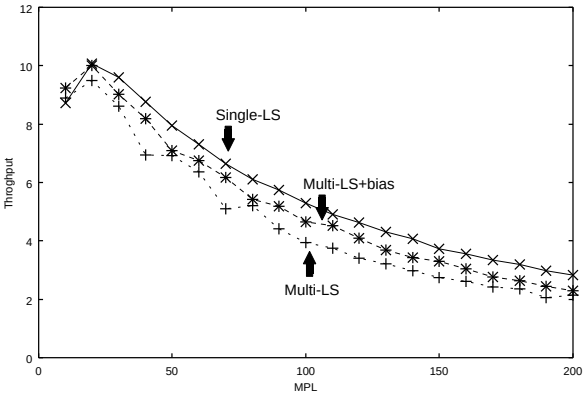

Fig.13: バイアスを導入した均質環境下でのスループット性能

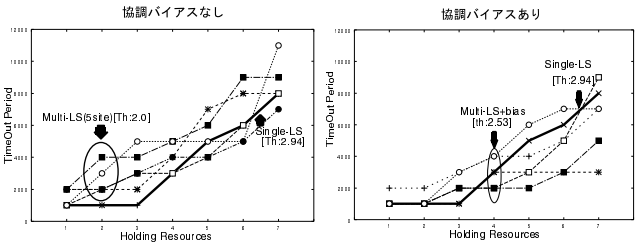

Fig.14: 獲得した政策表現

イムアウト間隔を ±1000 したものを近傍を設定する．

しかしながら，この設計のみでは，図 11 左側のような政策を獲得してしまい性能が向上しないことを確認している．そこで，このような領域の探索を禁止するため，保持資源数に対してタイムアウト間隔は広義単調増加するという制約を加えた政策バイアスを導入する．これにより，図 11 右側のような性能向上が見込める領域でのみ探索を行える．

4.3 実験に使用する提案・比較手法

- 提案手法
  Multi-LS – 各サイトがそれぞれ LS.
  Multi-LS+bias –協調バイアスを導入した LS.

- 比較手法
  Single-LS – 一つの政策を全サイトが共有する LS．マルチエージェント競合の影響を受けない．
  Q-learning – 先行研究による Q-learning 手法．
  Fixed Best – 各トランザクションのタイムアウト間隔を一定にする既存手法．
  Combination Best – いくつかのタイムアウト間隔を総当たりで組み合わせた網羅探索手法．オンライン処理で用いることは不可能である．

4.4 均質な環境での実験

均質な環境として，各資源数を 100 と設定した環境で実験を行い，結果を図 12 に示す．これにより，以下のことが考察される．

- Single-LS は，各 MPL 環境において Combination Best と同等の性能が得られている．この結果は，LS による学習が有効であり，その妥当性を示している．
- Multi-LS は，MPL が高い環境において，既存手法や Q-learning 手法より 2〜3 倍程度性能が向上している．このような劣悪な環境においても頑健に働く手法といえる．
- Multi-LS は Single-LS には及ばない．原因としてサイト間の競合が発生していると考えられる．これは，自己最大化のために，各サイトが系全体の性能を押し下げるような利己的な政策を受諾していくことで発生する．

このような問題を回避するために，提案した協調バイアスを導入した実験を行い，結果を図 13 に示す．

- 各 MPL 環境下において Multi-LS+bias の性能が，Multi-LS に比べ向上していることが確認できる．これは，利己的な近傍のみを受諾し続けることを抑制するメカニズムの導入により，競合が低く抑えられたことを示している．

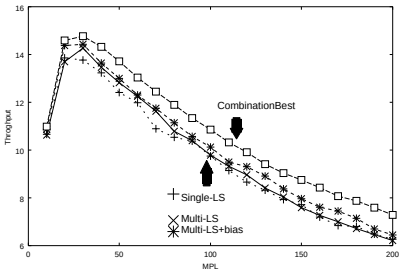

Fig.15: 不均質環境下でのスループット性能

Table. 2: 不均質環境における政策表現

Single-LS	政策表現	スループット 6.565
group1(1,2,3,4)	1265 1746 2409 3325 4589 6333 8740	1.554 1.477 1.533 1.5
group2(5)	**1265 1746 2409 3325 4589 6333 8740**	0.501
Multi-LS+bias	政策表現	スループット 6.84
site1	2124 2676 3372 4248 5353 6745 8499	1.634
site2	1140 1665 2431 3549 5182 7566 11047	1.466
site3	2752 3688 4942 6623 8875 11892 15936	2.002
site4	1281 1768 2441 3368 4648 6415 8853	1.461
site5	**780 1077 1486 2051 2831 3907 5392**	0.278
CombinationBest	政策表現	スループット 7.331
group1(1,2,3,4)	1625 2194 2962 4000 5400 7290 9841	1.791,1.8,1.786,1.649
group2(5)	**910 1183 1538 2000 2600 3380 4394**	0.305

- Fig 14 は，MPL=200 の環境において学習の結果獲得した政策表現である．協調バイアスを導入した政策の方が，保持資源数の少ない状況でのタイムアウト間隔を低く抑え，性能向上が達成できている．

4.5 不均質環境における実験

4.4 の均質な環境における実験と同様に，不均質な環境においても Multi-LS+bias を適用する．各資源数をサイト順に 300,300,300,300,100 とし，サイト 5 のみが混雑した環境で実験を行い，結果を図 15 に示す．なお CombinationBest は，サイト 1〜4 を group1，サイト 5 を group2 として政策のすべての組合せを試行した．

- 各 MPL 環境において，CombinationBest は Single-LS に比べて性能が向上していることが確認できる．これは，各サイトがマルチエージェントとしての能力を十分に発揮した結果といえる．学習による Multi-LS+bias においてもわずかながら上回っている．
- MPL=200 における Multi-LS+bias で獲得した政策表現を表 2 に示す．サイト 5 における政策表現とスループット値に注目すると，Single-LS に対して Multi-LS+bias と CombinationBest は，サイト 5 のタイムアウト間隔を低く抑えることで，他サイトのトランザクション処理を優先させ系全体のスループット向上を達成している．

5 考察

実験結果により，本研究で使用した LS が従来手法，Q-learning による先行研究手法に対して，良い性能を得られた．この理由は，マルコフ性という強い制約に縛られない Direct Policy Search より接近したことが有効であったためと考えられる．

各サイトの利得関係の解析により，本問題では，ある水準の利己的な政策を獲得してしまうと，それ以上の利己的な政策を選択しても大きな利得を得られないことが確認された．このような現象を踏まえて，大きな利得を得ることができない状況下では，利己的な政策を受諾しないよう制限した協調バイアスを提案した．これを LS に組み込むことによって，マルチエージェントの競合を緩和し，スループット性能向上に繋がる協調メカニズムを実現した．

6 結論

本論文では，分散データベースのトランザクション処理において，LS を用いてタイムアウト間隔の適応な制御を実現し，混み合った環境下でのスループットの向上を示した．また，協調バイアスと呼ぶ協調メカニズムを導入することにより，サイト間の競合を緩和させ，性能向上を達成できることを示した．

今後の課題として，不均質な環境でより多くの実験，協調バイアスの洗練化，また，本研究の目的であるタイムアウト値の適応制御の実用化が挙げられる．

参考文献

[1] Philip Bernstein,Eric Newcomer: "Principles of Transaction Processing",*Morgan Kaufmann Publishers*, 1997.

[2] 後藤 正徳: " トランザクション処理におけるタイムアウト間隔の学習", 第 28 回知能システムシンポジウム，pp.257-262，2001.

[3] Pradeep K.Sinha:"Distributed Operating Systems -Concept and Design",*the Institute of Electrial & Electronics Engineers,Inc.,*1997.

[4] 魚田 勝臣，小碇 暉雄: " データベース"，日科技連，1993.

[5] Jurgen Schmidhuber,Jieyu Zhao: "Direct Policy Search and Uncertain Policy Evaluation", *American Assocation for Artificial Intelligence,Menlo Park,Calif,pp. 119-124*,1999.

[6] Rakesh Agrawal,Michael J.Carey,Lawrence W.McVoy: "The Performace of Alternative Strategies for Dealing with Deadlocks in Database Management Systems",*IEEE Transactions on Software Engineering,Vol.SE-13,No12*,Deceber 1992.

[7] Omarn Bukhres:"Performance Comparison of Distributed Deadlock Detection Algorithms",*8th International Conference on Database Engineering*,pp.210-217,IEEE,1992.