PC クラスタを用いた Profit Sharing の並列化手法

串田 正幸　高橋 健一　上田 祐彰　宮原 哲浩

（広島市立大学大学院　情報科学研究科）

1. はじめに

強化学習は試行錯誤を通じて環境に適応していく学習制御の枠組みであり，環境との相互作用を繰り返し，最適な政策を学習することが目的である．しかし，強化学習は知識のない状態から試行錯誤により学習を行うため学習に時間がかかる．つまり，膨大な量の計算を要する．また，近年，高性能で安価なパーソナルコンピュータ PC を手に入れることができるようになっている．そこで，膨大な計算量を必要とする処理を高速に実行するための方法の一つとして，複数の PC を用いて計算量を分割し，並列に処理させる強化学習の並列化が提案されている[1][2].

本研究では，強化学習の代表的な手法のひとつである Profit Sharing[3]を用いた並列型強化学習手法を提案し，逐次型 Profit Sharing との性能比較を行う．

2. システム構成

本研究では RedHatLinux7.3 を実装した PC16 台(Master1 台，Slave15 台)を用いて，分散メモリ型の PC クラスタを構成している．各 PC の CPU は Pentium4 プロセッサ(2.2GHz)であり，主記憶は，Master プロセスに 512MB，Slave プロセスに 256MB を搭載している．通信は 1000BASE - T に対応した 24 ポートスイッチング HUB と LAN アダプタを使用している．ここで，各プロセッサ上で動いているプログラムやサブプログラムをプロセスと呼ぶ．プロセス間のデータの受け渡しの通信方法はメッセージパッシング方式である MPI(Message Passing Interface)を用いる．

3. Profit Sharing

Profit Sharing は，報酬に至るまでのエピソードにおける，状態 s と実際に行った行動 a の対からなるルール系列を記憶しておき，報酬が得られたときにそれまでの系列上のルールを一括して強化する学習手法である．また，各ルールは重みを保持しており，この重みを式(1)で強化することによって学習を行う．ここで $w(s_i,a_i)$ はエピソード系列上の i 番目のルールの重み，r は報酬値，f は強化関数である．強化関数 f は，r および i を引数とし，強化値を返す関数である．

$$w(s_i,a_i) \leftarrow w(s_i,a_i) + f(r,i) \qquad (1)$$

あるエピソードで，同一の状態に対して異なるルールが選択されているとき，その間のルール系列を迂回系列という．迂回系列上のルールは，報酬の獲得に貢献しない可能性がある．現在までの全てのエピソードで，つねに迂回系列上にあるルールを無効ルールと呼び，それ以外を有効ルールと呼ぶ．無効ルールと有効ルールが競合する場合，明らかに無効ルールを強化するべきでない．無効ルールの重みが最大でないとき，無効ルールは抑制されていると呼ぶ．式(2)を満たす強化関数であれば，有効ルールは無効ルールを抑制できることが合理性定理により証明されている．

$$L\sum_{j=i}^{W} f(r,j) < f(r,i-1) \qquad \forall_i = 1,2,\ldots,W \qquad (2)$$

ここで，W はエピソードの最大長，L は同一感覚下に存在する有効ルールの最大個数である．条件を満たす関数は様々存在するが，最も簡単なものとして式(3)のような等比減少関数が強化関数として考えられている．

$$f(r,j) = \frac{1}{S} f(r,j-1) \qquad j = 1,\ldots,W-1 \qquad (3)$$

ここで，S は報酬割引率であり，$S \geq L+1$ とする．

4. 並列 Profit Sharing

並列 Profit Sharing の学習モデルは Master/Slave 型である．Master プロセスと Slave プロセスでは役割が異なり，Master プロセスが全体をコントロールし，Slave プロセスが学習を行う．また，一定の間隔（更新間隔と呼ぶ）で情報を更新する．

4.1. 状態分割並列 Profit Sharing

状態分割並列 Profit Sharing ではエージェントの学習に Profit Sharing を用いる．状態を分割して各プロセスに割当て，並列に割当てられた状態 s と行動 a の対からなるルールの重みを更新し，学習を行う．図１に，迷路問題において，プロセス数４台としたときの分割の様子を示す．番号はプロセスの番号，G はゴール地点を表す．

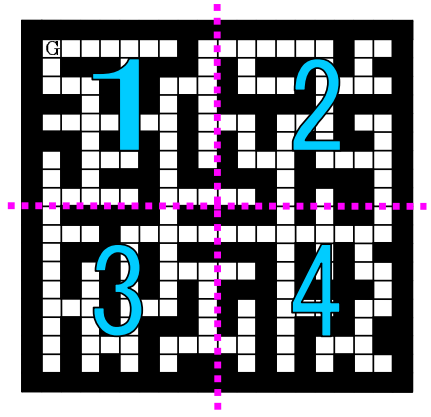

図１　分割の例

Profit Sharing は，報酬に至るまでのエピソードにおけるルール系列を記憶しておき，報酬が得られた際にそれまでの系列上のルールを一括して強化する．このため，図１のような G に到達すると報酬を得ることができ，それ以外には報酬が得られないという環境で状態を分割して並列化を行った場合，G を含まないプロセスでは報酬の伝達が必要となる．

そこで，他プロセスが担当する領域に達したときの重みを更新する方法として，以下の３つの方法をそれぞれ実装・実験を行った．

① 隣接する状態が得た最新の強化値を報酬の代わりに与え，強化関数によりルールの重みを更新する．最新の強化値とは，その状態が学習中

に最後に得られた強化値である．
② 状態が更新間隔中に得た最大の強化値を報酬の代わりに与える．
③ 更新間隔を問わず，状態が学習中に得る最大の強化値を報酬の代わりに与える．

4.2. 状態分割を行わない並列 Profit Sharing

状態分割を行わない並列 Profit Sharing では,状態を分割せず元の（分割していない）環境を各プロセスに割り当て，並列に学習・重みの更新を行う．ルールの重みの更新は，学習したルールの重みと Master から更新間隔ごとで受け取ったルールの重みとの差分を Master に送信する．Master は，保持するルールの重みに受信したルールの重みを足し合わせ，Slave に送信する．しかし，状態数が多くなった場合，状態分割並列 Profit Sharing が他領域と隣接している箇所の重みだけを通信すればいいのに対して，環境を分割していないため，環境全ての重みを更新するため通信に負荷がかかると考えられる．

そこで,slave ごとに担当領域を限定して割当て，割当てられた領域だけの重みを更新する．限定されているのは，更新する重みの領域だけで，Master から重みを受け取る際は全領域の重みを更新する．エージェントは全領域を学習し，担当領域外の領域もローカルで学習する．更新する重みの領域を限定することによって，通信の負荷を減らすことができる．

5. 実験，評価

本手法を評価するため，迷路問題，格子世界問題に適用し実験を行った．ここでは，迷路問題の結果を示す．学習の終了条件として，あらかじめ決められたエピソード回数に達したら終了とする．終了条件としたエピソード回数は，予備実験として学習させ,収束したと考えられるエピソード数としている．

図 2 に，迷路サイズ（101×101），学習パラメータとして，報酬割引率$1/S=0.5$，学習の終了条件として試行回数=32000, PS.v1~v3 の更新間隔=100, PS.v4~v5 の更新間隔=500 と設定しプロセス数を増やしたときの学習時間を，図 3 に学習の評価を計測した結果を示す．学習の評価は，全ての位置からエージェントが出発（スタート）し，目的地（ゴール）へ到達することのできるスタート位置の全位置に対する割合を用いる．この割合を成功率と呼ぶ．PS.v1~v3 は状態分割並列 Profit Sharing の①～③の手法を，PS.v4 は状態分割を行わない並列 Profit Sharing を，PS.v5 はその領域限定手法，PS は逐次型 Profit Sharing を表している．

図 2 より，更新間隔を広げプロセス数を増加させることで，逐次型 Profit Sharing より学習時間を短縮していることがわかる．状態分割並列 Profit Sharing と状態分割を行わない並列 Profit Sharing を比較すると，状態分割を行わない並列 Profit Sharing の方が学習に時間がかかっている．これは，環境を分割していないため 1 試行の学習時間が長いことと，更新する状態数が多いためであると考えられる．PS.v5 は更新する重みの領域を限定することで，PS.v4 に比べ通信の付加が減少しているため，学習時間が短いことがわかる．

図 3 より，状態分割を行わない並列 Profit Sharing は成功率が逐次型 Profit Sharing とほぼ変わらない．これは，直接報酬を得ることができているためである．逆に，状態分割並列 Profit Sharing では，状態を分割しているためプロセス数が増加すると成功率が低下しやすい．特に，PS.v2 では，プロセス数が増え分割する領域が増えた場合，強化値の受け渡しが必要な箇所が増え，強化値が 0 となる回数が増え，成功率が低下している．PS.v2 は成功率が大きく低下しているが,並列化を行なうことで,学習の評価を保ったまま学習時間を短縮することができることがわかる．

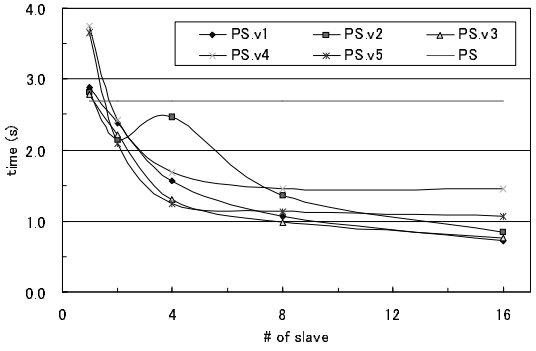

図 2　学習時間

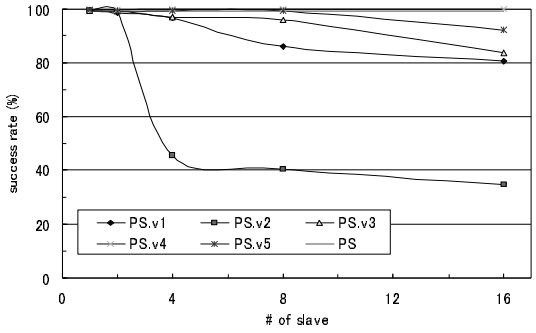

図 3　成功率

6. おわりに

本研究では，Profit Sharing の並列強化学習手法を提案し，問題に適用，実験を行い，逐次型 Profit Sharing と比較を行った．実験結果から，並列化することで，学習の評価を保ったまま学習時間を短縮することができることを示した．

今後の課題として，並列 Profit Sharing の改良，更新間隔の検討，新しい問題に対しての評価，他の並列型強化学習手法との性能の比較が挙げられる．

なお，本研究は広島市立大学特定研究費（No.6110）による支援を受けた．

参考文献

[1]A.M.Printista,M.L.Errecalde,C.I.Montoya, “A Parallel Implementation of Q-Learning Based on Communication with Cache”, Journal of Computer Science & Technology, Vol.6, 2002
[2] Masayuki Kushida, Kenichi Takahasi, Hiroaki Ueda, Tetsuhiro Miyahara, “A Comparative Study of Parallel Reinforcement Learning Methods with a PC Cluster System”, IAT2006：416-419
[3]宮崎和光，木村元，小林重信，”Profit Sharing に基づく強化学習の理論と応用”，人工知能学会誌，Vol.14, No5, pp.800-807, 1999