特集解説

状態汎化とマルチエージェント化による大規模システムの強化学習

正員 木村 元* 非会員 青木 圭*
非会員 小林 重信*

Reinforcement learning in large scale systems using state generalization and multi-agent techniques

Hajime Kimura*, Member, Kei Aoki*, Non-member, Shigenobu Kobayashi*, Non-member

This paper introduces several problems in reinforcement learning of industrial applications, and shows some techniques to overcome it. Reinforcement learning is known as on-line learning of an input-output mapping through a process of trial and error interactions with its uncertain environment, however, the trial and error will cause fatal damages in real applications. We introduce a planning method, based on reinforcement learning in the simulator. It can be seen as a stochastic approximation of dynamic programming in Markov decision processes. But in large problems, simple grid-tiling to quantize state space for tabular Q-learning is still infeasible. We introduce a generalization technique to approximate value functions in continuous state space, and a multiagent architecture to solve large scale problems. The efficiency of these techniques are shown through experiments in a sewage water-flow control system.

キーワード：強化学習，プランニング，汎化，関数近似，マルチエージェント
Keywords: Reinforcement learning, planning, generalization, function approximation, multi-agent

1. はじめに

本稿では強化学習を産業応用するにあたり突き当たる問題と，それに対処するための理論やノウハウを紹介する。

強化学習は，試行錯誤を通じて環境に適応していくことを特徴とする学習制御の枠組みの一つである[1]。強化学習は，不確実性の存在する環境下において，タスクの達成を反映した報酬という単純な信号を手がかりにして，それぞれの制御対象に適した複雑な制御規則を自動的に獲得していくことから，制御プログラミングの自動化・省力化への貢献が期待されている。しかし，強化学習の大きな特徴である「試行錯誤」による制御規則の探索は，特に産業応用においては制御対象に致命的な損害を与えかねない。そのため，産業応用において強化学習をオンラインで利用する場合には，パケットルーティング[2][3]やセルラー通信の無線バンド割当[4]，コールアドミッション制御[5][6]など，多少の試行錯誤で影響を受けない問題に適用するか，あるいは探索を行う範囲を制限して，制御規則を少しずつ改善していくといった限定的な使い方しかできない。

一方で，強化学習をプランニング (planning) のための探索エンジンとして利用する接近法の有用性が報告されている。これは実環境中で試行錯誤するのではなく，対象とする制御対象のシミュレータによって試行錯誤による強化学習を行い，得られた制御規則をプランニング結果として実環境へ適用するものである。状態遷移に不確実性を有する制御対象においてプランニングを行う場合，まず制御対象のダイナミクスをマルコフ決定過程 (Markov decision process: MDP) やセミマルコフ決定過程などの確率モデルによってモデル化し，これを動的計画法 (Dynamic programming) などの手法を用いて解くのが一般的である。しかし，産業応用を志向した大規模問題では，実データや機械の仕様をもとにシミュレータを作ることはできても，制御対象のダイナミクスを状態遷移マトリクスとして表現することが困難な場合が多い。例えば，状態量を特徴付ける変数が多次元連続値の場合，遷移マトリクス表現のために状態の量子化を行うと，状態空間の爆発によりメモリ容量的に実行不能に陥る。また，並列非同期な離散事象システムでは，システム全体としての状態遷移ダイナミクスを与えることは難しい。強化学習では，制御対象のダイナミクスに関して，こうしたモデルのような明示的な知識が不要なので，シミュレータさえあれば，これを用いて制御規則を得ることが可能である。シミュレータで強化学習を行うアプローチは，理論的には MDP などの確率モデルを動的計画法で解く手法にお

* 東京工業大学 大学院総合理工学研究科
〒 226-8502 横浜市緑区長津田 4259

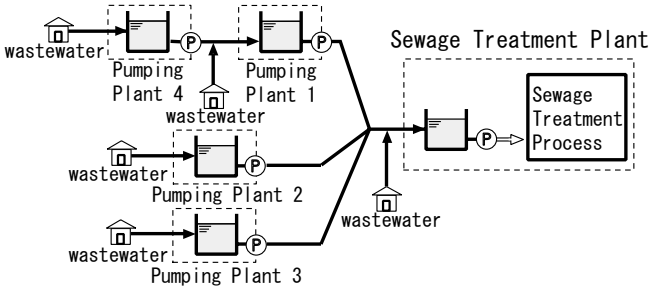

図 1　下水道送水系（５施設）

Fig. 1.　A sewage system composed of five pumping plants.

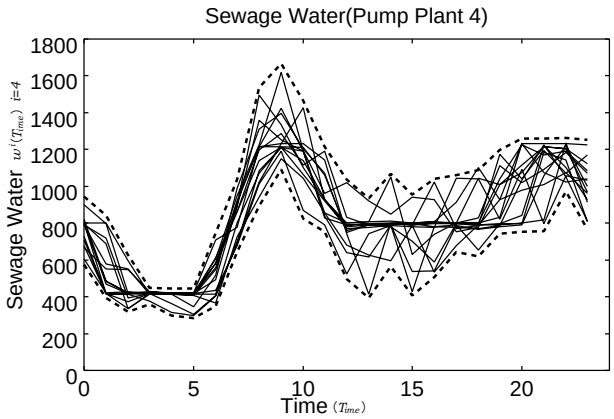

図 2　ポンプ場４の流入汚水量の時間系列データ（実線）。点線は変動幅の最大最小範囲を表す

Fig. 2.　The flow of the sewage water into the plant no.4. The solid lines denote instances, the broken lines show the variational range.

ける計算過程を，確率的サンプリングによる近似推定で置き換えたものと考えることができる。状態遷移分布に従ったサンプルをシミュレータから得て，それを平均することで計算を近似するものであり，状態遷移分布マトリクスを明示的に求める必要がないため，場合にもよるが必要なメモリー容量を $o(s^2)$ から $o(s)$ へ減らすことができる。ただし s は量子化した状態個数を表す。強化学習によるプランニングの応用例として，エレベータの制御規則獲得 [7] [8] [9] が多数報告され，生産システム管理問題 [10] [11] では，トヨタの KANBAN 方式や constant work-in-progress (CONWIP) 等のヒューリスティクスとの比較において優れた解を得たことが報告されている。

大規模プランニングにおいて，強化学習は有望な接近法ではあるが，大規模問題の状態空間を単純にグリッド分割によって量子化して，離散状態での単純な強化学習法（Q-learning など）をナイーブに適用するだけでは，状態や行動変数の次元の増加に伴って空間が爆発するため，限界がある。この問題への対処には，主に２通りの方法がある。一つは空間を汎化する方法で，状態や行動の評価値を少ないパラメータでなめらかに関数近似するものである。もう一つは制御対象を小さく分割してそれぞれの制御規則を得る分割統治法で，マルチエージェント系として学習するものである。

本稿では，下水道送水系計画問題のプランニング法として強化学習を用いた事例 [12] を取り上げ，シミュレータ環境での強化学習によるプランニングと，状態空間の汎化，マルチエージェントといったテクニックについて解説する。

2. 下水道系送水計画問題

〈2･1〉 問題の概要　下水道送水系は，各家庭や事業所から排出される汚水を集積し，処理場へ送る一連の施設群より構成される (図 1)。系の施設群は一つの処理場と複数のポンプ場からなり，それぞれが配管によって連結され，処理場をルートとした木構造になっている。各家庭より排出される汚水は，生活リズムに連動しており朝と夕方にピークを持つように確率的に変動して各ポンプ場に流入するが，下水処理場は生化学的な汚水処理を行うので，処理場への汚水流入量は一定であることが望ましい。各ポンプ場は，処理場での流量変動を吸収するバッファの役割を担うために貯水池が設けられ，ポンプで下流施設へ送水する際に流出量のコントロールを行う。制御問題として大雑把に定式化すると，意思決定を 1 時間毎に行う離散時間制御問題であり，状態量・操作量・制御目標は以下のとおりである。

- 観測できる状態量：各ポンプ場の貯水池水位，時刻（24 種類の離散値），および前回の意思決定で選択したポンプの送水量（これは運転コストがポンプの起動／停止の切替に依存するため）
- 操作量（行動）：各ポンプ場の送水量。ただし複数台ポンプの運転／停止しか指示できないので，各ポンプ場毎に数段階の離散値をとる。
- 制御目標：貯水池水位の上下限制約違反を起こさない範囲内で，ルートに配置された下水処理場への汚水流入量の平滑化，およびポンプの運転コスト最小化，

前述のとおり，系のさまざまな箇所へ流入してくる汚水は，時刻に依存して確率的に変動する。図 2 はポンプ場４へ流入する汚水量の時間変化を示す。各ポンプ場の貯水池にはその水位に運用上下限制約があり，それを超えることは汚水の流出や施設へのダメージにつながるため，その範囲内で運用しなければならない。

〈2･2〉 計画問題としての困難　現状の運用形態は，各ポンプの送水量についての運用計画を前日に作成し，それに従って当日の運用を行っている。運用計画は専門家が過去のデータと現在の状態などから各地区における汚水流入量の時間変化を予測し，これに基づいて決定論的な探索アルゴリズムを用いて時刻表のような計画が立案される。ところが汚水流入量は不確実性を伴うため，完全な予測は不可能であり，当日の運用では予測された状況から次第に誤差が生じてしまう。そこで各施設の現場において，制約違反を起こしそうになったら経験や勘で運用計画が補正さ

れているのが現状であり，運用計画で期待される処理量の平滑化は十分には行われていない。

このように状態遷移に不確実性を伴う場合には，時刻表のような運用計画を立てるのではなく，状態から行動への写像関数という形式の制御規則を与えるべきである。以降，この制御規則の獲得を考える。このような計画問題では，制御対象を MDP としてモデル化し，線形計画法や DP 等の最適化手法を用いるのが一般的である。しかし，問題が大規模になると，制御対象を状態遷移確率行列として表現すること自体がメモリ容量的に実行不能である。そこで，状態遷移確率行列を明示的に用意せずに，シミュレータをサンプラーとして確率的に近似計算を行う。これはシミュレータ環境において強化学習によって制御規則を獲得することと等価である。本接近法によって計算は可能になるが，依然として膨大な空間の扱いが障害となる。以下，強化学習において膨大な空間を扱うためのテクニックを紹介する。

3. 強化学習の適用

〈3•1〉 離散状態-行動の強化学習 MDP では意思決定の各時間ステップにおいて報酬信号が入ってくる。これは前章の問題では，各時間のポンプの運転コストや制約違反の有無を反映するものである。MDP では，制御規則（以後政策と呼ぶ）の評価規範として，長期的な報酬合計を考慮した割引報酬合計による評価が一般的である。強化学習の代表的手法である Q-learning[13] は，適切な条件下では割引報酬合計の期待値を最適化する政策の獲得が保障されている。Q-learning は，各状態-行動対に対し，そのルール実行直後から最適な政策を取り続けた場合の割引報酬合計期待値を Q 値として推定する。ある状態 s_t で行動 a_t を選択後，報酬 r_t を受け取って状態 s_{t+1} へ遷移した時，以下に従って Q 値を更新する：

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(r_t+\gamma\max_{a\in A}Q(s_{t+1},a)-Q(s_t,a_t)\right) \quad \cdots (1)$$

ただし $0<\alpha\leq 1$ は学習率，$0\leq\gamma<1$ は割引率，A は行動集合を表す。状態や行動空間を量子化して扱うと，上記の Q 関数は 2 次元配列のテーブル形式になり，実装は極めて簡単である。しかし，下水道系問題のように連続値状態変数パラメータが多次元になると，量子化される状態個数は指数的に増大する問題がある。

〈3•2〉 連続状態空間の汎化 量子化された状態-行動空間で Q 関数を正確に記述するには，巨大な記憶容量が必要なだけでなく，前節の学習方法では膨大な時間とデータが必要である。これを解決するため，これまで経験した状態空間から未経験の状態への汎化 (generalization) を行う。このような例題からの汎化は関数近似とも呼ばれ，強化学習に限らずニューラルネットワークなど幅広い分野で研究されており，強化学習または DP と関数近似法を組み合わせた計算法は neuro-dynamic programming[14] と呼ばれる。Q-learning の最適解への収束が保障される簡便な汎化方法として線形アーキテクチャ(linear architecture) がある。これは固定された基底関数の集合を用いて関数近似を行うもので，ラジアル基底やウエーブレット関数基底を用いたものなど様々だが，状態空間中の座標を基底関数によって特徴量ベクトルに変換し，このベクトルの線形重み付け和で関数を表現する点は同じである（図 3）。特徴量ベクトルの生成方法は問題に応じて設計者が与える必要があり，ベクトルのノルムが常に 1 となることが望ましい。図 4 は最も単純な特徴ベクトル生成方法の例を示す。2 次元空間を 1 枚のタイル（グリッド）によって分割し，各タイル要素に特徴量ベクトルの要素を割り当てる。状態入力があるタイル要素内にあるとき，該当する特徴量ベクトル要素の値は 1 に，それ以外の要素はゼロとする。Q 値を表現するときは，特徴ベクトル要素と同数のパラメータ w_i^a を用いて線形関数表現する：

$$Q(s,a)=\sum_{i=1}^{n} w_i^a x_i \quad \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

ただし n は特徴ベクトル要素の個数。離散行動が A 個のとき，パラメータ w_i^a は $n\times A$ 個必要である。線形アーキテクチャによる Q 値更新は，単純な勾配法に基づきパラメータ w_i^a を更新することで達成される：

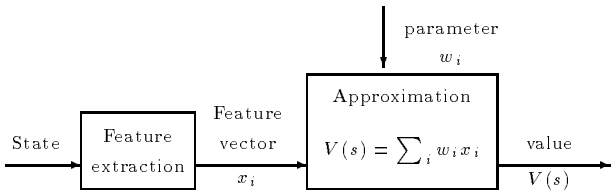

図 3 特徴量ベクトルへの変換を介した線形アーキテクチャによる汎化。

Fig. 3. A generalization of linear architecture through the feature vector.

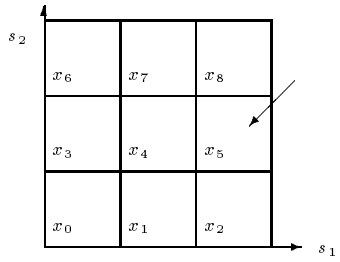

図 4 タイルコーディングによる特徴量ベクトル生成例。状態入力 (s_1,s_2) が矢印の位置のとき，特徴量ベクトル $(x_0,x_1,\cdots x_8)$ は，該当するタイル要素 $x_5=1$，それ以外の要素は全てゼロ。

Fig. 4. An example of tile coding to generate a feature vector. When the current state (s_1,s_2) is located at the arrow in the figure, the corresponding element x_5 equals 1, and the others are all zero in the feature vector $(x_0,x_1,\cdots x_8)$.

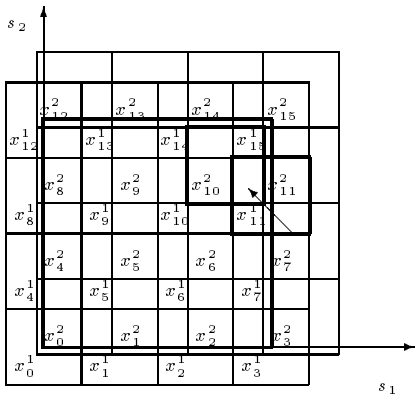

図 5　2 枚のタイルをずらせて重ねる CMAC による特徴量ベクトル生成。大きな太線の枠は状態の定義域，2 つの小さい太線枠は状態入力に対応するタイルを表す。状態入力が矢印の位置のとき，特徴量ベクトルは，該当するタイル要素 $x^1_{11} = 0.5$，$x^2_{10} = 0.5$，それ以外の全要素はゼロ。

Fig. 5. An example of CMAC, which generates a feature vector, composed of multiple overlapping gridtilings. The large rectangle with thick line shows the bound of the state space, two small boxes with thick lines are the tiles corresponding to the state input. When the current state is located at the arrow in the figure, the corresponding element x^1_{11} equals 0.5, x^2_{10} equals 0.5 and the others are all zero in the feature vector.

$$w_i^{a_t} \leftarrow w_i^{a_t} + \alpha \frac{\partial Q(s,a)}{\partial w_i^{a_t}} \left(r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right) \quad (3)$$

式 (2) より $\partial Q(s,a)/\partial w_i^{a_t} = x_i$ なので，

$$w_i^{a_t} \leftarrow w_i^{a_t} + \alpha \; x_i \left(r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right) \quad (4)$$

つまり線形アーキテクチャを用いた更新では，更新するパラメータ w_i^a に対応する特徴の要素の値 x_i だけを使って簡単に更新できる。図 4 のような単純なグリッド分割による線形アーキテクチャの更新式は離散的 Q-learning の更新式（式 (1)）と等価になる。

図 5 は CMAC[15] と呼ばれる汎化法による特徴ベクトル生成例を示す。これは図 4 のようなグリッドを k 枚（この場合 $k = 2$）用意し，ずらせて重ね合わせる。状態入力に対し，図のように複数のタイル要素が反応し，それに対応する特徴ベクトル要素の値が $1/k$，それ以外の要素はゼロとする。グリッドのずらせかたはランダムに行う。グリッド 1 枚における分割の粗さは，汎化の度合いを特徴付ける。つまり粗い分割メッシュを用いると，学習途中でサンプル数が少ない場合でも，未経験の状態では値を補間する傾向が強くなり，対象問題によっては学習高速化の効果が得られる。重ね合わせるタイル（グリッド）の枚数は，関数近似能力の高さを特徴付ける。この例では 2 次元空間なので

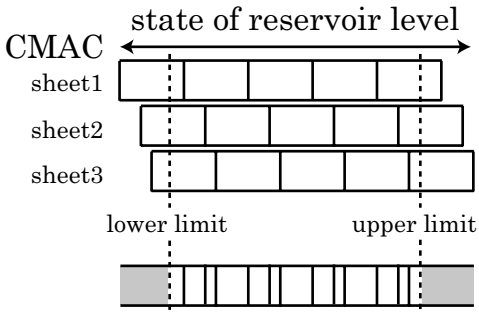

図 6　貯水池の水位を特徴ベクトルへ変換するための CMAC グリッドデザイン。等間隔グリッド 3 枚を基本に用いて，運用上下限位置で強制的に分割する。

Fig. 6. A gridtiling design to generate a feature vector from the water level. Here the tiles are based on uniform gridding, and they are divided at the upper and lower limit of the water level.

タイルは矩形だが，多次元になるとタイルも超矩形になる。CMAC も線形アーキテクチャなので Q 値の更新式は式 (4) と同じである。

下水道問題の強化学習でも CMAC を用いたが，予め分かっている問題の特徴に応じてタイルの構造に工夫を加えた。状態量として，貯水池の水位×時刻という 2 次元空間を扱うことを考える。貯水池の水位について補足すると，物理的上下限に達しないように運用上下限が設けられているので，水位がこれを超えると大きな負の報酬が与えられる。この運用上下限の位置は予め分かっており，その付近を境に Q 値が大きく変化することは明らかである。そこで，水位に関しては図 6 に示すように，運用上下限の位置でタイルを強制的に分割する。これにより，運用上下限に近い位置付近に小さなタイルが集まるので，微妙な制御が必要なこの領域の関数近似能力を高めることができる。逆に中間的な水位領域では大きなタイルだけでカバーしているので，補間能力が高く，メモリ容量の低減と学習高速化が期待できる。

また，時刻については 23 時から 0 時へ戻るような環状ループする特徴を持つ状態変数であるため，図 4 や図 5 のような定義域の端がない。そこで，図 7 に示すように特徴ベクトルのタイル配置もループさせた。

〈3•3〉 マルチエージェント化　前節で紹介した汎化を用いると，少ないパラメータを使って Q 値を近似表現し，少ない経験から補間してくれるという意味で，ある程度までの規模の問題を扱うことが可能になる。しかし，CMAC のタイル分割の例のように，状態変数の次元数が上がると，タイル分割数および特徴ベクトル個数が指数的に増大する。本稿の下水道問題においても，水位の変数だけで 5 次元なので，系全体の状態-行動を Q 値で評価することはメモリ容量的に実行不能である。ここでは解決策として，マルチエージェント系で問題を分割する。

表 1　下水道送水系問題における状態表現，報酬設計

Table 1.　Features of the states and rewards in the sewage plants.

下水道広域送水系（5施設）				
施設	状態次元数	行動数	状態観測	報酬
処理場	4	8	時刻（24），貯水池水位（10），前行動（8），上流合計揚水量（10）	揚水量変更量，貯水池水位違反
ポンプ場1	3	8	時刻（24），貯水池水位（10），上流合計揚水量（10）	処理場流入変化量，貯水池水位違反
ポンプ場2，3	2	11,8	時刻（24），貯水池水位（10）	処理場流入変化量，貯水池水位違反
ポンプ場4	2	11	時刻（24），貯水池水位（10）	ポンプ場1流入変化量，貯水池水位違反

かっこ内は空間量子化によりテーブル形式で学習する場合の状態分割数

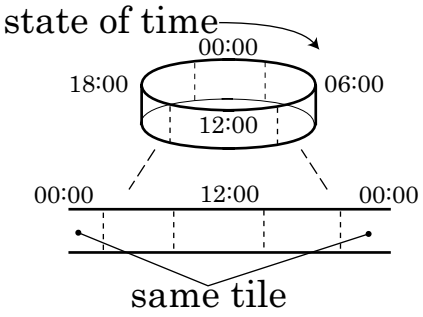

図 7　時刻を特徴ベクトルへ変換するための CMAC グリッドデザイン。時刻は，23 時の次は 0 時へ戻る環状ループの特徴を持つ状態変数であり，定義域の端がない。そこで，特徴ベクトルのタイル配置もループさせる。

Fig. 7.　A gridtiling design to generate a feature vector from the hour. It is a cyclical state variable such as the next value of 23 is 0, and there is no edge of the space. Therefore, we put the tiles cyclically.

Q-learning は一般にマルチエージェント系での学習の最適性を保障していない。しかし，あるエージェントから見て他のエージェントの政策が定常あるいは十分に学習がゆっくり行われ，かつ状態変数の観測が十分可能ならば，他のエージェントを含めた環境は MDP に近似して差し支えないと考えられる。よって大規模問題を適切に分割し，それぞれに Q-larning を用いる接近法にも合理性があり，適用例が報告されている [16]。

本稿の下水道問題は，施設がツリー状に結合されており，施設ごとにポンプ揚水量を決められるなど施設単位での分割がしやすい特徴がある。水の流れは配管を通して行われ，流れは一方通行なので，影響は下流施設にのみ限定される。ネットワーク的に離れている施設への影響は少ないと考えられる。そこで各施設をエージェントとしたマルチエージェント系を考える。各施設は自分の貯水池の水位だけでなく，上流の施設の影響も受けることにも対応するため，状態観測として一つだけ上流における全施設の揚水量合計も観測する。報酬としては，行動結果として自分の貯水池水位の制約違反の有無だけでなく，下流の流量変化も反映されて与えられる。表 1 に状態や行動，報酬設定についての概要を示すが，詳細な設定は参考文献 [12] を参照されたい。

4. 実験結果

ここでは主に前章で示した接近法のうち，状態汎化の効果を示す。実験では CMAC のタイル数を 4 もしくは 6 に設定し，(a) テーブル表現の場合とメモリ数が同じ場合，(b) 各タイルの分割数を減らしメモリ保持数を約 55.8 %圧縮した場合の 2 通りの設定で行った。図 8 に学習曲線を示す。グラフの横軸は学習ステップ数，縦軸は 1 日あたりのポンプの切替回数を表す。ポンプの切替は流量変化や運転コストの増大につながるので，これが少ないほど良い。このグラフでは，制約違反が反映されていないが，制約違反に伴う負の報酬はポンプの切替コストと比べると重みが 5 倍になっており，最終的に得られた政策ではどれも制約違反をほとんど起こさなかったので，政策の性能の差は本グラフにのみ反映される。

学習率などのパラメータ設定やマルチエージェントとしての問題設定は同一なので，性能の差は状態汎化の効果だけを反映している。グラフより明らかに従来の状態量子化によるテーブル表現に比べて汎化により格段に学習速度は向上し，最終的に得られた制御規則の性能も良くなっている。特に学習序盤は，パラメータ数が少なく，汎化能力が大きい (b) の性能向上が目立つ。これは本問題の汎化において，水位の中間付近ではあまり高い関数近似能力が必要無いことを示している。

図 8 のグラフでは，シミュレータ環境で 5×10^7 ステップの学習を行っている。PentiumIII-800MHz 程度の計算機上で SunOS 上の Java プログラムを用いてシミュレートすると，約 3 時間を要する。CMAC を用いた手法では，30 分程度で解を得ている。

5. おわりに

本稿では，シミュレータと強化学習を併用して制御規則を獲得していくプランニング手法を紹介した。大規模問題へ強化学習を適用するため，状態汎化を行う方法と，問題を分割してマルチエージェントとして対処する方法を示した。特に状態汎化は，対象問題の特徴を反映して，粗い汎化で良い箇所と細かい近似能力を必要とする箇所に対応した適切な関数近似手法を設定することにより，格段の性能向上が可能であることを示した。本稿で示したシミュレータによる実験結果は，参考文献 [12] 中で記述し切れなかったものをここで紹介したもので，複数の下水道制御の専門家

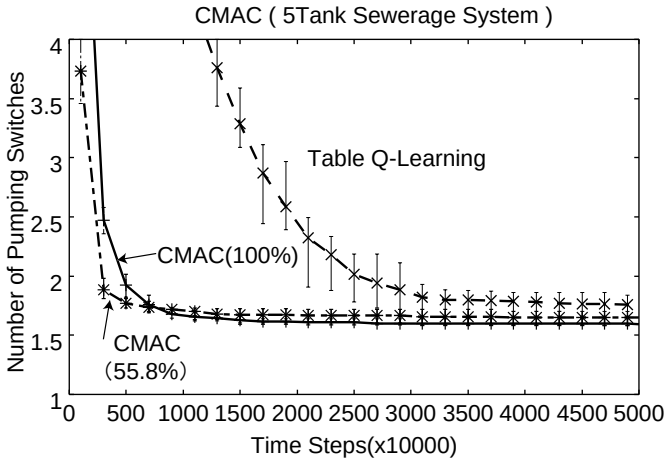

図 8　CMAC を適用した場合とテーブル形式のままの場合の学習比較。線は 10 試行平均，誤差棒は最良最悪ケース。
(a) テーブル形式と同じメモリ容量の CMAC,
(b) テーブル形式の 55.8 %のメモリの CMAC

Fig. 8.　The learning performance comparing the tabular Q-learning and the CMAC Q-learning. The lines are the average of 10 trials, and the bars are the best and the worst cases. (a) CMAC using the same amount of memory as the tabular Q-learning, (b) CMAC using 55.8% amount of memory as the tabular Q-learning.

より実運用の立場から十分評価できるとのコメントを得ている。強化学習を産業応用する場合，本稿で取り上げた課題以外にも連続行動空間の扱いや不完全観測の扱い [1]，少ない試行回数からの学習など様々なテーマが考えられ，活発な研究が行われている。このような強化学習の理論的な発展とともに，近年の計算機の目覚しい性能向上の影響も加わり，シミュレータでの学習によるプランニング手法は今後もますます発展していくと考えられる。

文　献

(1) H. Kimura, K. Miyazaki & S. Kobayashi: A guideline for designing reinforcement learning systems, Journal of the society of instrument and control engineers, vol.38, no.10, pp.618–623 (1999) in Japanese.

(2) Boyan, J. & Littman, M.: Packet routing in dynamically changing networks: A reinforcement learning approach, *Advances in neural information processing systems 6*, pp. 671–678 (1993).

(3) Kumar, S. & Miikkulainen, R.: Confidence based dual reinforcement Q-routing: An adaptive online network routing algorithm, *Proc. of 16th international joint conference on artificial intelligence*, pp.758–763 (1999).

(4) Singh, S., & Bertsekas, D.: Reinforcement learning for dynamic channel allocation in cellular telephone systems, *Advances in neural information processing systems 9*, pp.974–980 (1997).

(5) Marbach, P, Mihatsch, O. & Schulte, M. & Tsitsiklis, J. N.: Reinforcement learning for call admission control and routing in integrated service networks, *Advances in neural information processing systems 10*, pp.922–928 (1998).

(6) Brown, T., Tong, H. & Singh, S.: Optimizing admission control while ensuring quality of service in multimedia networks via reinforcement learning, *Advances in Neural Information Processing Systems 11*, pp.982–988 (1999).

(7) Crites, R. & Barto, A. : Improving elevator performance using reinforcement learning, Advances in neural information processing systems 8, pp.1017–1023 (1995).

(8) Pepyne, D. L. & Looze, D. & Cassandras, C. & Djaferis, T. : Application of Q-learning to elevator dispatching, 13th Triennial world congress, pp.317–322 (1996).

(9) S. Hikita & S. Abe: Elevator group control systems using AI technologies, Journal of the Japanese society for artificial intelligence, vol.17, no.1, pp.57–62 (2002) in Japanese.

(10) Mahadevan, S. & Marchalleck, N., Das, T. & Gosavi, A.: Self-improving factory simulation using continuous-time average-reward reinforcement learning, *Proc. of 14th international conference on machine learning*, pp.202–210 (1997).

(11) Wang, G. & Mahadevan, S.: Hierarchical optimization of policy-coupled semi-Markov decision processes, *Proc. of 16th international conference on machine learning*, pp.464–473 (1999).

(12) K. Aoki, H. kimura, A. Nagaiwa & S. Kobayashi: Adaptive Control of sewage systems using distributed reinforcement learning, Trans. IEE of Japan, vol.123-D, no.4, pp.462–469 (2003) in Japanese.

(13) Watkins, C. & Dayan, P.: Technical note: Q-learning, *Machine Learning 8*, pp.279–292 (1992).

(14) Bertsekas, D. & Tsitsiklis, J.: *Neuro-dynamic programming*, Athena Scientific (1996).

(15) Sutton, R. & Barto, A.: Reinforcement learning: An Introduction, *A Bradford book*, The MIT press (1998).

(16) S. Mikami: Reinforcement learning for multi-agent systems, Journal of Japanese society for artificial intelligence, vol.12, no.6, pp.845–849 (1997) in Japanese.

木 村　元（正員） 1997 年東京工業大学大学院知能科学専攻博士課程修了，同年 4 月日本学術振興会 PD 研究員，1998 年 4 月，東京工業大学大学院総合理工学研究科助手，現在に至る。人工知能，特に強化学習に関する研究に従事。

青 木　圭（非会員） 2002 年東京工業大学大学院知能システム科学専攻修士課程修了。同年 4 月博士課程在籍中。主に強化学習に関する研究に従事。

小 林 重 信（非会員） 1974 年東京工業大学大学院博士課程経営工学専攻修了。同年 4 月，同大学工学部制御工学科助手．1981 年 8 月，同大学大学院総合理工学研究科助教授。1990 年 8 月，教授，現在に至る。問題解決と推論制御，知識獲得と学習などの研究に従事。