Evolving Meta-Level Reasoning with Reinforcement Learning and A* for Coordinated Multi-Agent Path-planning

Extended Abstract

Mona Alshehri
Imam Abdulrahman Bin Faisal University, Saudi Arabia
Massey University, New Zealand
Maalshehri@iau.edu.sa

Napoleon Reyes
Massey University, New Zealand
N.H.Reyes@massey.ac.nz

Andre Barczak
Massey University, New Zealand
A.L.Barczak@massey.ac.nz

ABSTRACT

This work presents an extension to a graph-based evolutionary algorithm, called Genetic Network Programming with Reinforcement Learning (GNP-RL) to make it more amenable for solving coordinated multi-agent path-planning tasks in dynamic environments. We improve the algorithm's ability to evolve meta-level reasoning strategies in three aspects: genetic composition, search and learning strategies, using optimal search algorithm, constraint conformance and task prioritization techniques.

KEYWORDS

Evolutionary algorithms; Learning agent capabilities; Multi-agent learning; Reinforcement learning; Optimal Search

ACM Reference Format:
Mona Alshehri, Napoleon Reyes, and Andre Barczak. 2020. Evolving Meta-Level Reasoning with Reinforcement Learning and A* for Coordinated Multi-Agent Path-planning. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 3 pages.

1 INTRODUCTION

In 2000, a new algorithm emerged from Genetic Programming, called genetic network programming (GNP) [2]. This algorithm uses a concise graph structure to represent the chromosomes. Each graph has a number of nodes including one start node, processing (action) nodes, and judgment (query) nodes, but without a terminal node. Each node could be visited more than once, and from more than one agent (Fig.2). In 2007, reinforcement learning (RL) using the Sarsa algorithm [9] was added to the GNP algorithm, allowing the algorithm to solve more complex problems. It allowed for the incorporation of several sub-nodes within each computational node and run the RL to learn which subnode is the best. This algorithm, called (GNP-RL) [5], has achieved better results than previous genetic-based architectures in solving many problems such as inculcating wall following behaviour in robots [4], stock trading modelling [11], mobile robot navigation [8], and the tile world problem [5]. Other algorithms may also be incorporated as a node in GNP-RL, such as fuzzy logic in [10] and [8].

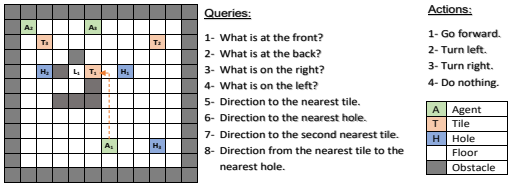

Figure 1: Challenges of evolving a meta-level reasoning strategy

2 CHALLENGES OF EVOLVING A META-LEVEL REASONING STRATEGY

A meta-Level reasoning strategy is evolved in the form of a graph, comprising of computational nodes (Fig. 2) The graph is evolved using the mechanisms of GNP, while RL refines the graph solutions, as the agent interacts with a dynamic environment. A* on the other hand is used in a variety of judgement (query) nodes.

In order to illustrate the significance of evolving a meta-level reasoning strategy, we have (Fig. 1) demonstrating one scenario in the Tile world problem [7] : an Agent (A1) is supposed to push the nearest tile (T1), into the nearest Hole (H1). Using A*, it can be seen that the optimal path between the agent and T1 is the one that eventually leads towards the right-hand side of tile T1. However, in order to accomplish the task, the Agent should actually push the tile T1 to the right towards hole H1. Apart from identifying the optimal paths, path-planning for this problem also needs to take into account the trap locations (e.g. L1 – if the Tile lands here, it will get trapped as the robot is not allowed to pull any of the tiles), as well as Agent (A3) who is in contention with Agent (A1) for tile T1. All these combinations of constraints checking, optimal path-planning, sensor querying (judgements) and taking actions needs to be logically and hierarchically arranged by the GNP-RL algorithm, requiring some levels of abstraction, in order to formulate a graph solution that generalises to different arrangements of the world.

3 GNP-RL WITH A* FRAMEWORK

The GNP works in tandem with the RL algorithm to find the best solution, represented as a graph. The algorithm starts with a random population of chromosomes (solutions), and each chromosome is composed of a number of nodes (genes). In turn, each node is comprised of a maximum number of sub-nodes, possessing three main characteristics: (Q-value for choosing the subnode, sub-node ID to identify the associated function, and connections that direct to the next nodes). In this work, we modified the chromosome structure presented in [5], in order to detect and handle conflicting

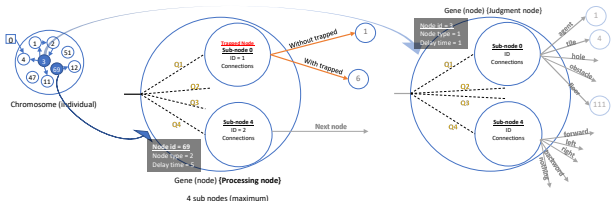

Figure 2: Chromosome Structure

situations (see Fig.2a). The new structure makes use of two output pathways: one if conflict was detected, and another otherwise.

Before running the GNP-RL, firstly, identify all queries and actions (Fig. 1) that may be utilised by the agent, as well as all the conflicting situations that may occur in the problem domain, then incorporate the appropriate functions (into the library of functions for which GNP-RL selects from) that implement them. Secondly, apply the optimal search algorithm to order the tasks required for each agent, allowing for the distribution of the tasks to each of the agents without requiring any communications between them. Thirdly, create N (e.g.300) individuals randomly as part of the initial population using the same techniques in [6].

The algorithm takes the following inputs, each time an agent visits a node in the graph: locations of agents L, set of actions A (processing functions), set of queries (judgment functions), set of constraints C (conflict situations), set of priorities P (the sorted tasks), set of rewards A (rewards for each <state,action/query>), set of states S, and set of objects O (contains all the objects in the environment). Algorithm 1 provides the steps for executing a node, utilising optimal search, constraint conformance, and task prioritization techniques to operate in a dynamic environment. The output of each node call directs the agent to the next node in the graph. The A* algorithm was utilised for identifying the nearest object, and the shortest path to this object.

$$Q_{ip} = Q_{ip} + (\alpha * (Reward + (\gamma * Q_{jq}) - Q_{ip})) \text{ [5]} \quad (1)$$

$$Fitness = \sum_{env=1}^{ENV} [(a * D_{tile}) + (b * D_{distance}) + (c * T_{remain})] \quad (2)$$

To evaluate a chromosome, the reinforcement learning is run on this chromosome for the three agents, on the tile world environments using equation 1 for updating the Q-value. Ten environments were used for each training and testing sets. The training set is the same training environment used in [6], while Testing set 1 is the same with [6]. We produced Testing set 2 to be similar in complexity with Li et al's work [3]. Finally, after executing the chromosome on the training environments, the fitness function is calculated using equation 2, which was modified from [6]. The settings of the weights a,b, and c depends on the size of the environment (the full details is in [1]).

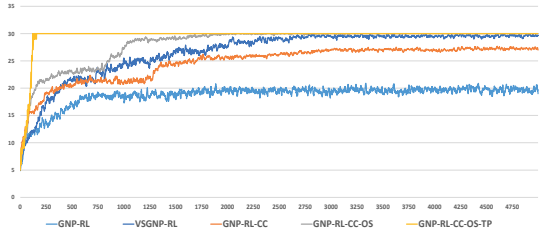

Figure 3: Training Results on Set 1

Crossover & Mutation: We employed crossover and mutation techniques similar to [5], but we relaxed the way crossover is performed; that is, not limiting the operation only to nodes with the same IDs.

Algorithm 1 General Node Execution

Inputs: current locations of Agents L, Set of actions A[], Set of queries Q[], Set of constraints C[], Set of priorities P[], Set of Rewards R[], Set of States S[], objects in the environment O[]
Output: nextNode
if NodeType is an Action Node **then**
  With probabilistic selection a in A[] based on Q-value
  if a causes any conflict c in C[] **then**
    $Reward \leftarrow$ -low reward from R[]
    $NextNode \leftarrow first\_connection$ ▷ deal with conflict
  else
    Apply action a, update state s
    if ((a,s) has priority p in P[]) **then**
      $r = R[p]$ ▷ includes both punishments and rewards
    end if
    $NextNode \leftarrow second\_connection$ ▷ without conflict
  end if
else if NodeType is a Query Node **then**
  With probabilistic selection q in Q[] based on Q-value
  if q is a directional query **then**
    path = A*(q,L,O) ▷ use the A* algorithm to answer q
    response = q(Direction(first_waypoint(path)))
  else
    response = q( )
  end if
  $NextNode \leftarrow connection(response)$
end if
Update Q-value(r)
Goto the $NextNode$

4 SUMMARY

Using training set (1), we trained and tested 5 different algorithms. Each of the experiments ran for 5000 generations (on both training & testing stages) for GNP-RL, VSGNP-RL, GNP-RL with constraint conformance (GNP-RL-CC), GNP-RL with Optimal search and constraint conformance (GNP-RL-CC-OS), and GNP-RL with task priority optimal search and constraint conformance (GNP-RL-CC-OS-TP). Fig.3 shows the dynamic training accuracy for the algorithms when running the RL with both exploration and exploitation modes on the GNP graph, in terms of the number of correctly dropped tiles for each generation. It is clear that GNP-RL-CC-OS-TP reached the top performance the earliest.

In the testing stage, the best chromosome was found in GNP-RL-CC-OS-TP, which was able to achieve 100% accuracy in training set (1), and (29/30) 96.66% of accuracy in the Testing set (1) as compared to a previous best result of 43.3% using (VS-DGNP) [6], and 29/30 (96.66%) in the Testing Set (2) vs. the best in the literature 63.3% [3].

REFERENCES

[1] Mona Alshehri. 2019. *Genetic network programming with reinforcement learning and optimal search component : a thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Computer Sciences at Massey University, Auckland, New Zealand.* http://ezproxy.massey.ac.nz/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat00245a&AN=massey.b4735499&site=eds-live&scope=site

[2] Hironobu Katagiri, Kotaro Hirasama, and Jinglu Hu. 2000. Genetic network programming - application to intelligent agents. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: "Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions", Sheraton Music City Hotel, Nashville, Tennessee, USA, 8-11 October 2000.* 3829–3834. https://doi.org/10.1109/ICSMC.2000.886607

[3] Xianneng Li, Meihua Yang, and Shizhe Wu. 2018. Niching genetic network programming with rule accumulation for decision making: An evolutionary rule-based approach. *Expert Syst. Appl.* 114 (2018), 374–387. https://doi.org/10.1016/j.eswa.2018.07.041

[4] Shingo Mabu, Hiroyuki Hatakeyama, Kotaro Hirasawa, and Jinglu Hu. 2006. Genetic Network Programming with Reinforcement Learning Using Sarsa Algorithm. In *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006.* 463–469. https://doi.org/10.1109/CEC.2006.1688346

[5] Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu. 2007. A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning. *Evolutionary Computation* 15, 3 (2007), 369–398. https://doi.org/10.1162/evco.2007.15.3.369

[6] Shingo Mabu, Kotaro Hirasawa, Masanao Obayashi, and Takashi Kuremoto. 2014. A variable size mechanism of distributed graph programs and its performance evaluation in agent control problems. *Expert Syst. Appl.* 41, 4 (2014), 1663–1671. https://doi.org/10.1016/j.eswa.2013.08.063

[7] Martha E. Pollack and Marc Ringuette. 1990. Introducing the Tileworld: Experimentally Evaluating Agent Architectures. In *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, USA, July 29 - August 3, 1990, 2 Volumes.* 183–189. http://www.aaai.org/Library/AAAI/1990/aaai90-028.php

[8] Siti Sendari, Shingo Mabu, Andre Tjahjadi, and Kotaro Hirasawa. 2011. Fuzzy Genetic Network Programming with Noises for Mobile Robot Navigation. *JACIII* 15, 7 (2011), 767–776. https://doi.org/10.20965/jaciii.2011.p0767

[9] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction.* MIT Press. http://www.worldcat.org/oclc/37293240

[10] Wenhan Wang. 2014. *Genetic network programming with fuzzy reinforcement learning nodes for multi-behaviour robot control : a thesis presented in partial fulfilment of the requirements for the degree of Masters of Science in Computer Science, Massey University, Albany campus, New Zealand.* http://ezproxy.massey.ac.nz/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat00245a&AN=massey.b3284741&site=eds-live&scope=site

[11] Yang Yang, Zhaoping He, Shingo Mabu, and Kotaro Hirasawa. 2012. A Cooperative Coevolutionary Stock Trading Model Using Genetic Network Programming-Sarsa. *JACIII* 16, 5 (2012), 581–590. https://doi.org/10.20965/jaciii.2012.p0581