The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

階層型強化学習 RGoal アーキテクチャへの再帰呼び出し用スタックの導入

Introducing a Call Stack into the RGoal Hierarchical Reinforcement Learning Architecture

一杉裕志 *1 Yuuji Ichisugi　高橋直人 *1 Naoto Takahashi　中田秀基 *1 Hidemoto Nakada　佐野崇 *2 Takashi Sano

*1産業技術総合研究所 人工知能研究センター
National Institute of Advanced Industrial Science and Technology (AIST), AIRC

*2成蹊大学 理工学部 情報科学科
Department of Computer and Information Science, Faculty of Science and Technology, Seikei University

Humans can set suitable subgoals in order to achieve some purposes, and furthermore, can set sub-subgoals recursively if needed. It seems that the depth of the recursion is unlimited. Inspired by this behavior, we had designed a hierarchical reinforcement learning architecture, the RGoal architecture. In this paper, we introduce a call stack into the RGoal architecture to increase reusability of subgoals. We evaluate its performance using a maze with multi-task setting. The result shows that the convergence speed improves as the maximum stack size increases.

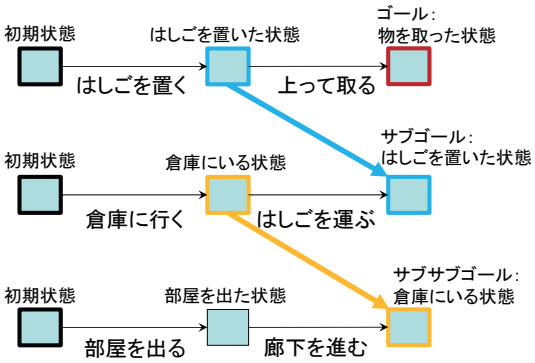

図 1: 人間が目標を達成するために、サブゴール（副目標）を再帰的に設定する例。高いところにあるものを取る（ゴール）ために、まずはしごを取る必要がある（サブゴール）。さらにはしごを手に入れるために倉庫にいく必要がある（サブサブゴール）。

1. はじめに

人間は何か目標を達成するために適切なサブゴール（副目標）を設定できる。さらに必要に応じてサブサブゴールを再帰的に設定することもでき、その再帰の深さには制約がないように見える（図 1）。この振る舞いにヒントを得た階層型強化学習のアーキテクチャとして、我々は RGoal アーキテクチャを提案した [一杉 18a]。我々はこの RGoal の機能を拡張していき、ヒトの脳の前頭前野周辺の情報処理を再現し、やがては汎用人工知能を実現するための中核技術とすることを目指している。

先行研究の 1 つ MAXQ [Dietterich 00] は、多層（ただし深さは固定）の階層型強化学習アーキテクチャであり、以下の 3 つの特徴により学習を効率化している。

1. Subtask sharing: マルチタスク環境において、サブルーチンをタスク間で共有することで学習を速くする。
2. Temporal abstraction: 複雑なタスクの学習において、より単純なサブルーチンの組み合わせ方のみに試行錯誤を限定することで、学習を速くする。
3. State abstraction: サブルーチンごとに実行に差し支えない程度に状態を抽象化することで、サブルーチンの学習を速くする。

RGoal では 1. は価値関数の分解と共有 (2.2 節)、 2. は思考モードと呼ぶ機構 (2.4 節) により実現する。 3. は現在のところ未実装だが、単一化の機構を利用したテーブル圧縮手法を検討している [一杉 18b]。

RGoal では、エージェントによるサブゴール設定は、プログラミング言語におけるサブルーチン呼び出しと似た振る舞いをする。実際、 RGoal は「強化学習によるプログラム合成システム」と見なすことができる [一杉 18b]。強化学習によるプログラム合成は汎用人工知能実現に向けた有望なアプローチの 1 つである。例えば、AIXI[Hutter 00] はチューリングマシンのプログラムを、UCAI[Katayama 18] はより一般的な文法を持った言語のプログラムを強化学習で合成する汎用人工知能の理論である。また、このアプローチで動作するシステムとして、DNC(Differentiable Neural Computers)[Graves 16] や MagicHaskeller [Katayama 08] がある。

以前提案した RGoal では、エージェントはスタックを持っておらず、サブゴール到達後には、それまでのコンテキストとは無関係にあらためてもともとのゴールを目指すように設計されていた。しかし、そのようなアーキテクチャでは、サブルーチンとして獲得された知識の再利用性が悪くなる [一杉 18b]。脳の中に頑健に動作するスタックがあるとは考えにくいが、それを代替する何らかの機構があることは考えられる。そこで本稿では、 RGoal にサブゴールを保存するスタックを導入し、それに伴い学習則も修正する。そして、迷路課題を用いて性能を評価する。また、 RGoal には思考モードとよぶ、演繹推論を行うための特殊なモデルベース強化学習の機構があるが、その性能も以前よりも詳しく評価する。

連絡先: 一杉裕志、 茨城県つくば市梅園 1 − 1 − 1 中央第 1 産業技術総合研究所、 y-ichisugi@aist.go.jp

2. RGoal アーキテクチャ

2.1 アーキテクチャの概要

RGoal アーキテクチャ[一杉 18a] について簡単に説明する。

エージェントは各ステップごとに、プリミティブ行動を取るかサブルーチン呼び出しを行うかのどちらかを選択する。サブルーチンの間に上下関係はなく、すべてのサブルーチンは対等であり、相互再帰的に呼び出しが可能である。

ここでは**サブルーチン** g を、「任意の環境の状態からある 1 つの状態（サブゴール） g に向かう方策」と定義する。また、サブルーチン g を実行し続けることで、環境の状態はいつか必ず g に到達すると仮定する。

Option-Critic アーキテクチャ[Bacon 17] などではサブルーチンの終了時の環境の状態は一意に決まらず、終了条件は学習によって獲得される。一方 RGoal では、終了条件はセンサー入力から得られる外界の特徴量の 1 つだと考える。つまり、終了条件は強化学習アーキテクチャの本体によってではなく、特徴抽出の機構によって獲得されるものと想定している。

2.2 価値関数分解

行動価値関数をサブルーチンごとに分解し、サブルーチンを複数のタスク間で共有することで、学習速度を上げることができる [Dietterich 00]。

ゴール G 、環境の状態 s 、サブゴール g としたとき、行動 a を取った後、方策 $\pi((s,g),a)$ に従って行動し続けたときに得られる報酬の総和の期待値を、行動価値関数 $Q_G^\pi((s,g),a)$ と定義する。（報酬割引は行わないものとする。）これは以下のように、g への到着前と到着後に分解することができる。

$$Q_G^\pi((s,g),a) = Q^\pi(s,g,a) + V_G^\pi(g) \tag{1}$$

ここで、 $Q^\pi(s,g,a)$ は状態 s において行動 a を取った後、方策 π に従って行動しサブゴール g に到着するまでの報酬の総和の期待値である。また、 $V_G^\pi(g)$ は、サブゴール g に到着後に方策 π に従って行動しゴール G に到着するまでの報酬の総和の期待値で、

$$V_G^\pi(g) \quad = \quad \Sigma_a \pi((g,G),a) Q^\pi(g,G,a) \tag{2}$$

として計算できる。

$Q^\pi(s,g,a)$ はもともとのゴール G に依存しないため、複数のタスク間で共有することができる。

なお、行動選択の式は通常の強化学習のもの同様であり、例えばテーブル Q のもとでグリーディーに行動を選択する場合は以下のようにする。

$$a' \quad = \quad \underset{a}{\operatorname{argmax}}\, Q(s,g,a) \tag{3}$$

2.3 スタックを導入した場合の学習則

ここまでの定義はスタックがない RGoal [一杉 18a] と本質的に違いはない（記法は一部変更した）が、学習則はスタックの導入に伴い、少し変更が必要となる。

スタックを導入した RGoal では、サブルーチン g' を呼び出した時に現在のサブゴール g をスタックに積み、サブルーチンの実行が終了した時、すなわち状態が s から g' に変化した時に、スタックから取り出した g をサブゴールに再設定する。サブルーチン呼び出し前は、想定される状態の変化は $s \to g \to G$ であるが、呼び出しによって $s \to g' \to g \to G$ に変化する。したがって以下の式が成り立つ。

$$\begin{aligned} & Q_G((s',g'),a') - Q_G((s,g),a) \\ = \; & (Q(s',g',a') + V_g(g') + V_G(g)) - (Q(s,g,a) + V_G(g)) \\ = \; & Q(s',g',a') - Q(s,g,a) + V_g(g') \end{aligned} \tag{4}$$

ただし、 s', a' は呼び出しの次のステップでの状態と行動である。この式は、行動が a がサブルーチン呼び出しに限らず、プリミティブ行動の場合も成り立つ。したがって、Sarsa で学習する場合の Q の更新式は以下のようになる。

$$\begin{aligned} & Q(s,g,a) \leftarrow Q(s,g,a) \\ & +\alpha(r + Q(s',g',a') - Q(s,g,a) + V_g(g')) \end{aligned} \tag{5}$$

この学習則には以前のもの [一杉 18a] と違い、もともとのゴール G が現れておらず、大域的な文脈に依存しない知識が獲得しやすくなっている。

2.4 思考モード

階層型強化学習には、学習済みの簡単なタスクを組み合わせて複雑なタスクを近似的に、しかし高速に解くという目的もある。RGoal アーキテクチャには、そのための機構として思考モードがある。思考モードは、学習済みの $Q(s,g,a)$ を環境のモデルと見なした一種のモデルベース強化学習 [Sutton 90] である。

本稿では [一杉 18a] とは違い、評価中の各エピソードの実行の直前に、思考モードにおける実行を定数回行うように実装を変更し、評価を行った。この振る舞いは、ヒトや動物が新たな状況に直面した時、知識を組み合わせて問題解決の行動計画を立ててから実行する振る舞いと似たものになっている。

2.5 サブルーチンの中断

以前の Roal アーキテクチャ[一杉 18a] には、実行中のサブルーチンを中断し、別のサブゴールに切り替える機能がある。サブルーチンの中断が可能であれば行動の自由度が増し、性能が上がる可能性がある [Kaelbling 93][Sutton 99][Dietterich 00] が、今回の実装ではアーキテクチャを簡単にするため、中断の機能を取り除いた。

2.6 アルゴリズム

以上の結果をまとめた、Sarsa に基づくアルゴリズムの疑似コードを図 2 に示す。

3. 評価

今回もアルゴリズムの基本動作の確認を行うことが目的のため、実行中の振る舞いの可視化が容易な迷路タスクを題材として性能を評価した。

ここでは、時間をかけた厳密解への収束ではなく、準最適解にできるだけ速やかに収束することを重視する。

マップとランドマークの集合は固定である（図 3）。ランドマークの中からエピソードごとにスタート S とゴール G がランダムに選ばれる。エージェントが S から移動して G に到達したときに与えられる報酬は 0 で、その時点でそのエピソードを終了し、スタートとゴールを変えて次のエピソードを始める。上下左右の移動は -1 、斜めの 4 方向いずれかへの移動は $-\sqrt{2}$ 、壁への衝突は -1 、サブルーチン呼び出しの実行は $R^C = -1$ の報酬が与えられる。（前に述べたように報酬割引はない。）ランドマークはサブゴールの候補でもあり、ランドマークの中の 1 つだけがある時点でのサブゴールになり得る。

```
1: procedure EPISODE(S, G, think-flag)
2:     s ← S; g ← G
3:     stack ← empty
4:     Choose a from s, g using policy derived from Q
5:     while s ≠ G do
6:         # Take action.
7:         if a = RET then
8:             s' ← s; g' ← stack.pop(); r ← 0
9:         else if a is C_m then
10:            stack.push(g)
11:            s' ← s; g' ← m; r ← R^C
12:        else
13:            if think-flag then
14:                s' ← g; g' ← g; r ← Q(s, g, a)
15:            else
16:                Take action a, observe r, s'
17:                g' ← g
18:        # Choose action.
19:        if s' = g' then
20:            a' ← RET
21:        else
22:            Choose a' from s', g'
23:                using policy derived from Q
24:        # Update.
25:        if s = g or (think-flag and a is not C_m) then
26:            # Do nothing.
27:        else
28:            Q(s, g, a) ← Q(s, g, a)
29:              +α(r + Q(s', g', a') − Q(s, g, a) + V_g(g'))
30:        s ← s'; g ← g'; a ← a'
```

図 2: 1 つのエピソードを実行する Sarsa に基づくアルゴリズムの疑似コード。テーブル Q の初期化方法については [一杉 18a] を参照。

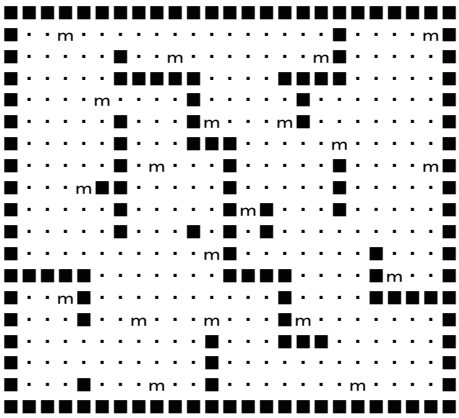

図 3: 評価に用いた 2 次元格子上の迷路のマップ。20 個のランドマーク (m で示した) がマップ上に配置されている。このランドマークの中からエピソードごとにランダムにスタート S とゴール G が選択される。

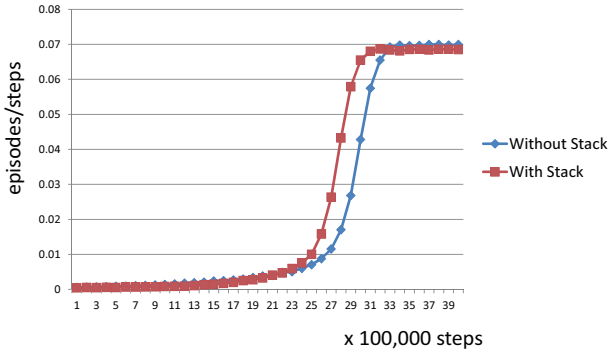

図 4: 実験 1 ：スタックのない RGoal [一杉 18a] と、スタックを導入した本稿のアルゴリズムとの比較。実験条件にもよるが、同程度の早さで収束している。

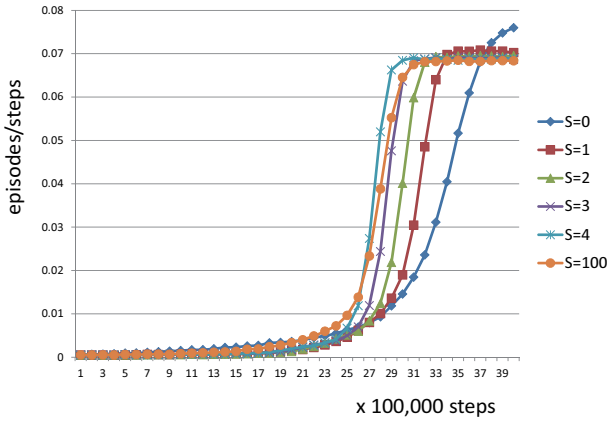

図 5: 実験 2 ：スタックの深さの上限 S と性能の関係。S=0 は普通の強化学習、S=1 は 2 層の階層型強化学習に相当する。上限が大きいほど収束は早くなる傾向を示している。

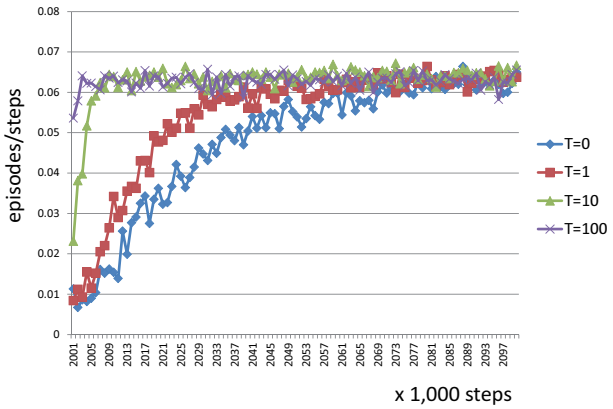

図 6: 実験 3 ：各ステップの実行直前に実行する思考フェーズの長さと性能の関係。思考フェーズの長さ T は、各エピソードの直前に思考モードで「脳内シミュレーション」するエピソードの回数。思考フェーズが十分長ければ、未経験のタスクであっても、近似解がゼロショットで得られている。

テーブルの初期値は、$s = g$ での $Q(s, g, a)$ に対しては 0 、それ以外は $-50 - n$ (n は小さなノイズ) に初期化した。また、学習を効率化するため、サブルーチン呼び出しはランドマーク上でのみ実行可能とした。

行動選択は softmax を用い、逆温度 $\beta = 1$ とした。学習率は $\alpha = 0.1$ である。

実験 1 、 2 、 3 はいずれも 1 0 回の実行結果の平均である。横軸はステップ数である。縦軸はステップ数あたりのエピソード数であり、値が大きいほど、各エピソードを短いステップで解けるということを示している。ここでステップ数とはマップ内の移動もしくは壁への衝突の回数である。サブルーチン呼び出し C_m 、サブルーチンからの復帰 RET の実行回数、思考モードでの実行ステップは、いずれもエージェントの内部で起こる仮想的な行動であると考え、ステップ数には含めない。

実験 1 （図 4）はスタックがないがサブルーチン中断の機能を持ったアルゴリズム [一杉 18a] と、スタックを導入した本稿のアルゴリズムとの比較である。どちらも同程度の早さで収束している。この実験条件では、スタックありの方が、サブルーチン中断の機能がないにもかかわらず、わずかに収束が早くなっている。ただし、実験条件によって結果は変わり、[一杉 18a] で用いた迷路の場合では、サブルーチン中断が起きやすく、本稿のアルゴリズムの方が収束はわずかに遅くなる。

実験 2 （図 5）はスタックの深さの上限 S と性能の関係を調べたものである。S=0 は普通の強化学習、S=1 は 2 層の階層型強化学習に相当する。実験結果は、 S が大きいほど収束が早くなる傾向を示している。（ただし S=100 では S=4 より収束が若干遅くなってる。）一方で、収束後のスコアは、サブルーチン呼び出しを一切行わない S=0 がもっともよくなっている。これは、サブルーチンが使える場合、ランドマークを経由して多少遠回りする局所解にとどまってしまうためである。なお、サブルーチンが使える場合でも、逆温度 β を小さくし探索傾向を強くして十分学習させた後、β を大きくすることで、遠回りしないほぼ最短経路に収束することを確認している。

実験 3 （図 6）は各エピソードの実行直前の思考フェーズの長さ T と性能の関係を調べたものである。ここで T は、各エピソードの直前に、与えられた S と G のもとで思考モードで実行するエピソードの回数である。評価の前に簡単なタスクのみを用いた事前学習フェーズを 2,000,000 ステップ実行し、そのあとの 1,000 ステップごとのスコアをプロットしている。事前学習フェーズでは、ユークリッド距離が 8 以内にある S と G のみを選択して実行する。これは、すべてのランドマークのペア 20x19=380 個のうちの 60 個である。短い距離にあるランドマーク間の移動方法が 事前学習フェーズで獲得され、長い距離の移動方法の近似解は、それをつなぐことで得られる。思考フェーズでは、そのような近似解が、実際に行動することなく、脳内シミュレーションだけで「演繹的」に獲得される。実験結果は、思考フェーズが十分に長ければ、経験したことのない未知のタスクであっても、それまでの経験で得た知識を組み合わせることで、近似解がゼロショットで得られることを示している。

4. まとめと今後

階層型強化学習アーキテクチャ RGoal にスタックを導入したアルゴリズムを実装し、評価した。今後このアーキテクチャを拡張し、状態抽象の機構を追加するなどした上で、より複雑なタスクでの評価を行っていく。

謝辞

ディー・エヌ・エー 甲野佑氏、東京電機大 高橋達二氏との議論から研究の示唆をいただいており、深く感謝いたします。

本研究は JSPS 科研費 JP18K11488 の助成を受けたものです。

参考文献

[一杉 18a] 一杉裕志, 高橋直人, 中田秀基, 佐野崇, RGoal Architecture:再帰的にサブゴールを設定できる階層型強化学習アーキテクチャ, 第 9 回 人工知能学会 汎用人工知能研究会 (SIG-AGI), 2018.

[一杉 18b] 一杉裕志, 高橋直人, 中田秀基, 佐野崇, 単一化の機構を利用した階層型強化学習のテーブル圧縮手法の検討, 第 10 回 人工知能学会 汎用人工知能研究会 (SIG-AGI), 2018.

[Dietterich 00] Thomas G. Dietterich, Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition, Journal of Artificial Intelligence Research 13, 227–303, 2000.

[Hutter 00] Hutter, M., A theory of universal artificial intelligence based on algorithmic complexity, Technical Report: cs.AI/0004001, 2000.
http://arxiv.org/abs/cs.AI/0004001

[Katayama 08] S. Katayama, Efficient Exhaustive Generation of Functional Programs using Monte-Carlo Search with Iterative Deepening , PRICAI 2008 , LNAI 5351, Springer Verlag, 199-211, 2008.

[Graves 16] A. Graves, G. Wayne et al., Hybrid computing using a neural network with dynamic external memory, Nature 538, 471–476, 2016.

[Katayama 18] Susumu Katayama, Computable Variants of AIXI which are More Powerful than AIXItl, 2018.
https://arxiv.org/abs/1805.08592

[Sutton 90] Sutton, R. S., Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Proceedings of the Seventh International Conference on Machine Learning, 1990.

[Kaelbling 93] Kaelbling, L.P.: Hierarchical Learning in Stochastic Domains: Preliminary Results. In: Proceedings of the 10th International Conference on Machine Learning, pp. 167–173, 1993.

[Sutton 99] Sutton, R. S.; Precup, D.; and Singh, S. P., Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence 112(1-2):181–211, 1999.

[Bacon 17] Bacon, P.-L., Harb, J., Precup, D. The option-critic architecture. Proceedings of AAAI, 1726–1734, 2017.