3F4-07

強化学習を用いたSoar情報収集機構の試作

Impremention of The Elaboration Unit based on Reinforcement Learning in Soar Architecture

保知 良暢[*1] 大囿 忠親[*2] 新谷 虎松[*2]
Yoshinobu Bochi Tadachika Ozono Toramatsu Shintani

[*1]名古屋工業大学大学院工学研究科
Graduate School of Engineering, Nagoya Institute of Technology

[*2]名古屋工業大学知能情報システム学科
Department of Intelligence and Computer Science, Nagoya Institute of Technology

This paper proposes a new elaboration unit in Soar. Soar is a problem solving architecture for a system that is to be capable of general intelligence. It is difficult to apply the architecture to a real world problem because of unknown state transitions and existence of unknown objects. The inductive learning and planning mechanism is significant in the problem. We provide a new elaboration unit based on Reinforcement Learning. This unit executes a Reinforcement Learning cycle to get an optimal policy in an unknown environment.

1. はじめに

不完全または記述困難な知識が存在する状況下においても，ユーザの負担を最大限軽減し，帰納的に知識を獲得しつつ問題を解決するアーキテクチャに関する研究は多く行われており，実アプリケーションにおいても応用範囲は広いと考えられる．

強化学習 [Sutton 98] は試行錯誤を繰り返し望ましい状態に対して報酬という数値化された信号を受け取ることで学習する．世界のモデルが未知である問題で高い性能を示すため，ヒューリスティックスが不十分な場合でも帰納的に学習できる問題解決器になると考えられる．しかし強化学習の実行前に予め知識を与えることは困難である．また学習後の価値関数は同一の状態を知覚しなければ利用できず，再利用性が低いと考えられる．価値関数から有益なルールを抽出し，将来の問題解決に利用する方法が求められる．本論文では，強化学習結果からプロダクションルールを生成する手法を提案する．

本論文では強化学習よりプロダクションルールを生成し，Soarによりプランニングする手法を提案する．Soar プロジェクトは知能一般のアーキテクチャの構築を目指して始められ，様々な研究がなされてきた [Laird 87][堀 94]．汎用的問題解決器として提案された Soar は現在も多くのアプリケーションで利用されている [Lent 01]．Soar の決定サイクルは情報収集段階と決定段階からなる．探索が行き詰まったらインパスに入り，新たな問題空間とサブゴールを設定し，問題解決を行う．Soarの処理系はプロダクションシステムで構築されており，プロダクションルールを利用し問題解決を図る．Soar では問題空間を予め記号で定義するため，オペレータや世界に関するモデルが既知である問題に適用でき，抽象的なレベルでのプランニングが可能である．Soar は記号で閉じた世界において高速なプランニングが可能だが，帰納的な学習機構を持たないため，環境に対して規則性を発見し適切な行動を学習することはできない．Soar は主にヒューリスティック関数を用いプロダクションの選好性を決定するが，予めヒューリティック関数を十分に与えられない場合はプランニングができないという欠点を持つ．

本論文では，Soar のアーキテクチャを改良し，強化学習を備えた新たな問題解決器を提案する．本問題解決器はインパスに入ると自動的に強化学習を呼び出し，強化学習の結果からプロダクションルールを生成する．生成されたプロダクションルールは将来の問題解決に利用可能となる．本論文では本システムを倉庫番において実装し評価する．

連絡先: 保知 良暢, 名古屋工業大学知能情報システム学科新谷研究室, 〒 466-8555 名古屋市昭和区御器所町, TEL:052-733-6550, FAX:052-735-5584, bochi@ics.nitech.ac.jp

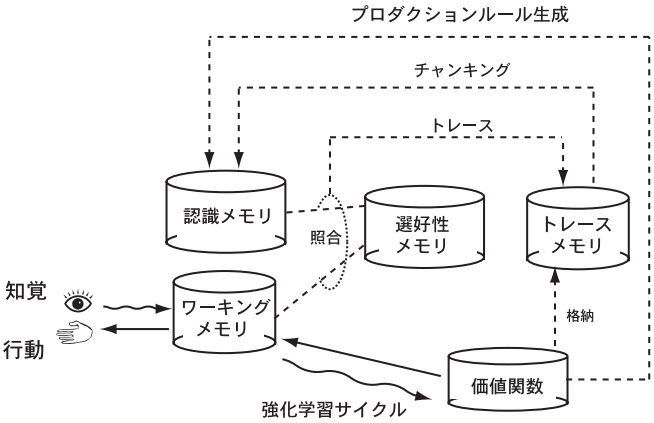

図 1: 本システムのメモリ構造

2. 強化学習によりインパス解消を行う Soar アーキテクチャ

2.1 本システムのメモリ構造

図 1 に提案するシステムのメモリの構造を示す．ワーキングメモリには現在のゴール，状態，オペレータなどの一時的なオブジェクトが格納される．オブジェクトは同定子と付加情報からなる．同定子はオブジェクトのユニークな名前であり，最初に作成される時にシステムが与える．付加情報は属性と値のペアのリストである．属性は ^ で与える．例えば，`(B2 ^type block)(B2 ^color red)` は赤い積み木を表現する．本システムの特徴は，プランニングに行き詰まったならば自動的に強化学習のサイクルを呼び出すことである．強化学習を行う場合，ワーキングメモリには学習する際の初期状態，終端状態の集合，報酬関数に関する情報が付加される．

永続的な知識は認識メモリに格納される．認識メモリの知識

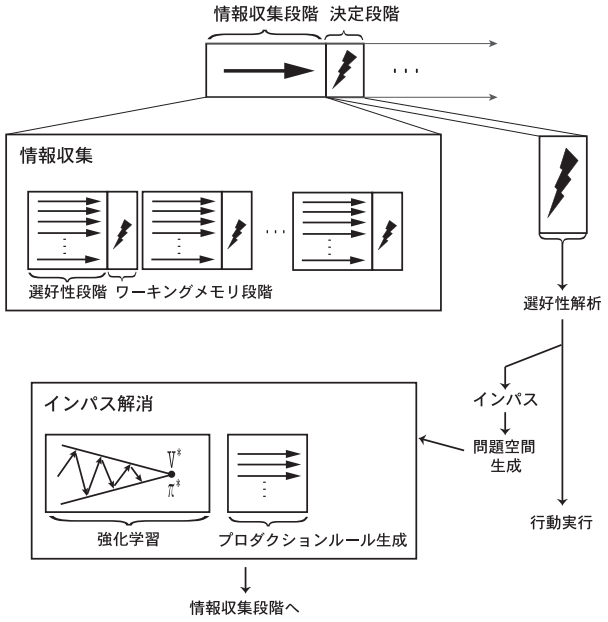

図 2: 強化学習スレッドを持つ決定サイクル

はプロダクションルールで表現される．プロダクションは条件部と行動部から構成される．条件部はワーキングメモリの内容と照合がとられ，照合が成功したプロダクションは発火する．選好性メモリには照合によって得られたオペレータに関する選好性が格納される．

トレースメモリには過去に経験した本システムの決定サイクルおよび価値関数を格納する．プランニングや強化学習において過去と同じサイクルを繰り返して無限ループになることを防ぐ．またトレースメモリは Soar のチャンキング機能や強化学習より新たなプロダクションルールを生成することに用いる．

価値関数は，状態，外界に対して可能な行動およびその価値という組の集合からなる．各状態と行動に対する価値は強化学習のサイクルによって更新される．学習が収束すると得られた結果からプロダクションルールを生成し認識メモリに格納する．強化学習により新たな知識となるプロダクションルールの生成が可能となる．

2.2 本システムの決定サイクル

本システムで実行する決定サイクルを図 2 で示す．決定サイクルは情報収集段階と決定段階からなる．

環境から知覚を得ると情報収集段階に入る．情報収集段階は選好性段階とワーキングメモリ段階からなる．選好性段階では認識メモリ内のうち発火可能な全てのプロダクションを発火させ，選好性を生成する．ワーキングメモリ段階では，選好性を全て集め，これに基づいてワーキングメモリを変更し，選好性段階をはじめる．新たなプロダクションインスタンスが生成されないという無活動状態になると，情報収集段階が終了する．

無活動状態に達すると決定段階に移る．決定段階では，先の段階で得られた選好性を解析する．解析の結果現状態に対してとるべき行動を決定し，外界に対して実行する．選好性解析の結果，十分な選好性が得られなかった場合はインパスに入る．インパスを解消するためには，問題空間を定義し，強化学習による情報収集サイクルを実行する．ここで問題空間とは，学習における初期状態，終端状態の集合，報酬関数を表す．問題空間の定義方法は 2.5 節で述べる．問題空間が定義されると価値

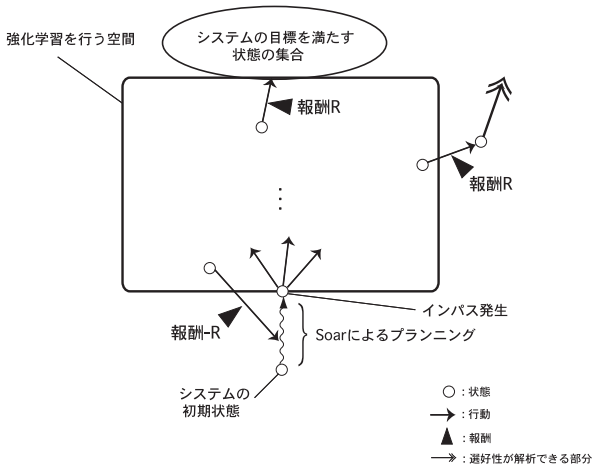

図 3: 強化学習を行う問題空間

関数を生成し，強化学習により各状態と行動に対する価値を更新する．学習が収束すると，得られた結果よりプロダクションルールを生成し，次のステップでの決定サイクルを実行する．生成されたルールにより，次のステップでの決定サイクルでは選好性が十分に解析され，外界に対して行動を実行できる．

2.3 選好性解析

選好性はワーキングメモリ中の付加情報の値にどれだけの価値があるかを示す．本システムでは，Soar と同様で 5 つのカテゴリーの選好性，(1) 受理可否 (受理可能, 却下)，(2) 必要性 (必須, 禁止)，(3) 優劣 (ベスト, ベター, 中立, ワース, ワースト)，(4) 排他性または両立および (5) 実行中断 (再考) がある．

インパスについても Soar と同様で 4 種類，(1) タイインパス，(2) 競合インパス，(3) 制約違反インパスおよび (4) 無変化インパスがある．インパスが解消されるのは最も良いと判断される行動が唯一決まった場合である．

2.4 本システムの入出力

入出力は，決定サイクルの決定サイクルの前後および強化学習の実行サイクルで行われる．入力は知覚器官から得られた情報がワーキングメモリに入る．出力も同様にワーキングメモリ内で表現される．行動がワーキングメモリに入っている場合，アクチュエータがこれを解析し実際に実行する．

決定サイクルの前後では，選好性段階の前あるいは無活動状態の前に入力が行われ，決定段階にて出力が行われる．決定サイクルの前後で行われる入出力は，本システムが行ったプランニングの結果である．本システムは強化学習でのサイクルにおいても入出力を行う．ここではプランニングをする為の探索がなされる．強化学習の情報収集サイクル終了後には，強化学習を実行する前に入力された状況に戻り，決定サイクルにおける出力を実行する．

2.5 強化学習を用いた情報収集

本システムでは強化学習をインパスを解消する手段として用いる．インパスに入ったならば，本システムは学習すべき問題空間，すなわち，初期状態，終端状態の集合および報酬関数を定義する．

初期状態，終端状態および報酬関数の定義方法についての概要を図 3 で示す．インパスに入った時，その知覚状態を初期状態として強化学習を実行する．図 3 の太線枠内が強化学習を行う問題空間を表す．太線枠外に伸びる矢印は，終端状態へ至

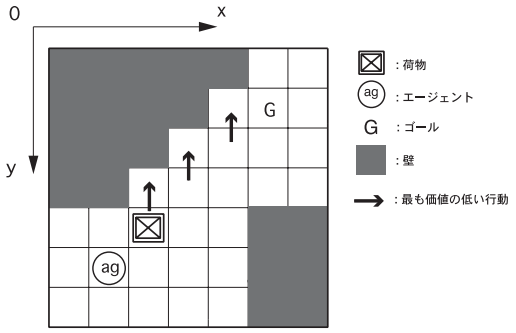

図 4: 倉庫番の例 1

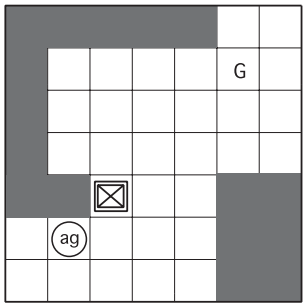

図 5: 倉庫番の例 2

る行動を表す．終端状態は，システムの目標を満たす状態，認識メモリ内のプロダクションルールで選好性が解析できる状態および現在までの決定サイクルで経験した状態からなる．システムの目標を満たす状態または認識メモリ内のプロダクションルールで選好性が解析できる状態に遷移した時報酬 $R(R > 0)$ が与えられる．現在までの決定サイクルで経験した状態に遷移した時報酬 $-R$ が与えられる．以上の設定より，強化学習によって正の報酬までの最適パスを探索する．強化学習は定義した問題空間において収束するまで行う．

2.6 強化学習結果からのルール生成機構

本システム特有の機能として，強化学習結果からプロダクションルールを生成することが挙げられる．強化学習が収束すると，各状態における行動の価値に優劣の差が得られ，選好性解析に用いることができる．前節で定義した問題空間内の全状態と全行動に関するプロダクションルールを認識メモリに追加した場合，ルール数が膨大になる．ルール数が多くなれば知覚した状態に対してどのプロダクションルールが発火しているかを調べる計算コストが大きくなる．また，強化学習で得られた結果は将来同一の状態を知覚しなければ発火しないので再利用性が低い．強化学習の結果から，未知の状態に対しても適用可能なルールを抽出し認識メモリに追加するメカニズムが望まれる．本論文では C4.5[Quinlan 93] による決定木を用いて汎化したプロダクションルールを生成する方法を提案する．

強化学習で用いる価値関数を Q と表し，以下で定義される．

$$Q : S \times A \rightarrow \Re$$

ここで S は状態の集合，A はとりうる行動の集合を表す．S と A はそれぞれワーキングメモリ内の知覚した情報と外界に対して実行する行動を表す．強化学習のサイクルにより，各状態と行動に対して実数の価値が得られる．ある状態 $s(\in S)$ に対して行動 $a(\in A)$ にベストの選好性を付加するプロダクションルールの生成を例に挙げ説明する．行動 $a(\in A)$ の価値が最も高くなっている状態の集合を $S^{a>}$ と表し，以下で定義する．

$$S^{a>} = \{s | \underset{s}{\operatorname{argmax}}\, Q(s,a), s \in S, a \in A\}$$

状態集合 $S^{a>}$ は，行動 a に対しベストと選好性が付加できる事例集合と考えることができる．また，1 つの状態は複数のオブジェクトの集合で表現される．本論文では，オブジェクト内に，行動 a をベストとする条件が含まれていると仮定し，C4.5 を用いて $S^{a>}$ の汎化を行う．a がワーストとなっている状態集合 $S^{a<}$，2 つの行動 a と a' を比較した時ベターとなっている状態集合 $S^{a>a'}$ においても，価値の大小を比較することで定義できる．各状態集合において汎化した結果をプロダクションルールに変換し，認識メモリに追加する．

3. 倉庫番における実行例

提案したシステムの有効性を確認するために倉庫番 [Junghanns 98] を例に示す．倉庫番は図 4 に示すような 2 次元上の世界において，横に x 軸，縦に y 軸をとる．エージェントは各マスにおいて荷物，エージェント，ゴール，壁または何もないことを知覚し，行動 up，$down$，$right$ および $left$ の行動をとる．エージェントの目標は全ての荷物を任意のゴールまで押すことである．同一マスに複数の物体が存在することはできない．図 4 と図 5 の例では，簡単のため荷物とゴールは 1 つずつ存在する．倉庫番においてプランニング上困難な点は，荷物がゴール位置以外で押すことができなくなってしまった場合，それ以降はデッドロックとなり問題を解くことができないことである．倉庫番においてプランニングに有効なルールまたはデッドロックに関するルールを予め書くことが困難であるため，試行錯誤により獲得することが望まれる．本論文では，多くの倉庫番の盤面を経験する程有効なプロダクションルールが生成されることを例により示す．本システムをまず図 4 に示す初期配置において実行し新たなプロダクションルールを生成する．続いて図 5 に示す初期配置で実行することで，プロダクションルールが汎化されていくことを示す．

3.1 図 4 での実行例

図 4 に示す初期位置において，本システムを実行する．認識メモリに何もプロダクションルールが無い場合，本システムではインパスとなり，強化学習を実行する．初期状態は図 4，荷物を G まで運ぶと報酬 1 とした．また，強化学習のステップが 100 を超えた場合初期位置に戻した．学習の収束結果を見ると，図 4 内に示すように壁の方向へ荷物を押し上げる行動の価値は低くなった．$S^{up<}$ は，C4.5 により汎化することで座標位置 $(2,4),(3,3)$ および $(4,2)$ において行動 up の選好性はワーストと与えられる．他の行動に対しても，座標位置と行動に対して選好性を付加するプロダクションルールが生成される．

3.2 図 5 での実行例

図 4 での強化学習後の価値関数はトレースメモリに保存されるものとする．図 4 に続き，図 5 において本システムを実行する．図 5 の状態ではエージェントは上へ移動できないため，強化学習を実行する．図 6 に示す方向に荷物を押す行動は，認識メモリ内のプロダクションルールで選好性が解析できるため，終端状態と見なせる．

図 4 と図 5 の学習結果を C4.5 で汎化する．図 4 で得たプロダクションルールは座標位置のみが前件に現れるルールであっ

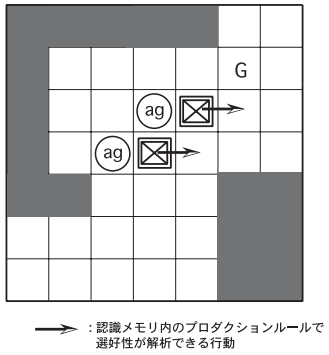

図 6: 終端状態の例

たが，図 4 と図 5 で学習することにより座標位置と壁の位置が前件に現れるルールが得られた．図 7 に行動 *up* がワーストとなるプロダクションルールの例を挙げる．壁，エージェントおよび荷物が図 7 左上に示す位置関係にあるとき行動 *up* が最も価値が低くなった．また，これを C4.5 により学習した結果とプロダクションルールとして書いた結果は図 7 右上と右下となった．図 7 のプロダクションルール内の < はワーストの選好性を表す．このプロダクションルールよりデッドロックとなる規則が得られている．

2 つの例で示したように，本システムは強化学習を自動的に呼び出し学習することができた．強化学習を実行する問題空間を本論文で示すように定めると，効率良く学習とプランニングを用いることができる．また，得られたプロダクションルールは図 4 および図 5 どちらにも利用できるルールとなった．

4. おわりに

本論文では，強化学習を備えた汎用問題解決器を提案した．Soar は帰納的な学習機構を持っていないが，本システムは強化学習を行い，学習結果からプロダクションルールを生成する点が大きく異なる．強化学習には問題空間が大規模になると状態空間の爆発が起こるという学習上の困難性がある．本システムでは，認識メモリ内のプロダクションルールが発火する状態を強化学習の終端状態とし，問題空間が大きくなることを防いだ．強化学習は Soar に対して教示する働きを持つ．Soar は自身の知識により抽象レベルでの高速なプランニングが可能となる．

本システムにおいて強化学習の結果よりルールを生成する目的は，チャンキングのような単なる高速実行だけではない点が従来の Soar と大きく異なる．試行錯誤から得られた情報を，記憶容量の減少と規則性の発見を目的に知識コンパイルをチャンキングにより行う．本システムでは強化学習の結果を C4.5 により汎化する方法を提案した．

本論文において倉庫番により例を示した．2 つの盤面で利用可能なプロダクションルールは生成できる．しかし，更に新たな盤面を解く場合，それまでで得たルールがどの程度再利用可かどうかは考察が不十分である．今後より多くの盤面において実行し，考察する予定である．未知の盤面においても利用可能なルールを得るには，ユーザが背景知識を与える必要がある．また，本システムで生成したプロダクションルールに誤りがある場合の修復方法を考慮する必要がある．本システムでは C4.5 を用いているが，誤ったルールを生成することは多い．どのプロダクションルールが誤っているかを判断するため

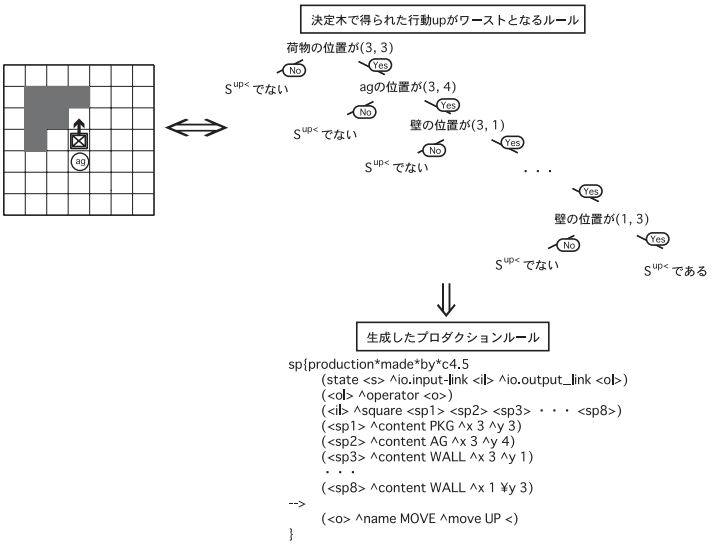

図 7: 行動 *up* がワーストであるルールの例

にの案として，トレースメモリに過去に適用したプロダクションルールとその効果を記憶し，目標状態に至らなかった場合にその選好性を下げるという方法が考えられる．

倉庫番はエージェントの行動に対して次状態は一意に決定する．しかし実環境を考慮した場合確率的に遷移することが考えられる．本システムでは強化学習を用いているため，実環境にも適用可能ではないかと考えている．

今後の課題としては，誤ったプロダクションルールを生成した場合の修復方法の提案と，より汎用的なプロダクションルール生成機構の開発が挙げられる．

参考文献

[Junghanns 98] Junghanns, A. and Schaeffer, J.: Sokoban: Evaluating Standard Single-Agent Search Techniques in the Presence of Deadlock, *In R. Mercer and E. Neufeld, editors, Advances in Artificial Intelligence*, pp. 1–15 (1998).

[Laird 87] Laird, J. E., Newell, A., and Rosenbloom, P. S.: Soar: An architecture for general intelligence, *Artificial Intelligence*, Vol. 33, pp. 1–64 (1987).

[Lent 01] Lent, M. V. and Laird, J. E.: Learning Procedual Knoledge through Observation, in *Proceeding of the International Confrence on Knoledge Capture*, pp. 179–186 (2001).

[Quinlan 93] Quinlan, J. R.: C4.5 : Programs for machine learning, *Morgan Kaufmann* (1993).

[Sutton 98] Sutton, R. S. and Barto, A. G.: Reinforcement Learning—An Introduction—, The MIT Press (1998).

[堀 94] 堀, 池田：小特集「Soar プロジェクト」にあたって, 人工知能学会誌, Vol. 9, No. 4, pp. 478–504 (1994).