Vol.2016-ICS-183 No.11
2016/3/16

経験から行動方策の知識表現を自己獲得・利用するエージェントアーキテクチャ

森田 信吾[1,a)]　加藤 昇平[1,b)]

概要：実世界における一般的な問題を解決するための知的な人工知能の設計を目的として，様々な研究がなされてきた．しかし，人工知能が高性能化されるにともないその作成のためのコストは増加すると考える．そのため数ある分野あるいは問題毎にそれに特化した人工知能を開発することは開発者に大きな負担がかかると考えられる．そこで本研究では，目標ベースの問題解決に着目した一般問題解決エージェントアーキテクチャを提案する．提案したアーキテクチャでシミュレーション実験を行うことでその有効性を示す．

1. はじめに

実世界における一般的な問題を解決するための知的な人工知能の設計を目的として，様々な研究がなされてきた [1], [2]．しかし，人工知能が高性能化されるにともないその作成のためのコストは増加すると考える．そのため数ある分野あるいは問題毎にそれに特化した人工知能を開発することは開発者に大きな負担がかかると考えられる．また，これらの人工知能の性能には開発者の技量が大きく関係し，コアとなる部分を作成できるような特定の開発者に開発負荷が集中することが考えられる．これらの問題を解決するためには，開発者が人工知能に対してその問題にあった作りこみをするのではなく人工知能自身が，与えられた問題およびその問題における達成目標から知識を獲得し，それを用いて効率的に問題を解決するようなシステムの実現が望まれると考える．
人工知能の実現においてはデータに対して統計的な処理を用いるアプローチ，あるいは強化学習などのように経験に基づく学習を繰り返すようなアプローチなどさまざまなものが存在する [3], [4]．本研究ではその中でも目標ベースの問題解決に着目する．これは，人工知能において問題を解決することは目標を達成するための一連のアクションの系列を発見することと同義であるという考え方である．この考えを人工知能の問題解決に用いた手法として手段目標分析が存在する．この手段目標分析では，システムは環境から自身あるいは外部環境の現在状態を受け取る機能と自身の状態あるいは外部環境を変化させるようなアクションを行う機能，加えてそれらの情報を格納するためのメモリを持つ．システムは現在状態と目標状態を近づけるようなアクションを選択していき，目標状態を達成する．

[1] 名古屋工業大学
Nagoya Institute of Technology, Dept. of Computer Science and Engineering, Graduate School of Engineering, Gokiso-cho, Showa-ku, Nagoya-si 466–8555, Japan
[a)] morita@katolab.nitech.ac.jp
[b)] shohey@katolab.nitech.ac.jp

1.0.1 先行研究

先行研究 [5] では，ある特定の問題だけに特化するのではなく，一般的な問題を解決することを目標として，手段目標分析による自動計画をベースとし，進化計算手法の一つである GeneticNetworkProgramming[6]（以下，GNP）を問題解決メソッドに用いたエージェントアーキテクチャを提案した．また，そのアーキテクチャを用いて迷路環境でのシミュレーション実験を行い，アーキテクチャの性能を検証した．
一方，先行研究アーキテクチャにおいては以下のような問題が考察された．

(1) GNP に関するパラメータの事前設定が必須である．

(2) 解そのものをグラフとして保持するため，得られた解のグラフの中に確率的に変化する要素を含む場合再現性がなく，次回試行などで解を表現したグラフとして不適切と判断され破棄される可能性がある．そのため毎回その部分に関する解を新たに 1 から作りなおさなければいけない可能性がある．

(3) ゴール分割に関して，連続値を状態として扱う場合は初期値から目標値までの連続的な状態の変化をサブゴールとしてしまい，分割数が爆発的に増える可能性がある．

(4) 問題を解く過程で得られる情報が，問題解決に必須なものしか含まれておらず，ノイズ情報が知識獲得にどのような影響を与えるかが検証されていない．

(5) 問題解決に関して，問題解決に至る一連のアクションは獲得できているが，手段目標分析は一部でしか実現ができていない．

(6) 実験環境が迷路のみであり，エージェントの問題解決能力の一般性の検証が不十分である．

本稿ではこれらの点を踏まえて先行研究を改良したアーキテクチャを提案する．

2. 提案アーキテクチャ

2.1 アーキテクチャ概要

図 1 にアーキテクチャ概要を示す．提案アーキテクチャでは Java をベースに設計されているが，知識処理には論理型言語である Prolog を用いた手段目標分析による問題解決を行う．その際の Prolog の Java インタプリタとして A Java Interface to Prolog（JPL）[7] を用いた．提案アーキテクチャにおいては，ある時点におけるエージェントの状況は論理式によって以下の例のように表現される．なお，提案アーキテクチャでは，それぞれの論理式を「状態」とし，状態の組み合わせを「状況」として表現する．

have(key)
now(home)
door(open)

この例においては，エージェントが鍵を持っており，家にいる状態であり，ドアが開いていることを表している．また，行動も状態と同様に論理式で表され，行動の種類を pred，項を term として pred(term) のように述語で表現する．例えば，鍵を手に入れるという行動は，get(key) のように表され，get が述語名，key が項である．提案アーキテクチャは問題環境および問題における選択可能な行動などの問題解決における事前知識を Prolog ファイルの形式で入力として受け取り，行動方策および最終目標状況を知識表現として出力する．ここで最終目標状況とは問題解決のために必要となる状態の組み合わせを指す．提案アーキテクチャの基本的な動作は問題解決メソッドとして手段目標分析を用いて問題を解決する．問題解決メソッドを先行研究で用いていた GNP から手段目標分析に変更することで，先行研究における問題点（1）のパラメータ設定の問題が解決できると考える．なお，提案アーキテクチャはエージェントアーキテクチャである．これは実際に問題環境において問題を解決するのはエージェントであり，アーキテクチャはそれらを統括する枠組みを提供するものである．本稿において，「経験」とはエージェントが問題を解決する過程で獲得した情報を指す．

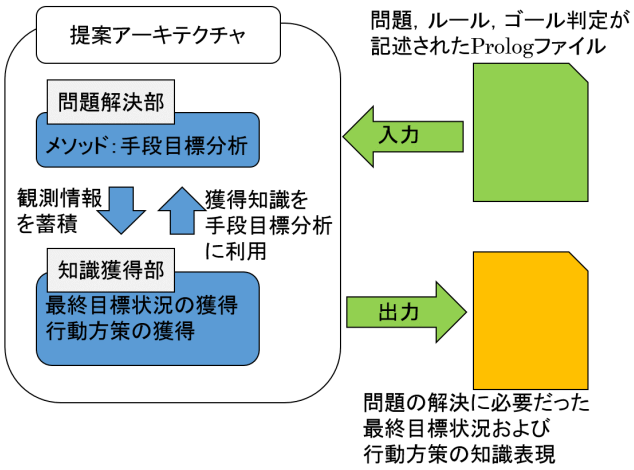

図 1 提案アーキテクチャ概要

2.2 手段目標分析のための基本機能

エージェントは手段目標分析によって問題解決を行うが，その実現のためにいくつかの機能を備えている．以下それぞれについて解説する．また，これらは暫定的なものであり，真の手段目標分析を実現するために随時拡張していくことが望ましい．

2.2.1 予測

エージェントはある状況においてある行動を選択した際，その結果発生する状態の変化を予測結果として観測することができる．この機能を用いることで，現状況と遷移可能な状況を知覚することができ，手段目標分析によるアプローチが可能になる．先行研究においては，GNP を用いて先に行動の系列を作り上げそれを実行するだけであった．そのためしばしば目標状況へと近づく行動が選択可能にも関わらず，現状況から目標に対しさらに離れてしまう行動を選択してしまうことがあった．提案アーキテクチャでは「予測」を用いることにより，先行研究の（5）の問題であった手段目標分析の実現が常に可能になると考える．

2.2.2 数の概念

手段目標分析においては，状態の差を小さくするということが重要となる．Prolog 処理系においては項はアトムまたは数によって表現される．状態の差については，アトム同士が近いかどうかの判定は難しいと考える．例えば，象と犬どちらが人間に近いか，といった問いに答えることは背景知識なしでは困難である．そこで，提案アーキテクチャには背景知識がなくともその距離を扱うことが可能である数の概念を与える．これにより，例えばエージェントは 1 と 2 ではどちらが 6 に近いかなどの理解が可能となり，手段目標分析による問題解決を効率的に行うことができる．また，先行研究の問題点（3）のように連続値を扱う場合での大量に部分ゴールを作成してしまう問題も回避可能になると考える．

2.3 問題解決部

エージェントは問題が与えられた当初は，その問題を解決するために必要となる最終目標状況がどのようなものかを理解していない．そのため，最初は最終目標状況を獲得し，次にそれを利用するために提案アーキテクチャは大きく分けて以下の 2 種類の問題解決アプローチを行う．

2.3.1 学習フェイズ

学習フェイズとは新たに与えられた問題など，その問題を解決するための具体的な状態の組み合わせが未知な場合を指す．その場合，アーキテクチャは「与えられた問題を解決する」という抽象的なゴールを設定する．このとき，エージェントは目指すべき目標状況が不明であるので手段目標分析による問題解決アプローチが取れない．そのような場合エージェントはその状況毎に選択可能な行動をランダムに選択する．ランダムな状況遷移を繰り返すことでゴールするために必要な状態の組み合わせの探索を行う．このような探索を行う場合，観測できた状態が多いほどその組み合わせで表現できる状況は多くなる．すなわちその組み合わせによって問題を解決する可能性が高くなることが考えられる．学習フェイズにおいてはこのヒューリスティクスに基づきエージェントはただ純粋なランダム探索を行うのではなく，自身が未だ観測したことのない状況に遷移するような行動を優先して行う．すなわち，ヒューリスティクスを意図として選定する．このような状況遷移を繰り返し問題が解決できたならば，後述する知識獲得部によりエージェントは最終目標状況および行動方策を獲得する．

2.3.2 応用フェイズ

応用フェイズとは学習フェイズで最終目標状況および行

動方策を獲得した後に問題を解く場合を指す．その場合エージェントは学習フェイズにおいて最終目標状況を獲得しているのでそれを用いて手段目標分析を行うことができる．その場合，エージェントは現状況と最終目標状況（別解として状況が複数存在することも考えられる）を用いて現状況から達成が最も容易である最終目標状況を選定する．そしてそれを実現するという意図を持ち，手段目標分析を用いて意図に沿った行動を行う．

2.4 知識獲得部

知識獲得部では問題解決部において問題を解決する過程で収集した情報を用いて知識の獲得を行う．収集される情報は以下の例のような形式で得られる．

PRESTATE:
have(none)
pos(5)
ACTION:
goto(6)
POSTSTATE:
have(key)
pos(6)

ここで，PRESTATE は行動を選択する前の状況を示し，ACTION は行動を，POSTSTATE は行動を実行した後の予測状況を示す．この例であれば移動をしたことで鍵を習得したということを表す．このような情報をトランザクション t と呼び，これを利用して以下に示す 2 種類の知識獲得を行う．なお，ここでエージェントそれぞれについて，収集したトランザクションの集合を T^{all}_{agent}（サイズを S_T とする）とし，その中に含まれるあるトランザクションを $t_n(0 \leq n \leq S_T)$ とする．また，t_n 中の PRESTATE に関する状況を pre_n（サイズ S_{pre-n}），POSTSTATE に関する状況を $post_n$（サイズ S_{post-n}）とし，それぞれの状態をアイテム $I_i(0 \leq i \leq S_{pre-n})$，$I_j(0 \leq j \leq S_{post-n})$ とする．また，あるトランザクション t_n の行動を a_n とし，a_n の述語名を $functor(a_n)$，項を $arg(a_n)$ とする．

2.5 最終目標状況の獲得

最終目標状況の獲得には，先行研究と同様に「問題を解決した際の状況には解となる状態の組み合わせが必ず含まれている」というヒューリスティクスに基づいて知識獲得を行う．エージェントはそれぞれ，エージェント毎に各試行で収集された情報 T^{all}_{agent} を用いて以下の手続き 1，2 を通じて最終目標状況を獲得する．以下にその獲得アルゴリズムを示す．全エージェントからなる集合を S_A とする．（サイズ $|S_A|$），あるエージェント $agent_n \in S_A$（n は識別子である）について，以下の手続き 1 を行う．また，具体例を図 2 に示す．

2.5.1 手続き 1：各エージェントごとの最終目標状況の獲得

(1) $agent_n$ について，T^n_{agent} の中から各試行において問題を解決したトランザクションの集合族 T^n_{goal}(サイズ $S_{T^n_{goal}}$) を抽出し，その中の各トランザクション $t_g(0 \leq g \leq S_{T^n_{goal}})$ の状況 $post_g$ から成る集合族 PS^n を抽出する．図 2 の例では $n=1$，$S_{T^n_{goal}}=3$ である．この場合，エージェント 1 は 3 回のゴールに成功したことを表す．

(2) $MS^n = I_{all} \subset \forall post_g \in PS^n$ かつ $\arg\max_{I_{all}}(|I_{all}|)$ ($|S|$ は集合 S の要素数を表す）となる集合 MS^n（必須状況）を獲得する．ここで，MS^n は問題を解決した時点で必ずエージェントの状況に含まれていた状況の集合族を表す．すなわち T^n_{goal} 内の全トランザクション t_g に共通して含まれていた状態の集合である．図 2 の例では，全ての $post_g$ に含まれていたアイテム $\{a(x), c(z)\} = MS^1$ となる．

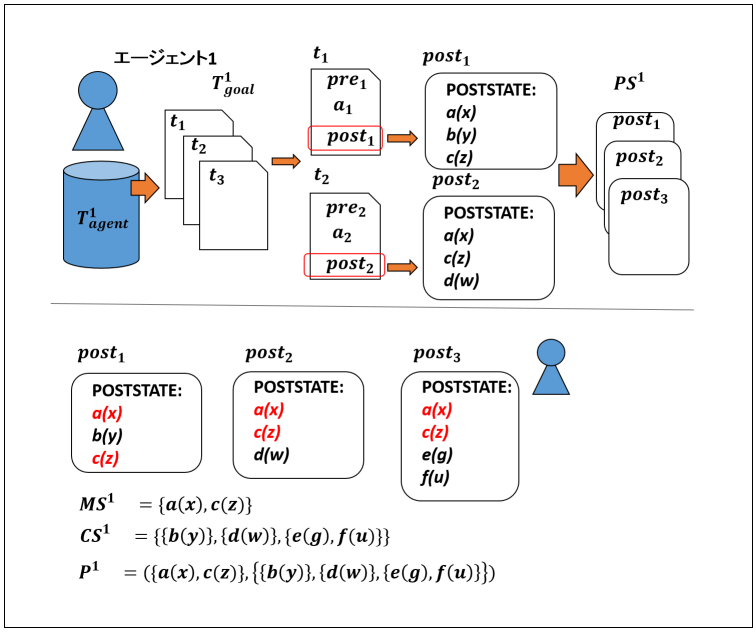

図 2 手続き 1 具体例

(3) $\forall post_g \in PS^n$ について，$CS^n = \{\bigcup_{\forall post_g \in PS^n} \{post_g \backslash MS^n\}\}$ となる集合族 CS^n（可能状況）を獲得する．ここで，CS^n は問題を解決した時点で状況に含まれていた状態の集合であるが，全ての t_g に共通しては含まれてはいなかった状態の集合を表す．図 2 の例では，それぞれの $post_g$ から $MS^1 = \{a(x), c(z)\}$ を除いた，$CS^1 = \{\{b(y)\}, \{d(w)\}, \{e(g), f(u)\}\}$ となる．

(4) エージェントが獲得した MS^n と CS^n から成るペア $P^n = (MS^n, CS^n)$ を作成する．

その後，各エージェントがそれぞれ獲得した P^n について，その集合族である
$P^{all} = \bigcup_{\forall agent_n \in S_A} \{P^n\}$ を作成する．

次に，以下の手続き 2 を行う．図 3 に具体例を示す．

2.5.2 手続き 2：各エージェントが獲得した最終目標状況の統合

(1) 任意のペア $P^m \in P^{all}$（m は識別子である）の必須状況 MS^m の中で，最も含まれるアイテム数が小さいものを $MS^{min} = \arg\min_{MS^m}(|MS^m|)$ とする．例においては，P^{min} は $|MS^m|$ が最も小さいエージェント 1（$m=1$）の P^1 となる．

(2) 最終目標状況ペア TP を MS^{min} が含まれるペア P^{min} で初期化する（$TP = P^{min}$）．

(3) $\forall P^m = (MS^m, CS^m) \in P^{all}$ に対し以下の操作を行う．

(a) $MS^{min} \not\subset MS^m$ を満たすならば，$P^m = (MS^m, CS^m)$ を最終状況ペアに加える（$TP =$

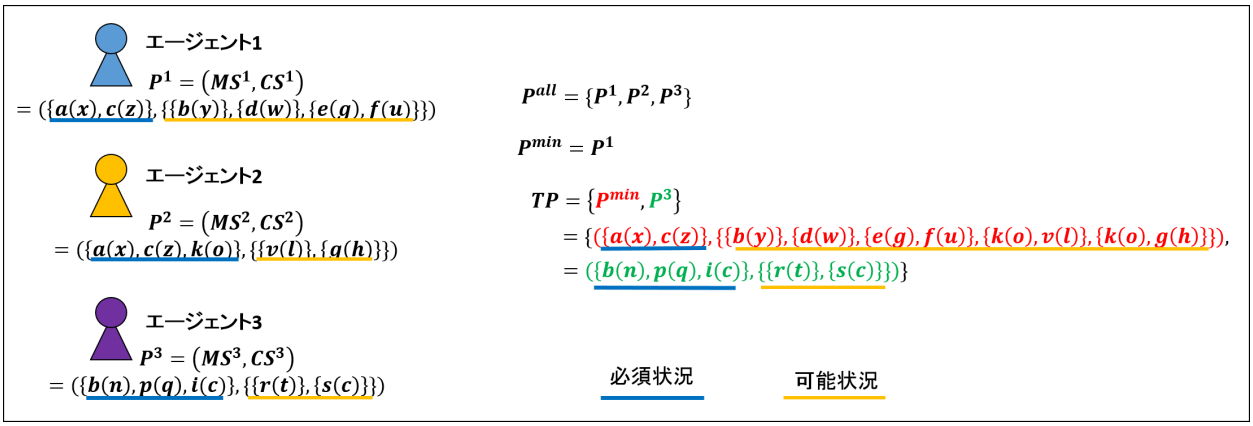

図 3 手続き 2 具体例

$TP \cup P^m$)．図 3 の例においては，$m = 3$ のエージェント 3 では $MS^{min} \not\subset MS^m$ が成り立つため，P^3 を TP に加える．

(**b**) $MS^{min} \subseteq MS^m$ を満たすならば，$MS^{dif} = MS^m \backslash MS^{min}$ として，$CS^{add} = \bigcup_{\forall S_{CS^m} \in CS^m} \{S_{CS^m} \cup MS^{dif}\}$ となる集合族 CS^{add} を作成する．この CS^{add} を用いて，TP の要素 $P^{min} = (MS^{min}, CS^{min})$ 中の可能状況 CS^{min} を $CS^{min} = CS^{min} \cup CS^{add}$ と更新する．図 3 の例においては，$m = 2$ のエージェント 2 では，$MS^{min} \subset MS^m$ が成り立つ．このとき，$MS^{dif} = MS^2 \backslash MS^{min} = \{k(o)\}$，$CS^2 = \{\{v(l)\}, \{g(h)\}\}$ となるので，$CS^{add} = \{\{k(o), v(l)\}, \{k(o), g(h)\}\}$ となり，$Y^{min} = \{\{b(y)\}, \{d(w)\}, \{e(g), f(u)\}, \{k(o), v(l)\}, \{k(o), g(h)\}\}$ と更新される．

この最終目標状況ペア TP の任意の要素 $P^l = (MS^l, CS^l)$（l は識別子である）を用いて，最終目標状況 $Target^l$ は，$Target^l = MS^l \cup \exists S \in CS^l$ と表現され，図 3 では，$\{a(c), c(z), k(o), v(l)\}$ や $\{b(n), p(q), i(c), r(t)\}$ となる．

2.6 行動方策の獲得

行動方策の獲得では，手段目標分析で用いる方策の獲得を行う．先行研究では解そのものを得ようとしていたため，ランダムな状態変化が起きるようなプロセスが解に含まれていた場合，解の再現ができずに毎回解を作り直すという先行研究における（2）のような問題点があった．提案アーキテクチャでは解そのものを得るのではなく解を実現するための方策を獲得する．ランダムな状態変化を起こす行動があったとしてもその結果に合わせて方策から行動を選択することで解にたどり着くことを目指し，先行研究における（2）のような問題を回避できると考える．提案アーキテクチャではある行動 a_n を選択したときに，現状況に依存せずに得られる項の変化に着目する．以下に行動方策獲得のアルゴリズムと具体例を図 4 に示す．

(1) まずエージェントがそれぞれ収集したトランザクション T^{all}_{agent} 内の全てのトランザクションに対し，そのトランザクションが持つ行動に着目する．全てのトランザクションを，行動の述語名 $functor(a_n)$ で分類する（$T_{functor(a_n)}$）．図 4 の例においては，f_1，f_2，f_3 の 3 つの行動をエージェントが取ったことを表す．例えばこれは，get(X)，move(X)，hit(X) というような行動をエージェントが取ったことを表す．

(2) さらに（1）にて $functor(a_n)$ で分類したトランザクションを，さらにその行動の項 $arg(a_n)$ で分類する（$T_{functor(a_n),arg(a_n)}$）．図 4 の例においては，述語名 f_1 の行動では，項が ar_1，ar_2，ar_3 の 3 つあったことを表す．例えばこれは，get(key)，get(money)，get(banana) というような行動をエージェントが取ったことを表す．以下，分類した $T_{functor(a_n),arg(a_n)}$ に含まれる全てのトランザクションについて，（3）～（4）を行い，(5) で行動 $T_{functor(a_n),arg(a_n)}$ についての方策を獲得する．

(3) 分類した $T_{functor(a_n),arg(a_n)}$ に含まれる全てのトランザクション $t_{functor(a_n),arg(a_n)}$ について，その $pre_{functor(a_n),arg(a_n)}$ と $post_{functor(a_n),arg(a_n)}$ に共通して存在する同一の述語名 $pred$ を持った要素 $state_{pre}$ と $state_{post}$ の組を全て抽出する．図 4 の例においては，t_1 では $pred = \{a, c, e\}$ が pre_1，$post_1$ に共通して存在する．

(4) （3）で抽出した要素の組について，その項 $term$ の種類によって以下の分岐および処理を行う．なお，項は問題で観測されたアトムあるいは数に具体化されているとする．

(**a**) 項がアトム同士あるいはアトムと数の場合，$state_{post}$ の項を変化として記録する．図 4 の例のトランザクション t_1 では，$pred = \{c\}$ となる要素について，$term$ である z を項の変化として記録する．

(**b**) 項が数同士の場合，$state_{post}$ と $state_{pre}$ の項の差を，{ 減少，増加 } の 2 値のうちのいずれかを用いて項の変化として記録する．図 4 の例のトランザクション t_1 では，$pred = \{a, e\}$ となる要素について，a は増加，e は減少を項の変化として記録する．

（a），（b）で記録した項の変化を $change$ として，$S_{t_n} = \bigcup_{\forall pred} \{(pred, change)\}$ となる集合 S_{t_n} をトランザクションごとに作成する．図 4 の例では，トランザクション t_1 については $S_{t_1} = \{(a,$ 増加$),(c, z),(e,$ 減少$)\}$ となる．

(5) $T_{functor(a_n),arg(a_n)}$ 内の全てのトランザクションの S_{t_n} からなる集合 S_{all} を作成する（$S_{all} = \bigcup S_{t_n}$）．作成した S_{all} を $S_{all} = S_{all} \backslash \{(a, b) | \forall (a, b) \in S_{all}, \exists (a, c) \in S_{all} \backslash \{(a, b)\}, b \neq c\}$ に更新する．図 4 の例では，$S_{t_1} = \{(a,$ 増加 $),(c, z),(e,$ 減少 $)\}$，$S_{t_2} = \{(a,$ 増加 $),(c, z),(e,$ 増加 $)\}$，$S_{t_3} = \{(a,$ 増加 $),(e,$ 増加 $)\}$ となるので，$S_{all} = \{(a,$ 増加 $),(c, z),(e,$ 増加 $),(e,$ 減少 $)\}$ となる．この S_{all} を更新すると，更新式における $\{(a, b)\} = \{(e,$ 増加 $),(e,$ 減少 $)\}$ となるので，$S_{all} = \{(a,$ 増加 $),(c, z)\}$

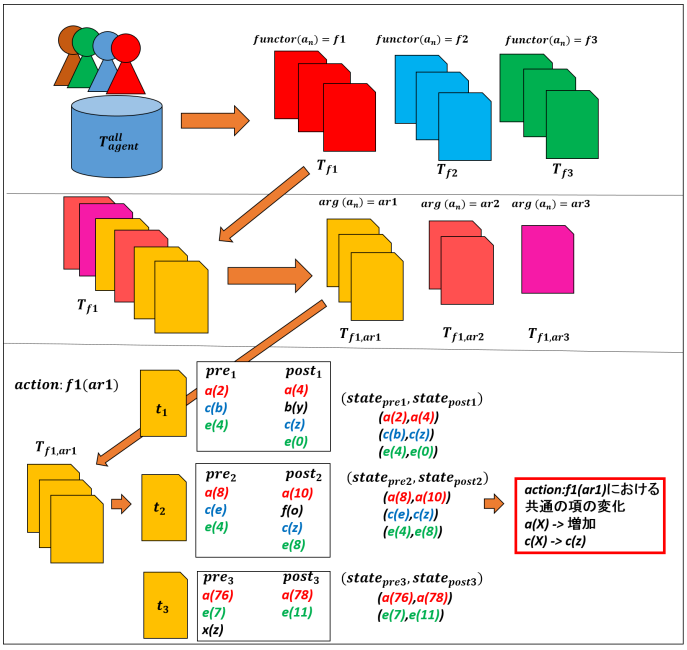

図 4　行動方策の獲得例

となる．すなわち 3 つのトランザクションにおいて，$(a,$ 増加 $), (c, z)$ が無矛盾なので，行動 $f_1(ar_1)$ を実行するとどのような状況でも $a(X) \rightarrow$ 増加, $c(X) \rightarrow c(z)$ の変化が発生するという知識を方策として獲得する．

このようにして $T_{functor(a_n),arg(a_n)}$ 毎に獲得された項の変化を行動 a_n の方策として獲得する．

3. シミュレーション実験

本稿では 3 つの実験環境で実験を行うことにより提案アーキテクチャの問題解決能力の汎用性の検証および先行研究アーキテクチャの問題点 (4)(6) についての検証を行う．

3.1 実験 1：先行研究実験環境での実験

3.1.1 実験目的

提案アーキテクチャは先行研究における問題点を基に改良を加えた．従って，先行研究アーキテクチャで解決可能な問題が解決できることが当然必要となる．本セクションでは先行研究の実験で用いた迷路環境を用いて提案アーキテクチャでのシミュレーション実験を行う．

3.1.2 実験設定

図 5 に実験 1 で用いた迷路環境を示す．この環境は先行研究において用いた実験環境と同様で，問題設定も同じくエージェントは S のスタートから探索を行い，G のゴールにたどり着くことを目標としている．しかし，G のゴールの前にはドアが存在し，K の鍵を取得した後，ドアを開かなければエージェントはゴールに到達することができないことも同様である．

3.1.3 パラメータ設定

表 1 に実験 1 のパラメータを示す．

表 1　パラメータ設定

エージェント個体数	1
試行数	100
1 試行中のステップ数	300

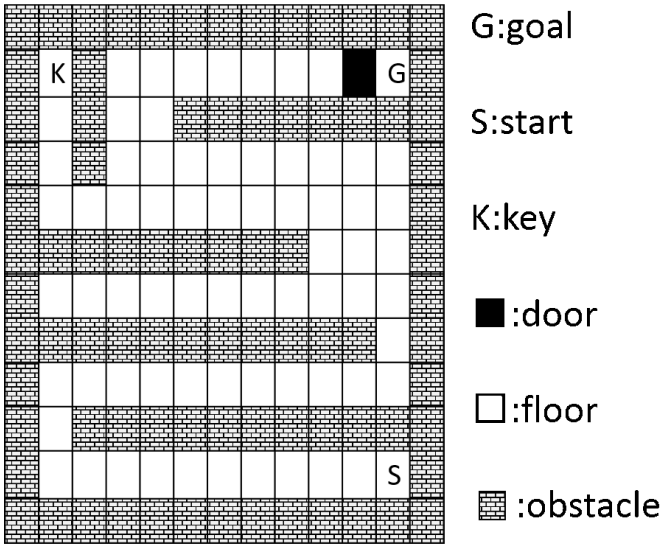

図 5　迷路環境

先行研究と同様に，エージェントが 1 回の試行で取れる行動回数（ステップ）は 300 とした．先行研究においては GNP を用いていたため，複数のパラメータを設定したが，提案アーキテクチャにおいてはそれらは設定せず，実験上のパラメータのみを設定する．さらに，表 2 に実験 1 でエージェントが選択できる行動を示す．

表 2　エージェントの選択可能な行動

述語名	項
move	X

この行動において，項 X はグリッドを表す．X は後述する状態 now(P) の項 P と同様の状態を取る．エージェントは Prolog ファイルによって記述されたルールから，現在のグリッドに隣接するグリッドのうち，移動可能なグリッドの情報を獲得する．そして，その移動可能なグリッドのうちの 1 つを選択して移動する．次に，表 3 にエージェントが観測できる状態を示す．
それぞれの状態について述べる．まず，now(P) はエージェントの位置を表す．しかし，正確な座標を観測しているのではなく，単純にそれぞれのグリッドが異なるものであるということを認識しているだけである．項 P が取りうる状態は，{a,b,c,d,...,z,aa,ab,...,az,ba,bb,...,bz,ca,...,ez} の 156 通りである．このとき a は最も左上のグリッドを指し，m は最も右上を指す．ez は最も右下である．すなわち，エージェントは el からスタートし，y のゴールを目指す．ドアは x，鍵は o に存在する．次に door(S) はドアの状態を指す．項 S は先行研究と同じく {open,close} の 2 通りの状態を取る．最後に have(O) はエージェントの所持物を指す．項 O も先行研究と同じく {key,none} の 2 通りの状態を取る．実験 1 においてエージェントの状況はこれら 3 つの状態を組み合わせて表現される．なお，エージェントの初期状況，および問題の解決のために必要な条件を表 4 と 5 に示す．

ここで，問題を解決するための条件はゴールの位置にたどり着くことであり，そのために必要となるドアを開くことは条件に考慮されていない．実験 1 では，最初に 100 試行の学習フェイズを行った後，知識の獲得を行う．そして，獲得した知識を用いて再度 100 試行の応用フェイズを行い

表 3　エージェントの観測できる状態

状態	内容
now(P)	エージェントは P にいる
door(S)	ドアが S である
have(O)	エージェントは O を所持している

表 4　初期状況

初期状況
now(el)
door(close)
have(none)

表 5　問題解決に必要な条件

問題解決に必要な条件
now(y)

問題を解くことでアーキテクチャの性能を評価する．

3.2　実験 1 結果

表 6，7 に獲得された知識表現を，表 8 に実験結果を示す．

表 6　最終目標状況

MUSTSTATE:
now(y)
have(key)
door(open)

表 7　行動方策（一部）

ACTION:move(o)
now(o)
have(key)

表 8　実験 1 結果（100 試行 1 エージェント計 100 試行）

	学習フェイズ	応用フェイズ
ゴール数	49	100
平均ステップ	196.8	149

3.2.1　考察

実験結果から，提案アーキテクチャを用いて先行研究と同様に迷路問題が解決可能であることが確認された．また，学習フェイズでは 100 試行中半分程度しかゴールできなかったのに対し，応用フェイズにおいては 100%解が達成できており，ゴール達成率に 50%の上昇が確認できる．また，平均ステップ数も 50 ステップの改善が確認された．一方で，先行研究とと比較すると，先行研究で獲得された最終平均ステップ数には劣っている（先行研究では 75 ステップ程度）．これは，提案アーキテクチャの問題解決メソッドに最適化という概念が存在しないためであると考える．適応度関数や評価関数といった解の質を向上させるための仕組みを導入することでこの問題は解決できると推測するが，問題毎にそれらを用意し作りこむことは開発者に大きな負担になると考えるとともにアーキテクチャの使いやすさが損なわれると考える．そのため，本アーキテクチャではパラメータなどの設定を極力排除し，解の質を知識獲得によって補っている．
次に知識獲得による解の質について考察する．充分な学習がなされたあとのステップ数はたしかに先行研究に劣ってはいるが，実験 1 で行われた試行数は学習，応用を含めて高々 200 試行であり，これは先行研究の 1 世代の試行数よりも少ない．ここで先行研究における 1 世代目の獲得解に着目すると，ゴールに 250 ステップ以上かかっていることが確認できる．先行研究の解が 149 ステップ以内になるのはおよそ 250 世代頃であり，それまでには 75000 試行が必要となる．従って，提案アーキテクチャは解の質を向上させる能力では劣っているが，少ない試行で有効な解を得られる可能性が示唆された．
最後に，獲得された知識表現に着目する．最終目標状況に着目すると，ゴール条件である now(y) が必須条件として確認できる．問題解決のために必要な条件として与えたのはこの状態だけであるが，この問題では暗黙的に鍵を取得し，ドアを開くことがゴールのためには必須である．それらの条件もゴールのために必要な知識として獲得できていることが確認できる．また，表 7 の獲得された行動方策に着目すると，o のグリッドにたどり着くことで鍵が取得できる（have(key)）ことが知識として獲得できている．

3.3　実験 2：ボードゲーム Parchis における問題解決

3.3.1　実験目的

実験 2 ではボードゲーム Parchis[8] を環境とし，アーキテクチャの汎用性および観測状態にノイズが¥含まれる問題における性能を検証する．

3.3.2　Parchis

まず，本セクションで実験環境として用いる Parchis について述べる．Parchis のゲーム盤を図 6 に示す．このボードゲームは 4 人対戦型のすごろく型ボードゲームで，プレイヤーはそれぞれ自分に割り当てられた色を 1 色持つ．プレイヤーはそれぞれサイコロを用いて自分が持つ 4 つの駒をゴールさせるのが目的である．ゲームの原則として，1 つのマスに 2 つまでしか駒は存在できない．以下ルールについて述べる．

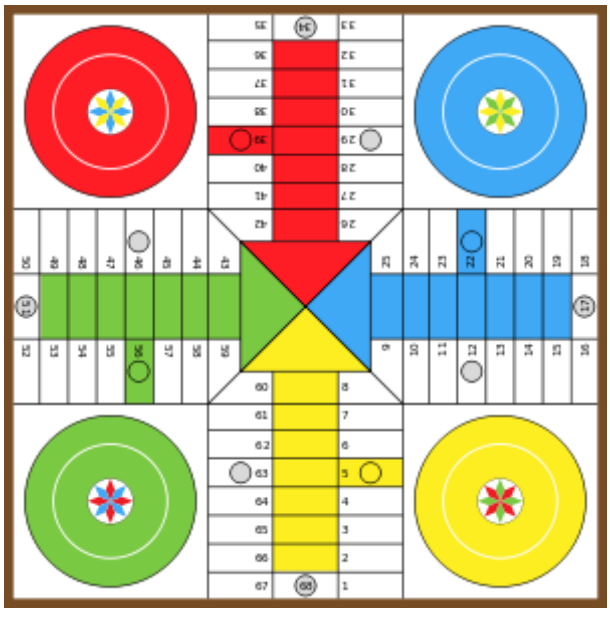

図 6　ゲーム盤 [8]

3.3.3　ゴール条件

ゴールのための条件は自分の色に対応する 4 つの駒全てを矢印マスの先頭（ゲーム盤の中心）に移動させることである．なお，ゴールはそれぞれの色のスタート位置から 72 マス目に存在する．

3.3.4　実験設定

実験 2 では Parchis のルールを記述した Prolog ファイルを入力として，提案アーキテクチャを用いて 4 エージェン

トでのゲームプレイを通じて問題を解決し，最終目標状況および行動方策の知識表現の獲得を目指す．なお，ここでの最終目標状況の獲得とは，ボードゲームのゴールのための条件を獲得することを意味する．

3.3.5 パラメータ設定

表 9 に実験 2 のパラメータを，表 10 にエージェントが実行可能な行動を示す．

表 9 パラメータ設定

エージェント個体数	4
試行数	300
1 試行中のステップ数	100

表 10 エージェントの選択可能な行動

述語名	項
mvr1	X
mvr2	X
mvr3	X
mvr4	X
mvb1	X
:	:
mvg1	X
:	:
mvy1	X
:	:
mvy4	X

この行動において，項 X はマスを表す．mvr3 は色が赤(red) の 3 つ目の駒を移動させることを表す．同様に mvg1 は色が緑（green）の 1 つ目の駒を移動させることを表す．なお，ここで X はそれぞれの色のスタートを 1，ゴール（矢印の先）を 72 としたエージェントの相対座標を表す．絶対座標はゲーム盤上のマスである．すなわち，もし青の駒 2 をゲーム盤の 33 マス（絶対座標）へ移動させたいならば，mvb2（12）という行動を選択する．エージェントは Prolog ファイルによって記述されたルールを基にサイコロを振り，移動可能なマスの相対座標情報を獲得する．そして，その移動可能なマスのうちの 1 つを選択して移動する．次に，表 11 にエージェントが観測できる状態を示す．ここでゲームをプレイしている人間と同様にエージェントは盤面の状態を確認できるとし，自身の駒だけではなくほかのエージェントが操作する駒の位置も観測することができる．それぞれの状態における項 P は {0,1,2,3,...,72} までの 73 通りの状態を取る．これらはエージェントの相対座標であり 1 はスタート，65 以降は矢印マスを表し，72 がゴールである．実験 2 においてエージェントの状況はこれら 17 状態を組み合わせて表現される．

次にエージェントの初期状況，および問題の解決のために必要な条件の例を表 3.3.5 と表 3.3.5 に示す．この例は赤色が割り当てられたエージェントの例である．もしエージェントが黄色であれば，初期状況における corlor の項が r から y に変わり，問題解決に必要な条件は r1，...，r4 ではなく y1，...，y2 となる．実験 2 においてはエージェントが観測できる状態に自身のゴールに無関係な状態が含まれている．そのため，先行研究における問題点（4）であったゴールに無関係な状態が含まれる環境下での知識獲得の性能評価が行えると考える．実験 2 においても実験 1 と同様に最初に 300 試行の学習フェイズを行った後，知識の獲得を行う．そして，獲得した知識を用いて再度 300 試行の応用フェイズを行い問題を解くことでアーキテクチャの性能を評価する．

表 11 エージェントの観測できる状態

状態	内容
color(C)	エージェントの色は C である
r1(P)	赤の駒 1 は P にいる
r2(P)	赤の駒 2 は P にいる
r3(P)	赤の駒 3 は P にいる
r4(P)	赤の駒 4 は P にいる
b1(P)	青の駒 1 は P にいる
:	:
g1(P)	緑の駒 1 は P にいる
:	:
y1(P)	黄の駒 1 は P にいる
:	:
y4(P)	黄の駒 4 は P にいる

表 12 初期状況

初期状況
color(r)
r1(1)
r2(0)
r3(0)
r4(0)
b1(1)
b2(0)
b3(0)
b4(0)
g1(1)
g2(0)
g3(0)
g4(0)
y1(1)
y2(0)
y3(0)
y4(0)

表 13 問題解決に必要な条件

問題解決に必要な条件
color(r)
r1(72)
r2(72)
r3(72)
r4(72)

3.3.6 実験 2 結果

表 14，15，16，17，18 に獲得された最終目標状況に関する知識表現を，表 19 に実験結果を示す．

3.3.7 考察

実験結果から，このボードゲーム環境においても提案アーキテクチャは問題を解決できていることが確認できる．学習フェイズにおいては 1200 試行中 1091 回とほぼ 90%近い問題解決率を達成している．学習後はほぼ 100%のゴール達成に成功しており，平均ステップ数も減少している．これは，最終目標状況が設定できるようになったことで，例えば，r1(67)，r2(50) といった状況においてサイコロで 5 の目が出た際に，r1(72) となる行動を一意的に選択することで駒をゴールに進め易くなったためだと考えられる．しかし，解の達成率や総ステップ数が共に改善されたといえども，もともとの問題解決率が高いためその改善率は 10%ほどと実験 1 よりも大きく劣っている．これは，このゲームにおいては行動の選択余地がほぼ存在しないことやそもそも問題の解決が用意な環境であったことが原因であ

表 14 最終目標状況（赤色のエージェント）

MUSTSTATE:
color(r)
r1(72)
r2(72)
r3(72)
r4(72)

表 15 最終目標状況（青色のエージェント）

MUSTSTATE:
color(b)
b1(72)
b2(72)
b3(72)
b4(72)

表 16 最終目標状況（緑色のエージェント）

MUSTSTATE:
color(g)
g1(72)
g2(72)
g3(72)
g4(72)

表 17 最終目標状況（黄色のエージェント）

MUSTSTATE:
color(y)
y1(72)
y2(72)
y3(72)
y4(72)

表 18 行動方策（赤色のエージェント一部）

ACTION:mvr1(3)
color(r)
up(r1)

表 19 実験 2 結果（300 試行 4 エージェント計 1200 試行）

	学習フェイズ	応用フェイズ
ゴール数	1091	1190
平均ステップ	75.3	63.3

ると考えられる．そのことから，問題解決が容易な問題や行動に選択の余地が少ない問題などにおいては提案アーキテクチャの知識利用の効果が薄い可能性がある．

次に獲得された最終目標状況に着目すると，必須状態としてそれぞれのエージェントがゴールのために必要な条件を獲得していることが確認できる．最後に表 18 の実験 2 において獲得された行動方策に着目する．この問題においては，観測状態の中から共通する項の変化を抽出しようとした際に後退する場合があまり存在しないため，up(r1) のような「とにかく行動することで前進できる」という当然の方策しか獲得できなかった．

3.4 実験 3 : ボードゲーム Hase und Igel における問題解決

3.4.1 実験目的

実験 3 ではボードゲーム Hase und Igel[9] を環境とし，アーキテクチャの汎用性および最終目標状況が複雑な条件で与えられる問題での性能を評価する．

3.4.2 Hase und Igel

まず，本セクションで実験環境として用いる Hase und Igel について述べる．Hase und Igel のゲーム盤を図 7 に示す．このボードゲームは複数人対戦型のすごろく型のボードゲームで，プレイヤーはそれぞれ自分の駒をゴールさせるのが目的である．原則として 1 つのマスには 1 つの駒しか入ることができない．以下ルールについて述べる．

3.4.3 ゴール条件

ゴールのための条件はゴールマスに到着することであるが，そのためには以下の制約を満たさなくてはならない．

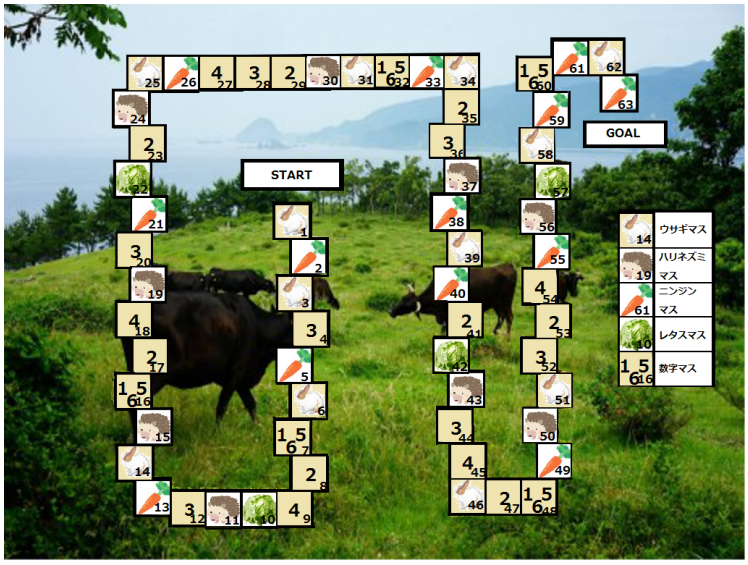

図 7 ゲーム盤

(1) 所持しているニンジンの数が O*10 本以下であること．（O は現在の順位であり，順位はゴールに近いマスにいるほど高くなる）

(2) レタスの数が 0 個であること．

3.4.4 実験設定

実験 3 では Hase und Igel のルールを記述した Prolog ファイルを入力として，提案アーキテクチャを用いて 4 エージェントでのゲームプレイを通じて問題を解決し，最終目標状況および行動方策の知識表現の獲得を目指す．なお，ここでの最終目標状況の獲得とは，ボードゲームのゴールのための条件を獲得することを意味する．

3.4.5 パラメータ設定

表 20 に実験 3 のパラメータを，表 21 にエージェントが実行可能な行動を示す．

表 20 パラメータ設定

エージェント個体数	4
試行数	300
1 試行中のステップ数	50

表 21 エージェントの選択可能な行動

述語名	項
decide	X

この行動において，項 X は通常移動における「移動可能なマス」あるいはニンジンマスやウサギマスに止まった際のイベントの結果としてニンジンを「10 本獲得するか 10 本減らす，あるいは通常の移動を行う」の 3 つの選択を表す．もし通常移動であれば X の取りうる状態は {0,1,2,...,64} の 65 状態である．このとき 0 はスタートマスを，64 はゴールマスを表す．また，ニンジンマスやウサギマスでのイベントの際には X は {plus,minus,move} の 3 つの状態を取る．このとき，plus はニンジンを 10 本獲得することを，minus はニンジンを 10 本減らすことを，move は通常移動を行うことを表す．move によって通常移動を行う場合，項 X には新たに移動可能マスがエージェントは Prolog ファイルによって記述されたルールを基に移動可能なマスあるい

表 22 エージェントの観測できる状態

状態	内容
carrot(C)	エージェントの持つニンジンは C 本である
lettuce(L)	エージェントの持つレタスは L 個である
order(O)	エージェントの順位は O である
now(P)	エージェントは P にいる
state(S)	エージェントが止まっているマスは S マスである

は選択可能な行動を獲得する．そして，その移動可能なマスあるいは選択可能な行動のうちの 1 つを選択して移動する．次に，表 22 にエージェントが観測できる状態を示す．それぞれの述語について解説する．まず，carrot(C) はエージェントの持つニンジンの本数を表し，C は {0,1,....} と無数の状態を取りうる．次に lettce(L) はレタスの個数を表す．Hase und Igel のルールではレタスが増加することはないので，L の取りうる状態は {0，1，2，3} の 4 状態である．order(O) はエージェントの順位を表す．実験 3 では，4 エージェントによるゲームプレイを行うので，O の取りうる状態は {1，2，3，4} の 4 状態である．now(P) はエージェントの現在いるマスを表す．P の取りうる範囲はエージェントの選択可能な行動でも述べた通り，{0，1，2，..., 64} の 65 通りである．最後に，state(S) はエージェントの現在いるマスがどのような種類のマスかを表す．S の取りうる状態は，{start，goal，hase，igel，carrot，lettuce，s156，s2，s3，s4} の 10 状態である．このとき，hase はウサギマスを，igel はハリネズミマスを，数字の入っている状態は数字マスを表す．実験 3 ではエージェントの状況はこれら 5 つの状態の組み合わせにより表現される．

次に，実験 3 におけるエージェントの初期状況および問題解決のために必要な条件を表 3.4.5 と表 3.4.5 に示す．

表 23 初期状況

初期状況
carrot(65)
lettuce(3)
order(1)
now(0)
state(start)

表 24 問題解決に必要な条件

問題解決に必要な条件
state(goal)

実験 3 では問題解決に必要な条件はゴールにたどり着くことであり，そのために必要なニンジンやレタスに関する条件は考慮されていない．この実験 3 はいわば実験 1 のゴール条件をさらに複雑化したものであり，アーキテクチャの汎用性および知識獲得の有効性が検証できると考える．実験 3 においても実験 1，2 と同様に最初に 300 試行の学習フェイズを行った後，知識の獲得を行う．そして，獲得した知識を用いて再度 300 試行の応用フェイズを行い問題を解くことでアーキテクチャの性能を評価する．

3.4.6 実験 3 結果

表 3.4.6，3.4.6 に実験 3 で獲得された知識表現を示す．次に，表 27 に実験結果を示す．

3.4.7 考察

実験結果から，このボードゲーム環境においても提案アーキテクチャは問題を解決できていることが確認できる．学習フェイズにおいては 1200 試行中 83 回しかゴールできなかったのに対し，応用フェイズにおいてはおよそ 10 倍の 803 回のゴールに成功している．また，平均ステップ数もわずかではあるが短縮されており，解の質も向上していることが確認された．実験 3 における学習前と学習後のゴール回数の改善率は 7%から 67%へと 3 つの実験の中でも最も高い 60%の改善となっている．そのことから，ゴールするために必要な条件が厳しい制約になるほど，最終目標状況を用いた手段目標分析によるアプローチが有効である可能性が示唆される．

表 25 最終目標状況（一部）

MUSTSTATE:
lettuce(0)
now(64)
state(goal)
CANSTATE:
carrot(3)
order(1)
CANSTATE:
carrot(8)
order(2)
CANSTATE:
carrot(8)
order(1)
CANSTATE:
carrot(11)
order(2)

表 26 行動方策（一部）

ACTION:decide(24)
down(now)
up(carrot)
state(igel)
ACTION:decide(57)
down(lettuce)
up(now)
state(lettuce)

表 27 実験 3 結果（300 試行 4 エージェント計 1200 試行）

	学習フェイズ	応用フェイズ
ゴール数	83	803
平均ステップ	33.2	32.1

次に獲得された最終目標状況に着目すると，必須状態として，ゴールのために必要な条件である state(goal) が含まれていることが確認できる．また，それ以外にも now(64) という状態が含まれていることが確認できる．この目標状態を獲得し手段目標分析を行うことは，ゴールに近づく行動を選択しやすくなるということを示す．これは，ほかの強化学習などにおけるゴールまでの距離が近いほど報酬が高くなるといった評価関数に相当するものであると考えられ，提案アーキテクチャは評価関数を開発者が与えなくとも，知識獲得によりそれに類するものを獲得できることが示唆された．また，実験 3 においてはゴールに到達するためにはレタスとニンジンに関する制約が存在する．レタスに関してはゴールのときに必ず 0 個であるので，必須状態として獲得されているが，ニンジンは順位によってその本数に幅が存在する．そのため，可能状態として知識獲得されていることが確認できるが，この可能状態と必須状態を組み合わせることで，ニンジンのように範囲を持つような解も最終目標状況として表現することが可能となる．

最後に獲得された行動方策に着目する．これらは一部抜粋したものであるが，decide(24) に着目すると，図 7 において 24 マスはハリネズミマスである．ハリネズミマスには後退することでしか侵入できないので，ハリネズミマスに止まる行動を選択することで now(X) は現在の位置よりもゴールから離れてしまう（down(now)）がニンジンの本数は増加する（up(carrot)）ことが知識として獲得できていることが確認できる．また，decide(57) に着目すると，57 マスは図 7 においてレタスマスである．そこに止まる行動を選択することでレタスの個数を減らす（down(lettuce)）こ

とが可能になることが知識として獲得されている．

4. おわりに

本稿では，実世界における一般的な問題を解決するための知的な人工知能の設計を目指して手段目標分析を用いて問題を解決するエージェントアーキテクチャを提案した．その際，先行研究においては複数の問題点が存在することを考察した．それら問題に対し，以下のようなアプローチを加え，先行研究を改良したアーキテクチャを提案した．

(1) 問題解決メソッドに手段目標分析をそのまま採用することで細かなパラメータの設定を省略可能にした．

(2) 解そのものを獲得するのではなく，手段目標分析によって解に至ることができるような行動方策，目標状況を獲得するよう改良した．それによって途中で確率的な変化があったとしても，変化後の状況から手段目標分析によって問題が解決が可能となった．

(3) 手段目標分析を用いることにより，ゴールを独立したいくつかのサブゴールに分割し解決するのではなく，現在の状態からゴールに近づくことで問題解決を行うよう改良し，サブゴール爆発の問題を解決した．

(4) 実験 2 としてあるエージェントにとっては問題解決に必要であるがほかのエージェントにはノイズとなるような観測状態が含まれる環境を構築し実験を行った．

(5) GNP によるランダム探索ではなく，行動方策を用いることで手段目標分析がそのまま行えるよう改良した．

(6) 迷路以外にも，実験 2，3 で異なる 2 種類のボードゲーム環境を構築し実験を行った．

また，提案したアーキテクチャを用いて 3 種類の実験を行い，その性能を検証した．
実験 1 においては，先行研究が解決した問題環境環境においても提案アーキテクチャが動作することを確認し，問題解決のために必要な知識表現が獲得できていることを確認した．また，先行研究における GNP 初期世代にあたる試行数では提案アーキテクチャのほうが良質な解を獲得できていることを確認した．実験 2 においては，解が複数存在し，観測状況にノイズを含むような環境での問題解決および知識獲得を行い，問題解決のためにに必要な知識表現が獲得できていることを確認した．一方，行動にあまり選択の余地が存在しないような問題では，提案アーキテクチャの知識獲得はあまり有効に働かない可能性が示唆された．
実験 3 においては，より複雑なゴール条件を持つ問題に対し問題解決および知識獲得を行い，問題解決のために必要な知識表現が獲得できていることを確認した．また，より複雑なゴール条件を持つような問題において提案アーキテクチャにおける知識獲得および手段目標分析が有効に働く可能性が示唆された．

4.1 今後の課題

今後の課題としては，大きく 3 つの改良が考えられる．
1 点目は最終目標状況の洗練である．
現在，最終目標状況における可能状態集合にはエージェントの観測できる状態全てが含まれている．そのため実験 2 の知識獲得の結果のように，必須状態集合に含まれる状態だけで真のゴールを満たしているような場合でも，可能状態も考慮した手段目標分析を行ってしまうという問題がある．このように不必要な可能状態が大量に存在する場合，メモリを圧迫し，また行動決定のコストが高くなってしまうという状況が起こる．この問題を解決するために可能状態集合の洗練手法を考案したいと考える．現在のアイデアとしては，観測頻度に基づき統計的な処理を加えることで洗練を行うというものがある．
2 点目は行動方策の獲得方法の緩和である．
現在，行動方策は状況に寄らず全てで共通する状態変化の傾向を行動方策として獲得している．しかし，実際には確率的に偏った状態変化を起こす行動や，たった 1 度その傾向が観測されなかっただけで切捨てられてしまった状態変化の傾向も存在する．したがってこれらの高頻度で起こる状態変化も獲得できるよう行動方策の獲得アルゴリズムを改良したいと考える．こちらも改善点 1 点目と同様に，観測頻度に基づく統計処理によるアプローチを考えている．
3 点目は手段目標分析の改良である．
現在，手段目標分析はある行動を選択した際にある状態が達成できるかということしか考慮していない．しかし，現実ではある状態が達成されたときには異なるある状態が未達成となってしまうような状況が充分考えられる．そこで，それらを考慮した手段目標分析が行えるように改良を加える必要がある．また，現在提案アーキテクチャは一手先を考慮したグリーディな手段目標分析しか行っていない．従って，問題設定によってはある 2 つの状況を行き来することも考えられる．これを回避するため，一手先のみだけでなくゴール方向からトップダウンでの手段目標分析と組み合わせるなどの改良を加えたいと考える．
そのほかにも本稿で扱ったボードゲームはすごろく型のものであり，ボードゲームの中でも 1 つのカテゴリにすぎない．本当の意味での汎用性を実現し，一般的な問題を解決するためには，今回導入した数の概念のように，多様な概念をアーキテクチャに加えて改良していくことが望まれる．

参考文献

[1] 森田純哉：文献紹介認知アーキテクチャを利用したスキル獲得に関する最近の研究の紹介，認知科学，Vol. 15, No. 4, pp. 699–704 (2008).
[2] 巻渕有哉，申　富饒，長谷川修：実世界における一般問題解決システムの提案とそのヒューマノイドロボットへの実装 (人工知能, データマイニング)，電子情報通信学会論文誌. D, 情報・システム， Vol. 93, No. 6, pp. 960–977 (2010).
[3] Weizenbaum, J.: ELIZA - A Computer Program For the Study of Natural Language Communication Between Man And Machine, *Communications of the ACM*, Vol. 9, No. 1, pp. 36–45 (1966).
[4] Homma, T., Atlas, L. and , R. M.: An Artificial Neural Network for Spatio-Temporal Bipolar Patters: Application to Phoneme Classification, *Advances in Neural Information Processing Systems*, Vol. 1, pp. 31–40 (1988).
[5] 森田信吾，加藤昇平：GNP を用いて知識を自己獲得・利用するエージェントアーキテクチャ，情報処理学会 172 回知能システム研究会 (2014(採録決定)).
[6] 間普真吾，平澤宏太郎，古月敬之，JINGLU, H.：強化学習を用いた遺伝的ネットワークプログラミングとそのエージェントの行動生成における性能評価，情報処理学会論文誌，Vol. 46, No. 12, pp. 3207–3217 (2005).
[7] : JPL. http://www.swi-prolog.org/packages/jpl/java_api/.
[8] : Parchis-ウィキペディア. https://en.wikipedia.org/wiki/Parch%C3%ADs.
[9] : Hase und Igel-ウィキペディア. https://de.wikipedia.org/wiki/Hase_und_Igel.