3S-1

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

森田信吾† 加藤昇平†

†名古屋工業大学

1 はじめに

人工知能の研究テーマの一つとして自動計画がある．自動計画はプランニング問題に適用されるもので，与えられた問題の初期状態，目標状態，問題に対してエージェントがとりうるアクション集合を入力として，目標状態を達成する一連のアクション系列を生成する．自動計画を実現する手法としてSTRIPS[1]などが提案されている．しかし，自動計画においてはあらかじめアクション集合を定義しなければならず，実世界における問題を解く場合とりうるアクションをすべて定義することは不可能である．そのため，与えられた問題が動的に変化するような場合は新たにアクションを獲得したり，既存のアクションを組み合わせて新たなアクションを生成できることが望ましい．そのため本稿では，述語表現によって表された知識をアクションとし，知識を組み合わせることで与えられた問題を手続き化し，部分的なゴールに分割する．その後，進化計算手法の一つであるGNP[2]を用いて部分ゴールを達成することにより，問題全体を解決するアーキテクチャを提案する．

2 提案手法

図1に本稿で提案するアーキテクチャの概要を示す．提案アーキテクチャは大きく分けてプランナー，SolverTree，知識抽出器，知識プールの4つの要素から構成される．プランナーはある問題およびゴール状態が与えられた際，知識プールに存在する知識を用いてゴール状態の達成に必要な部分ゴール状態群を作成する．なお提案手法において知識は述語表現で表される．知識の例を以下に示す．

$$\mathrm{have(key1)} : \text{-}\ \mathrm{at(key1), find(key1, movable)}.$$

後件に当たる箇所がその知識におけるゴール状態であり，前件はそれぞれ部分ゴール状態となっている．後件を達成するために前件を満たす必要があり，知識は一種の手続きであるとみなせる．

2.1 プランナー

プランナーは与えられた問題およびゴール状態が現在知識プールに存在する知識を利用して解決できるかを判定する．解決可能であれば，そのために必要な部分ゴール状態群をもとに後述するSolverTreeをトップダウンで作成する．解決が不可能であれば，単一のSolverのみを用いて最終ゴール状態を達成するGNPグラフを作り上げる．

2.2 SolverTree

SolverTreeはSolverからなる木構造である.Solverは一つのゴール状態を持ち，そのゴール状態を達成するためのGNPグラフ，さらにエージェント群および子Solver群を持つ．SolverTreeはボトムアップ的にSolverの持つGNPグラフを組み上げることで問題解決を図る．個々のエージェントはGNPグラフを一つずつ持ち，それらを進化させてゴール状態の達成を目指す．各Solverはゴール状態を達成したエージェント群の中から最も適応度が高いものを一体選び，そのエージェントが持つGNPグラフを自身のGNPグラフとする．終端Solverは子Solver群を持たないので，自身に与えられたゴール状態を達成するGNPグラフを作成する．非終端Solverは子Solver群が作成したGNPグラフを連結させ，さらに自身のゴール状態を達成するGNPグラフを作り上げる．ここで任意のSolverは知識と対応しており，子Solver群が知識の前件，自身に与えられたゴール状態が知識の後件に相当する．また，SolverTreeは一度完成した終端Solverおよび非終端Solverから適応度の低いGNPグラフを持つSolverを選び，新たに進化させていくことでより最適な解を探索する．

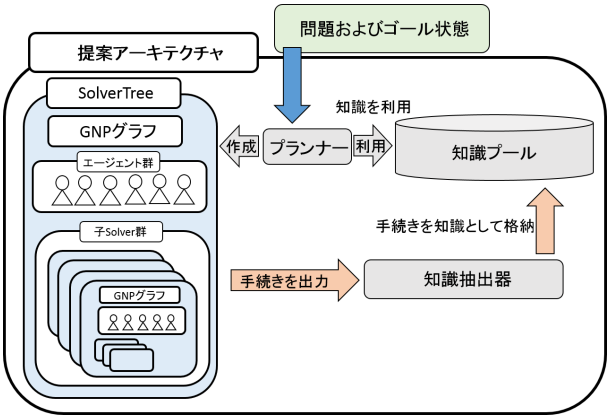

図1: アーキテクチャ概要

2.3 知識抽出器

知識抽出器はエージェントが問題を解く過程で得られた情報をマイニングする．この情報はゴールを達成するための手続きとなっているのでそれらを知識化し，知識プールに格納する．

2.4 知識プール

知識プールにはプランナーが部分ゴール状態群を作る際に利用する知識が格納されている．本アーキテクチャはこの知識プールをアーキテクチャ自身が自律的に拡張していくことでより複雑な問題を解決することを目標としている．

3 性能比較実験

3.1 実験概要

本稿で提案したアーキテクチャの有効性を確認するため，シミュレーションによる比較実験を行った．知識利用を行う本アーキテクチャの比較手法としてGNPのみを用いた手法を用意し，実験環境として迷路問題を用いて2通りの実験を行った．一つ目の実験は知識が利用できる場合とできない場合との性能比較である．二つ目の実験は環境が変化する場合における実験である．両実験の共通項として独立50試行で実験を行い，エージェントはSをスタートとし，Kの部分にある鍵を取得し，Gにあるゴールの前に存在するドアを開かなくてはゴールができない．また，今回は知識の利用の可否による比較を行うため，提案アーキテクチャは迷路1において鍵を取得し，ドアを開きゴールするという知識をすでに自律的に獲得し，知識プールを拡張しているものとする．図2および図3に本実験で用いた迷路を示す．

3.2 知識利用のオーバーヘッドについて

提案手法では，GNPのみの手法と比較してSolverTreeを作成する際のオーバーヘッドが存在する．そこで，オーバーヘッドが与える時間的影響について調べるために両手法の実時間比較を行った．両手法ともに後述する実験パラメータを用い，後述の知識

---

*Agent Architecture That Self-accuriers And Applies Knowledge With Using GNP, Shingo MORITA†, and Shohei KATO†

†Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan
{morita, shohey}@katolab. nitech.ac.jp

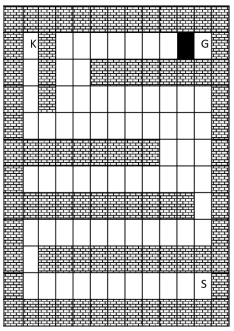

図 2: 迷路 1

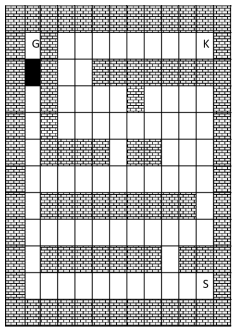

図 3: 迷路 2

S スタート
G ゴール
K 鍵
ドア
壁
通路

表 1: 実時間比較 (独立 50 試行平均)

	提案手法	GNP のみ
最小時間（ms）	40,318	55,124
最大時間（ms）	53,951	74,682
平均時間（ms）	46,201	60,288

利用による性能差を検証した実験と同設定で同一計算機を用いて独立 50 試行実験を行った．表 1 に実験結果を示す．このことから提案手法においては GNP のみの手法よりも計算時間が短くなっていることがわかる．これは，GNP のみの手法は一つの GNP グラフで問題を解決しようとするのでエージェントのゴール到達にかかる平均ステップ数が大きくなることが要因であると考える．一方，提案手法は問題を分割しているため GNP のみの手法よりもそれぞれの GNP グラフの達成に必要なステップ数が小さく，なおかつ部分ゴールの達成は容易であるため，エージェントが最大ステップ数まで探索せずともゴール状態を達成する可能性が高いことが要因であると考える．したがって提案手法は知識利用のオーバーヘッドを考慮してもより短い計算時間で問題を解決していることがわかる．

3.3 実験設定

表 2 に実験で用いたパラメータ設定を示す．実験で用いた GNP 処理ノードの種類は前進，右 90 °回転，左 90 °回転，現在のマスに落ちているアイテムの取得の 4 種類となっている．一方，GNP 判定ノードは前方 1 マスの区別の 1 種類となっており，図 2，図 3 右に示した 6 状態の判定を行う．エージェントの適応度はゴール個体，ゴールできなかった個体の順に以下のように設定した．

$$\text{適応度} = 300 - \text{ゴールするのに要したステップ数} - 1$$

また本稿は試行数を 2 手法間で統一するために，GNP のみの手法の進化方法に合わせて提案手法は 300 世代毎に最も適応度の低い GNP グラフを持つ Solver を一つ選択して進化を行う．

3.4 知識利用による性能差

一つ目の実験では単純な GNP と知識利用ができる本アーキテクチャとで収束速度や解の質についての比較を行う．図 4 に知識利用性能実験の結果を示す．横軸は世代，縦軸は適応度となっている．提案手法の方が GNP のみの手法に比べて解が早い段階で収束し，さらにより良い適応度を獲得していることがわかる．これは，提案手法は知識を利用してサブゴール群を作成し，問題を部分的に分割して進化させるので，GNP のみの手法のようにグラフ全体を進化させるよりも進化が容易であることが要因であると考える．

3.5 環境変化による影響

二つ目の実験では単純な GNP と知識利用ができる本アーキテクチャとで環境が変化した際の比較を行う．実験に用いるパラメータは一つ目の実験と同様であるが，この実験では世代数が 1500 世代に到達した際に迷路が 1 から 2 へと変化する．スタート位置は変化しないが，それぞれの作りあげたゴールを達成するためのグラフが変化後の環境では通用しないので新たにグラフを構築し直さなければならない．図 5 に実験結果を示す．知識利用性能実験と同様に提案手法の方が GNP のみの手法に比べて良い性能を示していることがわかる．さらに環境が変化した直後においてはその差が顕著に現れていることがわかる．これは GNP のみの手法は新たに迷路 2 を解く GNP グラフを作成しなければならないのに対し，提案手法では「鍵の位置まで移動した後に鍵を取得する」,「ドアまで移動した後にドアを開く」といった環境変化の前後で共通する GNP グラフは再利用をすることで新たに作成する必要がないためであると考える．

表 2: パラメータ設定

エージェント数	300
世代数	3000
ステップ数	300
親個体選択方法	トーナメント選択（サイズ 6）
交叉確率	0.1
突然変異確率	0.01

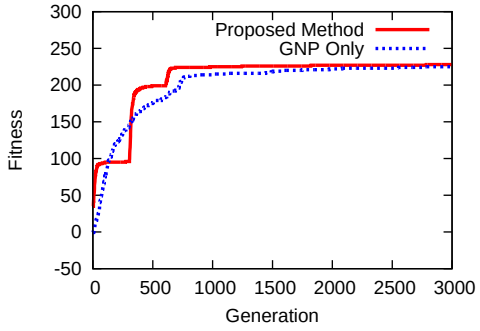

図 4: 知識利用性能実験 (独立 50 試行平均)

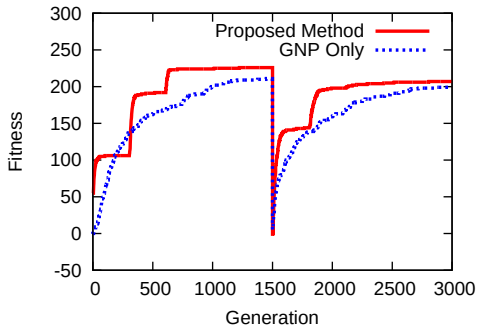

図 5: 環境変化実験 (独立 50 試行平均)

4 おわりに

本稿では，問題を解決する過程で得られた手続きを知識として獲得し，それを利用することでより環境の変化に頑強になるアーキテクチャを提案した．GNP との性能比較実験によって，提案アーキテクチャは知識利用することでより早く問題を解決し，また環境変化に対してより頑強であることを確認した．

現在プランナーは問題が知識利用により導出可能である場合にしか対応しておらず，導出が失敗した箇所が存在した場合，部分的に知識利用が可能な場合であるにもかかわらず SolverTree を作成できない．今後は導出過程において成功した箇所については知識利用を行い，導出に失敗した箇所は GNP を用いて知識獲得を行えるように改良を行いたい．

参考文献

[1] Fikes, R. E. and Nilsson, N. J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, in *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pp. 608–620 (1971).

[2] 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).