IJCAI–18

Di Wu, Zhennan Wang, Wenbin Zou, Xia Li, Chen Xu
Shenzhen University*
dwu,...@szu.edu.cn

Abstract

The *IJCAI–18 Proceedings* will be printed from electronic manuscripts submitted by the authors. The electronic manuscript will also be included in the online version of the proceedings. This paper provides the style instructions.

1 Introduction

The *IJCAI–18 Proceedings* will be printed from electronic manuscripts submitted by the authors. These must be PDF (*Portable Document Format*) files formatted for 8-1/2″ × 11″ paper.

1.1 A* search algorithm

The following is an excerpt from Wikipedia `https://en.wikipedia.org/wiki/A*_search_algorithm`.

A* is an informed search algorithm, or a best-first search, meaning that it solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance travelled, shortest time, etc.). It is an extension of Edsger Dijkstra's 1959 algorithm. A* selects the path that minimses:

$$f(n) = g(n) + h(n) \tag{1}$$

where n is the last node on the path, $g(n)$ is the cost of the path from the start node to n, and $h(n)$ is a heuristic that estimates the cost of the cheapest path from n to the goal. The heuristic is problem-specific. For the algorithm to find the actual shortest path, the heuristic function must be admissible, meaning that it never overestimates the actual cost to get to the nearest goal node. Typical implementation of A* use a priority queue to perform the repeated selection of minimum (estimated) cost nodes to expand.

Notation	meaning
xid, yid	

Table 1: Results of 3D performance evaluation on mean coverage (higher is better) and center error (lower is better).

Configuration	train (prev 12-05)		test (prev 12-05)
	mins	crash	min
Straight Line	51318	31	69376
A* 2D	51318	31	65656
A* 3D(risky)	**43010**	**21**	65720
A* 3D(conservative)	48874	29	**62766** [1]

Table 2: Results of 3D performance evaluation on mean coverage (higher is better) and center error (lower is better).

2 dataset

2.1 real weather update

3 Experiments

3.1 Notations

3.2 Baselines

Straight lines

:

A* 2D

A* 3D: risky scheme

A* 3D: conservative scheme

References

*Shenzhen Key Lab of Advanced Telecommunication and Information Processing, College of Information Engineering, Shenzhen University.

Algorithm 1: 3D A* Double(I haven't implemented double yet) Expected Sarsa (Q-learning(which is better Q or ES?)), Dyna model planning with prioritized sweeping

Data:
$L^s = \{x^s, y^s\}$: 2D location of the starting city
$L^g = \{x^g, y^g\}$: 2D location of the goal city
$W_m = \{w_{x,y,t}\}$: Wind predictions of model $m \in \{1, 2, \dots, M\}$ (M is the number of models) with size as $X \times Y \times T$ where X, Y are the grid world size and T is the maximum allowed time

Generate M trajectories: J_m from 3D A* algorithm (conservative(?)) and their corresponding policies: π_m
Initialise action-value function $Q : X \times Y \times T \leftarrow 0$, $Model(s, a)$ and $PQueue$ to empty
Initialise starting states S_S from L^s as $(x^s, y^s, 0)$; goals states S_G from L_g as $(x^s, y^s, t), t \in (1, 2, \dots T)$ and terminal states S_T as $(x, y, T), x \in (1, 2, \dots X), y \in (1, 2, \dots Y)$
L is the total A* looping number(How to choose the number?), N is the number of planning steps, θ is the priority threshold, γ is the discount rate, α is the learning rate
for $l \leftarrow 1$ *to* L **do**
  $s_1 \leftarrow S_S$
  With probability $\frac{1}{m}$, uniformly select a random wind model number m_w from $\{1, 2, \dots, M\}$
  With probability $\frac{1}{m}$, uniformly select a random policy number m_{A*} from $\{1, 2, \dots, M\}$ (Should m_w, m_{A*} be the same?)
  for $t \leftarrow 1$ *to* T **do**
    $a_t \leftarrow \pi_{m_{A*}}$ (with probability ϵ randomly select an action, if s_t is not in $\pi_{m_{A*}}$, then action is selected greedily according to Q) (How to handle expected sarsa action selectoin?)
    Execute action a_t; observe reward r_t from wind model W_{m_w}, and state s_{t+1}
    $Model(s_t, a_t) \leftarrow r_t, s_{t+1}$
    $P \leftarrow |r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)|$
    $P \leftarrow |r_t + \gamma \sum_a \pi_\epsilon(a|s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t)|$
    if $P > \theta$, then insert s_t, a_t into $PQueue$ with priority P
    for $n \leftarrow 1$ *to* N *while PQueue is not empty* **do**
      $s_n, a_n \leftarrow first(PQueue)$
      $r_n, s_{n+1} \leftarrow Model(s_n, a_n)$
      $Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha[r_n + \gamma \max_a Q(s_{n+1}, a) - Q(s_n, a_n)]$
      $Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha[r_n + \gamma \sum_a \pi_\epsilon(a|s_{n+1}) Q(s_{n+1}, a) - Q(s_n, a_n)]$ (here, π_ϵ is ϵ soft policy)
      for *all* $\bar{s}, \bar{a}$ *predicted lead to* s_n **do**
        $\bar{r} \leftarrow$ predicted reward for $\bar{s}, \bar{a}, s_n$
        $P \leftarrow |\bar{r} + \gamma \max_a Q(s_n, a) - Q(\bar{s}, \bar{a})|$
        $P \leftarrow |\bar{r} + \gamma \sum_a \pi_\epsilon(a|s_n) Q(s_n, a) - Q(\bar{s}, \bar{a})|$
        if $P > \theta$, then insert $\bar{s}, \bar{a}$ into $PQueue$ with priority P
    if s_{t+1} is in terminal states S_T or goal states S_G, break

Result:
Trajectory J from the updated Q using greedy action selection