Automatic Abstraction in Reinforcement Learning Using Ant System Algorithm

Mohsen Ghafoorian
Intelligent Systems Laboratory,
Sharif University of Technology
Index

• Reinforcement Learning
• Hierarchical Reinforcement Learning
• Proposed Method
  – Transition Graph Creation
  – Sub-goal Identification
  – Skill Creation
• Experimental Results
  – Taxi Environment
  – Playroom Environment
• References
Reinforcement Learning

- Action Selection based on agent policy:
  - Agent’s goal:
    - Maximizing cumulative discounted reward:
      \[ R = E \left\{ \sum_{t=0}^{\infty} \gamma^t r_t \right\} \]
      \[ \pi : S \times A \rightarrow [0, 1] \]
  - Approximation of Action-value function:
    \[ \hat{Q}(s, a) = (1 - \alpha)\hat{Q}(s, a) + \alpha \left[ R(s, a) + \gamma \max_{a' \in A} \hat{Q}(s', a') \right] \]
Reinforcement Learning

Q Learning Algorithm:

Input: \((\alpha, \gamma)\)

1. Initialize \(\hat{Q}(s, a)\) randomly
2. Repeat for each episode
   1. Indicate policy \(\pi\) using \(\hat{Q}\)
   2. Initialize \(s\)
   3. Repeat
      1. Choose action \(a\) considering policy \(\pi\)
      2. Do action \(a\) and observe reward \(R(s, a)\) and next state \(s'\)
      \[\hat{Q}(s, a) \leftarrow (1 - \alpha)\hat{Q}(s, a) + \alpha \left[ R(s, a) + \gamma \max_{a' \in A} \hat{Q}(s', a') \right]\]
3. Until \(s\) is a final state.
Reinforcement Learning

Back gammon Environment:

- Number of states: about $10^{20}$
- Number of learning episodes until convergence: 1.5 millions!
Hierarchical Reinforcement Learning

In enormous environments, use abstraction:
  – State Abstraction
  – Temporal Abstraction
Option Framework:

- Formal description of macro actions.
- Option is a sorted tuple: \( o = (I, \pi, \beta) \)
  - \( I \): Set of states that \( o \) is permitted on.
  - \( \pi \): Agent’s policy, \( \pi: S \times A \rightarrow [0, 1] \)
  - \( \beta(s) \): Function to indicate episode finishing.

- Primitive actions as options
  - \( I \): the state action is permitted on.
  - \( \pi(s, a) = 1 \)
  - \( \beta(s) = 1 \)
Ant Colony Optimization

• Ant System
  – Find Shortest path from s to t
  – $n_t$ episodes
  – $n_k$ ants
  – Stochastic path creation based on pheromone values.
Ant Colony Optimization – Ant System

– Path generation

\[ p_{ij}^k(t) = \begin{cases} \frac{\alpha \tau_{ij}(t) + (1 - \alpha) \eta_{ij}(t)}{\sum_{j \in N_i^k(t)} \alpha \tau_{ij}(t) + (1 - \alpha) \eta_{ij}(t)} & \text{if } j \in N_i^k(t) \\ 0 & \text{if } j \notin N_i^k(t) \end{cases} \]

– Evaporation:

\[ \tau_{ij}(t + 1) = (1 - \rho) \tau_{ij}(t) \]

- Pheromone deposit:

\[ \Delta \tau_{ij}^k(t) \propto \frac{1}{L^k(t)} \]

– Use taboo list.
Proposed Method – Bird’s Eye View

1- Transition Graph Generation
- Interaction in env
- Interaction history stored in weighted directed graph

2- Sub-goal Identification
- Execution of ant system
- Roughness consideration of edges on shortest path
- Selection of edges with lower amounts of Roughness
- Deletion of consecutive bottleneck edges

3- Skill Generation
- Shortest path segmentation
- Identification of communities on the shortest path
- Acquiring of optimal policies in each community
Proposed Method- Sub-goal Identification

Execution of ant system on transition graph

- Regular participation of $u$ in generated shortest paths
- Irregular participation of $v$ in generated shortest paths
Proposed Method- Sub-goal Identification

- Comparison of pheromone values of u and v during ant system

Variation of pheromone values of v in time

Variation of pheromone values of u in time
Proposed Method- Sub-goal Identification

Proposed Criteria for separation of these edges, called Roughness:

\[ - R_F = \frac{\sigma_M^2}{(\max_i F_i - \min_i F_i)^2} \]

Where \( M_i = F_{i+1} - F_i \)
Proposed Method- Sub-goal Identification

• Sorting shortest path edges based on pheromone values

Edge roughness values based on edge pheromone Roughness rank
Proposed Method- Sub-goal Identification

• Separation of bottleneck edges:
  – Using threshold values:
    • For pheromone values: $\tau_v$
    • For pheromone increase slope: $\tau_d$

  $b$ is the rank border between bottleneck and non-bottleneck edges iff:

  \[
  \text{Fail}(b) = \text{true and } \forall i < b: \text{Fail}(i) = \text{false}
  \]

  \[
  \text{Fail}(i) = (d_i > \tau_d \cdot d_{\text{init}} \text{ or } v_i > \tau_v \cdot v_0)
  \]
Proposed Method- Sub-goal Identification

• Removing consecutive bottleneck edges.
• Getting vertices on bottleneck edges as final sub-goals.
Proposed Method- Sub-goal Identification

- Sub-goal discovery algorithm

Algorithm 1: The proposed method for sub-goal detection

1. **Input:** \((n_k, t_d, \alpha, \rho, \tau_d, \tau_v)\)
2. **Output:** SubGoals: a list of sub-goals
3. Run Ant System \((\tau_d, \alpha, \rho)\) and have \(SP\) with shortest path.
4. Sort \(SP\) increasingly according to field \(R_P\).
5. \(v_0 \leftarrow SP[0].R_P\)
6. for \(i \leftarrow 1\) to length\((SP)\) do
7. \(d_{init} \leftarrow SP[i].R_P - SP[i-1].R_P\)
8. if \(d_{init} \neq 0\) then
9. exit the loop.
10. end if
11. end for
12. for \(b \leftarrow 1\) to length\((SP)\) do
13. if \((SP[b].R_P - SP[b-1].R_P > \tau_d.d_{init}) \lor (SP[b].R_P > v_0.\tau_v)\) then
14. exit the loop.
15. end if
16. end for
17. for Adjacent: adjacent set of edges in \(SP[0 \ldots b-1]\) do
18. \(best \leftarrow \arg\min_i \{Ad\text{~}jacent.\text{Edges}[i].R_P\}\)
20. end for
Proposed Method- Sub-goal Identification

- Incremental variance calculation:

\[ \sigma_n^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{X}_n)^2}{n} = \frac{y_n}{n} \]

\[ y_n = \sum_{i=1}^{n}(x_i - \bar{X}_n)^2 = \sum_{i=1}^{n}x_i^2 + n\bar{X}_n^2 - 2\bar{X}_n\sum_{i=1}^{n}x_i \]

\[ y_n = \sum_{i=1}^{n}x_i^2 + n\left(\frac{s_n}{n}\right)^2 - 2\frac{s_n}{n}s_n = \sum_{i=1}^{n}x_i^2 - \frac{s_n^2}{n} \]

\[ y_{n+1} = \sum_{i=1}^{n+1}x_i^2 - \frac{s_{n+1}^2}{n+1} \]

\[ y_{n+1} = y_n + x_{n+1}^2 - \left(\frac{s_{n+1}^2}{n+1} - \frac{s_n^2}{n}\right) \]
Environments

• Taxi Environment
  – Goal: take person to destination
  – Actions:
    • Movement in 4 directions
    • Take passenger in taxi
    • Take passenger out of taxi
  – Reward
    • +10: taking the passenger in taxi
    • +20: take passenger out in destination
    • -1: every other action
Environments

• Playroom environment
  – Goal: making monkey scream
  – Actions:
    • 1) look at a random object
    • 2) look at object at hand
    • 3) hold object it is looking at
    • 4) look at object marker is placed on
    • 5) place marker on object it is looking at
    • 6) move object in hand to location it is looking at
    • 7) turn over light switch
    • 8) press music button
    • 9) hit ball toward the marker.
  – Rewards:
    • +1000: reaching the goal
    • -1: every other action
Experimental Results

- Taxi environment

$$n_t = 10, n_k = 25, \alpha = 0.9, \rho = 0.98, \tau_v = 1.01, \tau_d = 1.5$$
Experimental Results

• Playroom Environment

\[ n_t = 200, n_k = 10, \alpha = 0.9, \rho = 0.98, \tau_v = 2.0, \tau_d = 1.5 \]
References


References


