Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

State-chain sequential feedback reinforcement learning for path planning of autonomous mobile robots*

Xin MA†, Ya XU, Guo-qiang SUN, Li-xia DENG, Yi-bin LI

(*School of Control Science and Engineering, Shandong University, Jinan 250061, China*)

†E-mail: maxin@sdu.edu.cn

Received July 19, 2012; Revision accepted Oct. 12, 2012; Crosschecked Jan. 4, 2013

Abstract: This paper deals with a new approach based on Q-learning for solving the problem of mobile robot path planning in complex unknown static environments. As a computational approach to learning through interaction with the environment, reinforcement learning algorithms have been widely used for intelligent robot control, especially in the field of autonomous mobile robots. However, the learning process is slow and cumbersome. For practical applications, rapid rates of convergence are required. Aiming at the problem of slow convergence and long learning time for Q-learning based mobile robot path planning, a state-chain sequential feedback Q-learning algorithm is proposed for quickly searching for the optimal path of mobile robots in complex unknown static environments. The state chain is built during the searching process. After one action is chosen and the reward is received, the Q-values of the state-action pairs on the previously built state chain are sequentially updated with one-step Q-learning. With the increasing number of Q-values updated after one action, the number of actual steps for convergence decreases and thus, the learning time decreases, where a step is a state transition. Extensive simulations validate the efficiency of the newly proposed approach for mobile robot path planning in complex environments. The results show that the new approach has a high convergence speed and that the robot can find the collision-free optimal path in complex unknown static environments with much shorter time, compared with the one-step Q-learning algorithm and the $Q(\lambda)$-learning algorithm.

Key words: Path planning, Q-learning, Autonomous mobile robot, Reinforcement learning
doi:10.1631/jzus.C1200226 **Document code:** A **CLC number:** TP242.6

1 Introduction

The ultimate goal of mobile robotics is the development of autonomous mobile robots that can automatically navigate and perform a certain task in complex environments (transiting cargos, monitoring environments and patrolling, service, space exploration, rescue in hostile environments, etc.). Autonomous navigation plays a key role in the success of robots. As a key issue of mobile robot navigation, path planning refers to using a mobile robot to search for an optimal or sub-optimal collision-free path from its start point to its end point while satisfying some optimal criteria, such as the shortest path, time, or the minimum energy (Latombe, 1991).

There are two types of planning for autonomous mobile robots based on how much information is known about static environments: global path planning when the environment is clearly known, and sensory-based local path planning when partial or no information about the environment is known in advance. The techniques for path planning in known environments are relatively mature. The globally optimal or near-optimal path is computed with the visibility graph method (Alexopoulos and Griffin, 1992), Dijkstra's method (Saab and VanPutte, 1999), A* (Hart *et al.*, 1968; Dolgov *et al.*, 2010; Kala *et al.*,

* Project supported by the National Natural Science Foundation of China (Nos. 61075091, 61105100, and 61240052), the Natural Science Foundation of Shandong Province, China (No. ZR2012FM036), and the Independent Innovation Foundation of Shandong University, China (Nos. 2011JC011 and 2012JC005)

2010), etc., based on graph-based representation of environments. Although avoiding the heavy computation involved in establishment and maintenance of a global map, sensory-based path planning is complicated due to the fact that only partial or no information about the environment is known. Artificial potential field-based path planning algorithms utilize local sensory information in a purely reactive fashion (Barraquand *et al.*, 1992; Ge and Cui, 2002; Poty *et al.*, 2004; Agirrebeitia *et al.*, 2005; Cao *et al.*, 2006). Although their implementation is simple and they are robust in producing motions, a major problem is that the robot can be trapped into local minima before reaching its goal. The techniques for escaping local minima have to be studied.

Recently, there has been a growing tendency towards artificial intelligent approaches to sensory-based path planning in partially observable environments which are often stochastic in nature. Fuzzy logic-based path planning methods can summarize the motion rules for avoiding obstacles based on blurring sensory information without requiring accurate environment models (Yang *et al.*, 2005; Hachour, 2009). However, it is difficult to develop comprehensive fuzzy logic rules for complex environments and the generalization ability is not strong. Neural network based path planning methods build neural network models that map the sensory information to the motion controls for searching for the shortest path and avoiding collision (Yang and Meng, 2000; Ghatee and Mohades, 2009). Training the weights of neural networks is time-consuming. Based on the selection mechanism and gene inheritance, genetic algorithms help mobile robots travel from the start position to the desired goal with the shortest path without colliding with obstacles (AL-Taharwa *et al.*, 2008; Yun *et al.*, 2010; Tsai *et al.*, 2011). Castillo *et al.* (2007) presented a multi-objective genetic algorithm for off-line point-to-point autonomous mobile robot path planning which considers length and difficulty together. Particle swarm optimization (PSO) (Gong *et al.*, 2009) and ant colony optimization (ACO) (Garcia *et al.*, 2009; Wang *et al.*, 2011) have also been used for path planning. Their performance heavily depends on various parameters whose values cannot be chosen with theoretical guidelines. This limits the possibility of acquiring the optimal performance.

As a powerful machine learning technique that learns through interaction with the environment, reinforcement learning (RL) is suitable for autonomous mobile robots control in unknown environments. Q-learning algorithms can be used to learn a control policy that maximizes a scalar reward by using a trial-and-error process without requiring a prior knowledge of the environment (Watkins and Dayan, 1992). Owing to its simplicity and well-developed theory, the Q-learning algorithm has been widely used. However, the convergence process is a long one with the traditional Q-learning algorithm. Aiming at the slow convergence and long learning time of traditional Q-learning, many improved Q-learning approaches have been proposed. The improved Q-learning approaches can be classified into four categories according to four strategies as follows.

Q-value update strategy: The $Q(\lambda)$-learning algorithm extends the traditional one-step Q-learning algorithm with activity traces, which are used to handle delayed reward (Peng and Williams, 1996). The $Q(\lambda)$-learning algorithm is an incremental multi-step Q-learning. By allowing corrections to be made incrementally to the predictions of observations occurring in the past, i.e., to back more than one step at a time, the $Q(\lambda)$-learning method propagates information rapidly to the position where it is important. The learning rate can be increased for path planning in complex unknown static environments (Hwang *et al.*, 2011).

Action selection strategy: Local optimum could result from the greedy action selection strategy which is purely based on the current values of the state-action pairs, i.e., exploitation. The ϵ-greedy action selection strategy (Sutton and Barto, 1998) ($0 \leq \epsilon < 1$), with larger ϵ corresponding to a larger probability of choosing a non-optimal action in the current situation, can obtain more knowledge, i.e., exploration, with which local optima can be overcome. However, excessive exploration becomes unnecessary after a period of initial interaction. By representing Q-values as probability distributions and using these distributions to select actions that best balance exploration and exploitation, Bayesian Q-learning takes advantage of much more information and exhibits substantial improvement on convergence rate (Dearden *et al.*, 1998). The simulated annealing (SA) Q-learning algorithm is presented to balance exploitation and exploration with the Metropolis criterion (Guo *et al.*, 2004). In

Framling (2007), pre-existing knowledge was used to guide exploration in order to speed up reinforcement learning. In Still and Precup (2012), in addition to maximizing the expected reward, a learner selects the action that can maximize the prediction power of its own behavior measured by the information that the most recent state-action pair carries about the future.

Q-value initialization strategy: Q-learning can be accelerated by appropriately specifying initial Q-values (Koenig and Simmons, 1996). Fuzzy rules (Oh *et al.*, 1998), potential function (Wiewiora, 2003), and neural networks (Song *et al.*, 2012) are used to initialize Q-values to speed up learning.

State space reduction strategy: For complex environments there are a large number of states, which makes the learning time and storage for Q-table insufferable. In Senda *et al.* (2003), the state space was reduced by coordinates exclusion. In Hamagami and Hirata (2003) and Lampton and Valasek (2009), the number of states was adjusted adaptively. In Jin *et al.* (2009), the large state space was partitioned into multiple smaller state spaces based on the critical states. In Jaradat *et al.* (2011), the number of states was limited by a new definition of the discrete state space.

Although all above algorithms have sped up Q-learning, there is still room for speeding up convergence of Q-learning. To make reinforcement learning work on path planning with real robots, it is still an urgent problem to find optimal or near-optimal collision-free paths with fast learning speed in complex unknown static environments.

In this paper, we present a state-chain sequential feedback Q-learning algorithm for solving the problem of mobile robot path planning in complex unknown static environments. A state chain is built during searching. The state chain, which is saved as the memory matrix, records in order the state-action pairs from the start point until the current state. After one action is chosen with a greedy strategy and the reward or penalty is received, the Q-values of the state-action pairs on the state chain are sequentially updated with the one-step Q-learning algorithm. Theoretical analyses and extensive simulations have shown that the new approach has high convergence speed and can find the collision-free optimal or near-optimal path in complex unknown static environments with much shorter time, compared with one-step Q-learning and $Q(\lambda)$-learning algorithms.

The proposed state-chain sequential feedback Q-learning algorithm is distinctive in that it speeds up learning with a new Q-value update strategy. The Q-values of all state-action pairs on the current state chain are updated sequentially backward along the state chain after one action is chosen. By means of sequential feedback, the number of Q-values updated in one step increases. Although the proposed Q-learning algorithm is similar to $Q(\lambda)$-learning in that both of them are multi-step Q-learning, their Q-value update strategies are different. By using TD(λ) return estimation, the $Q(\lambda)$-learning algorithm propagates information backward. In contrast, the proposed Q-learning algorithm updates Q-values along the state-action chain. Theoretical analysis is given to compare the minimum number of steps required for convergence of Q-values of the three Q-learning algorithms in order to explain the different learning efficiencies of the three Q-learning algorithms.

2 One-step Q-learning algorithm and $Q(\lambda)$-learning algorithm

2.1 One-step Q-learning algorithm

The one-step Q-learning algorithm is a simple model-free incremental algorithm for delayed reinforcement learning (Watkins, 1989). A learning agent tries an action at a particular state, and evaluates its consequences in terms of the immediate reward or penalty it receives and its estimate of the value of the state to which it is taken. By trying all actions in all states repeatedly, it learns which are best overall judged by long-term discount reward. At time step t, the state is s_t. The action a_t that maximizes the Q-function $Q(s_t, a)$ is chosen. The Q-function is the estimated utility function that evaluates how good an action a_t is given a certain state s_t. The Q-value $Q(s_t, a_t)$ is the expected discount reward for executing action a_t at state s_t. The aim is to maximize the total discount expected reward. The Q-value $Q(s_t, a_t)$ is updated with

$$\begin{aligned} Q_{t+1}(s_t, a_t) =&(1-\lambda_t)Q_t(s_t, a_t) + \lambda_t[r_t \\ &+ \gamma \max_{a_{t+1}\in A} Q_{t+1}(s_{t+1}, a_{t+1})], \end{aligned} \quad (1)$$

where A is the set of actions, r_t is the immediate reward or penalty received by taking action a_t at state s_t, $0 < \gamma < 1$ is the discount factor, and γ reflects the influence of the successive state on the previous action. With the discount factor γ, the action decision can be influenced by the successive state. λ_t is the learning rate. It is proved that $Q_n(s,a)$ converges to $Q^*(s,a)$ with probability 1 as $n \to \infty$ with the learning rate $\lambda_t \in (0,1)$ and the bounded rewards or penalty $r_t \leq \mathcal{R}$, where $\mathcal{R}$ is a given positive real number (Watkins and Dayan, 1992).

The one-step Q-learning algorithm is summarized in Algorithm 1. It does not specify what action the agent should take at each state as it updates its Q-value estimates. The learning agent may take any action randomly. The optimal Q-function will be found eventually while trying out each action in every state many times without building the model of the environment. After Q-function convergence, the mobile robot selects its action at each state with a greedy policy $a_t^* = \arg\max_{a \in A} Q^*(s_t, a)$ from its start point to its end point. The optimal path is found by maximizing the expected total discount reward received. The one-step Q-learning algorithm updates the Q-value estimate of one state-action pair in one step. The learning time is long and the convergence is slow.

Algorithm 1 One-step Q-learning algorithm

$Q(s,a) = 0$ for all s and a // Initialization
loop
  observe its current state s_t
  select and perform an action a_t randomly
  observe the subsequent state s_{t+1}
  receive an immediate reward r_t
  update the Q-value $Q_{t+1}(s,a)$
  For $s = s(t)$ and $a = a(t)$, update $Q_{t+1}(s_t, a_t)$ with Eq. (1). Otherwise, $Q_{t+1}(s,a) = Q_t(s,a)$
end loop
until $Q(s,a)$ converges for all s and a

2.2 $Q(\lambda)$-learning algorithm

The $Q(\lambda)$-learning algorithm is the incremental multi-step Q-learning which extends the one-step Q-learning algorithm (Peng and Williams, 1996). It combines TD(λ) returns with the one-step Q-learning in an incremental way. With the multi-step $Q(\lambda)$-learning method, the previous actions can be affected by the feedback of the follow-up states. If the current action decision is false, then the previous actions should be punished. If the current action decision is correct, then the previous actions should be rewarded. The influence is added to the Q-value estimation of the previous state-action pairs with forgetting traces.

The $Q(\lambda)$-learning algorithm is summarized in Algorithm 2. One activity trace $\mathrm{Tr}(s,a)$ is defined for each state-action pair (s,a). After each step, the $Q(\lambda)$-learning algorithm loops through a set of state-action pairs which grow linearly with time. The Q-values of the set of state-action pairs whose activity trace $(\gamma\lambda)^n$ is significant are updated, where $\lambda \in (0,1)$ is the forgetting attenuation parameter, which is used to distribute credit throughout sequences of actions. Owing to the use of the TD(λ) return estimation process, which has the effect of making alterations to the past predictions through each step, the $Q(\lambda)$-learning algorithm propagates information backward much faster than the one-step Q-learning algorithm. The learning efficiency is improved significantly with the $Q(\lambda)$-learning algorithm.

Algorithm 2 $Q(\lambda)$-learning algorithm

$Q(s,a) = 0$ and $\mathrm{Tr}(s,a) = 0$ for all s and a // Initialization
loop
  observe its current state s_t
  select and perform an action $a_t = \arg\max_{a \in A} Q(s_t, a)$
  observe the subsequent state s_{t+1}
  receive an immediate reward r_t
  compute $e'_t = r_t + \gamma V_t(s_{t+1}) - Q_t(s_t, a_t)$
  compute $e_t = r_t + \gamma V_t(s_{t+1}) - V_t(s_t)$
  where $V_t(s) = \arg\max_{a \in A} Q(s,a)$
  for each state-action pair (s,a) **do**
    $\mathrm{Tr}(s,a) = \gamma\lambda \mathrm{Tr}(s,a)$
    $Q_{t+1}(s,a) = Q_t(s,a) + \lambda_t \mathrm{Tr}(s,a) e_t$
  end for
  update $Q_{t+1}(s_t, a_t) = Q_{t+1}(s_t, a_t) + \lambda_t e'_t$
  compute $\mathrm{Tr}(s_t, a_t) = \mathrm{Tr}(s_t, a_t) + 1$
end loop
until $Q(s,a)$ converges for all s and a

3 State-chain sequential feedback Q-learning algorithm

In this section, the state-chain sequential feedback Q-learning algorithm is presented for speeding

up the convergence of Q-learning. A state chain is built during the searching process. After an action is carried out and the reward or penalty is received, the Q-values of the state-action pairs on the state chain are sequentially updated with the traditional one-step Q-learning algorithm.

The state-chain sequential feedback Q-learning algorithm is summarized in Algorithm 3. At time step t, the state is s_t. One action a_t is chosen based on the greedy policy $a_t = \arg\max_{a\in A} Q(s_t, a)$. After action a_t is carried out, the reward or penalty r_t is rewarded. The memory matrix $\boldsymbol{M}(t)$ is extended with a new row $[s_t, a_t, r_t, \lambda_t]$. The state chain is saved as the memory matrix. The Q-values of state-action pairs are updated sequentially backward along the state chain:

For $k = t : -1 : 0$,

$$\begin{aligned} Q_{t+1}(s_k, a_k) =&(1-\lambda_k)Q_t(s_k, a_k) \\ &+ \lambda_k \left[r_k + \gamma V_t(s_{k+1})\right], \end{aligned} \tag{2}$$

where $V_t(s_{k+1}) = \arg\max_{a\in A} Q(s_{k+1}, a)$.

Algorithm 3 State-chain sequential feedback Q-learning algorithm

1: $Q(s, a) = 0$ for all s and a // Initialization
2: **loop**
3: observe its current state s_t
4: select and perform an action $a_t = \arg\max_{a\in A} Q(s_t, a)$
5: observe the subsequent state s_{t+1}
6: receive an immediate reward r_t
7: add a new row for memory matrix $\boldsymbol{M}(t)$ with $[s_t, a_t, r_t, \lambda_t]$
8: **for** $k = t$ to 0 **do**
9: update Q-values along the state chain recorded in $\boldsymbol{M}(t)$ $Q_{t+1}(s_k, a_k) = (1-\lambda_k)Q_t(s_k, a_k) + \lambda_k[r_k + \gamma V_{t+1}(s_{k+1})]$
10: **end for**
11: **end loop**
12: until $Q(s, a)$ converges for all s and a

Thus, the influence of previous action choice on the successive states can feed back in time to the Q-values of the previous state-action pairs recorded by the memory matrix with the state-chain sequential feedback Q-learning algorithm.

In each step, more Q-values are updated compared with the one-step Q-learning algorithm. The number of actual steps for convergence is decreased. Thus, the convergence is sped up. The next section gives a theoretical analysis, which explains the reason why the convergence can be improved with the state-chain sequential feedback Q-learning algorithm.

4 Theoretical analysis

In this section, the condition for convergence of Q-values is first analyzed. Then, the minimum number of updates required for the convergence of Q-values of one state is given. Finally, the required numbers of updates for the convergence of Q-values of all state-action pairs are compared for the three Q-learning algorithms.

4.1 Condition for convergence of Q-values

The Q-value of state-action pair (s_t, a_t) is $Q_t(s_t, a_t)$. (s_t, a_t) points to the next state s_{t+1}. The immediate reward or penalty is λ_t. The Q-value of state-action pair (s_t, a_t) is updated with the one-step Q-learning algorithm (Eq. (1)). If $\lambda_t \in (0, 1)$, and the Q-values of all state-action pairs (s_{t+1}, a) are convergent, then the Q-value $Q_t(s_t, a_t)$ must be convergent after sufficient updates.

From Eq. (1), it can be seen that the necessary and sufficient condition for stability of the Q-value $Q_t(s_t, a_t)$ is that $\max_{a\in A} Q(s_{t+1}, a) = Q_{t+1}$ is constant. Otherwise, the Q-value $Q(s_t, a_t)$ would change with Q_{t+1}.

With the assumption that the Q-values of all state-action pairs (s_{t+1}, a) are convergent, $\max_{a\in A} Q(s_{t+1}, a) = Q_{t+1}$ is constant. Assuming $\lambda_t = \alpha$, we obtain the update of $Q_t = Q(s_t, a_t)$ with Eq. (1).

After the first update,

$$Q_t = (1-\alpha)Q_t + \alpha(r + \gamma Q_{t+1}).$$

After the second update,

$$\begin{aligned} Q_t =&(1-\alpha)^2 Q_t + (1-\alpha)\alpha(r + \gamma Q_{t+1}) \\ &+ \alpha(r + \gamma Q_{t+1}). \end{aligned}$$

After the third update,

$$\begin{aligned} Q_t =&(1-\alpha)^3 Q_t + (1-\alpha)^2\alpha(r + \gamma Q_{t+1}) \\ &+ (1-\alpha)\alpha(r + \gamma Q_{t+1}) + \alpha(r + \gamma Q_{t+1}). \end{aligned}$$

After the nth update,

$$\begin{aligned}Q_t =&(1-\alpha)^n Q_t + (1-\alpha)^{n-1}\alpha(r+\gamma Q_{t+1})\\ &+(1-\alpha)^{n-2}\alpha(r+\gamma Q_{t+1}) + \cdots + \alpha(r+\gamma Q_{t+1})\\ =&(1-\alpha)^n Q_t + \alpha(r+\gamma Q_{t+1})\\ &\cdot[(1-\alpha)^{n-1} + (1-\alpha)^{n-2} + \cdots + 1]\\ =&(1-\alpha)^n Q_t + (r+\gamma Q_{t+1})[1-(1-\alpha)^n].\end{aligned}$$

As $\alpha = \lambda_t \in (0,1)$, $0 < 1-\alpha < 1$. When there are sufficient updates, $(1-\alpha)^n \to 0$, and

$$Q_t = r + \gamma Q_{t+1}.$$

That is, $Q(s_t, a_t)$ becomes convergent.

4.2 Minimum number of updates required for convergence of the Q-value of one state-action pair

From the above analysis of the condition for convergence of Q-values, we can see that the necessary condition for convergence of the state is the convergence of its subsequent state. The state-action pairs in the state chain converge at different times. The goal state would be convergent first. The Q-values corresponding to the state-action pairs near the goal state would be convergent earlier. The Q-values corresponding to the state-action pairs further away from the goal state would be convergent later. The convergent Q-values of state-action pairs help to make correct actions for subsequent learning. The non-convergent Q-values of state-action pairs would cause meaningless motions for mobile robots.

If the Q-values of the subsequent state are convergent, then the necessary condition for convergence of the Q-values of state s_t is sufficient updates. We assume that the minimum number of required updates is M.

$\forall \epsilon > 0$, if required $(1-\alpha)^m < \epsilon$, $m > \frac{\ln\epsilon}{\ln(1-\alpha)}$ should be satisfied.

Define $M = \left\lfloor \frac{\ln\epsilon}{\ln(1-\alpha)} \right\rfloor$. Hence, M is the minimum number of updates required for convergence of the Q-value of state-action pair (s_t, a_t) after the Q-values corresponding to the subsequent state s_{t+1} converge.

4.3 Minimum number of steps required for convergence of Q-values for each of the three Q-learning algorithms

The Q-values of all state-action pairs are convergent in the ideal stable situation. The state chain is shown in Fig. 1. The length of the state chain from the start state $s_0 = s_{\text{start}}$ to the goal state s_{goal} is T. The kth state s_k in the positive sequence is the R_Lth state s_{R_L} in the inverse sequence of the state chain at the same time. The distance from state s_k to goal state s_{goal} in the state chain is L. From goal state s_{goal} to start state s_0, the Q-values of all state-action pairs would be convergent progressively along the state chain. The number of steps required for the convergence of Q-values with the one-step Q-learning algorithm, the $Q(\lambda)$-learning algorithm, and the state-chain sequential feedback Q-learning algorithm are shown in Tables 1 and 2.

With the one-step Q-learning algorithm and the $Q(\lambda)$-learning algorithm, the minimum numbers of steps required for convergence of the Q-values of all state-action pairs in the state-chain, from state s_{R_1}

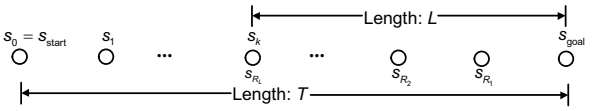

Fig. 1 Schematic diagram of the state chain. The length of the state chain from start state $s_0 = s_{\text{start}}$ to goal state s_{goal} is T. The kth state s_k in the positive sequence is the R_Lth state s_{R_L} in the inverse sequence of the state chain at the same time. The distance from state s_k to goal state s_{goal} in the state chain is L

Table 1 Convergence of Q-values of state-action pairs along state-chain with the one-step Q-learning algorithm and $Q(\lambda)$-learning algorithm

Order convergence	State	Required number of updates	Number of updates in every episode from s_{start} to s_{goal}	Required minimum number of episodes	Required minimum number of steps
1	s_{R_1}	M	1	M	MT
2	s_{R_2}	M	1	M	MT
3	s_{R_3}	M	1	M	MT
⋮	⋮	⋮	⋮	⋮	⋮
L	s_{R_L}	M	1	M	MT

Table 2 Convergence of Q-values of state-action pairs along state-chain with the state-chain sequential feedback Q-learning algorithm

Order of convergence	State	Required number of updates	Number of updates in every episode from s_{start} to s_{goal}	Required minimum number of episodes	Required minimum number of steps
1	s_{R_1}	M	1	M	MT
2	s_{R_2}	M	2	$M/2$	$MT/2$
3	s_{R_3}	M	3	$M/3$	$MT/3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
L	s_{R_L}	M	L	M/L	MT/L

to state s_{R_L}, are N_Q and N_λ.

$$N_Q = N_\lambda = L \cdot M \cdot T. \tag{3}$$

With the state-chain sequential feedback Q-learning algorithm, the minimum number of steps for convergence of the Q-values of all state-action pairs in the state-chain, from state s_{R_1} to state s_{R_L}, is N_{SQ}.

$$N_{\text{SQ}} = M \cdot T \cdot \left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{L}\right). \tag{4}$$

For $L > 1$, $L > 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{L}$. Therefore, $N_{\text{SQ}} < N_Q$ and $N_{\text{SQ}} < N_\lambda$.

From the above theoretical analysis, it can be seen that the proposed state-chain sequential feedback Q-learning algorithm decreases the minimum number of steps required for the convergence of the Q-values of all state-action pairs by increasing the number of Q-values updated in one step. In each episode from the start state s_0 to the end state s_{goal}, the Q-values of each state-action pair are updated many times. The Q-values are updated by the next states sequentially backward along the state-chain with the update equation of the one-step Q-learning algorithm. With the increasing number of Q-values updated in one step, the number of steps for convergence decreases and thus, the learning time decreases.

5 Experimental results

In this section, we evaluate the performance of the proposed state-chain sequential feedback Q-learning algorithm through MATLAB simulation. In three different environments, we compare the convergence of the proposed state-chain sequential feedback Q-learning algorithm with that of the one-step Q-learning algorithm and the $Q(\lambda)$-learning algorithm. The three Q-learning algorithms are different in the Q-value update strategy.

5.1 Environment representation and action set

The environment is represented by a 2D occupancy grid map in which each grid is given a value 1 (with obstacle) or 0 (free). The occupancy grid map representation is particularly useful for the task of obstacle avoidance. The action set A includes five kinds of actions: moving up, moving down, moving left, moving right, and staying at rest. The five actions a_t are represented by $[0, 1]$, $[0, -1]$, $[1, 0]$, $[-1, 0]$, and $[0, 0]$, respectively. The state of the mobile robot s_t is its current position at time step t in the environment. The current state s_t has five state-action pairs (s_t, a_t), each of them corresponding to a Q-value $Q(s_t, a_t)$. All Q-values are initialized with zeros. The optimal action is selected with the greedy strategy, $a_t = \arg\max_{a \in A} Q_t(s_t, a)$, according to the current state-action pair Q-value. s_{t+1} is the next state (a new position in the environment) after action a_t is carried out. r_t is the immediate reward or penalty from action a_t. If the next position is occupied by an obstacle, then $s_{t+1} = s_t$, and $r_t = -0.2$. If the next position is free, then s_{t+1} is equal to the next position and $r_t = -0.1$. If the next position is the goal, then $s_{t+1} = s_{\text{goal}}$ and $r_t = 1$. The discount factor γ reflects the influence of subsequent state s_{t+1} on the Q-value of the previous state-action pair $Q_t(s_t, a_t)$. $0 < \gamma < 1$. The larger the γ, the larger the influence of the subsequent state on the previous action choice. In our simulations, we set the discount factor $\gamma = 0.95$ and the learning factor $\lambda_t = 0.3$.

5.2 Simulations

We simulate three different environments. In each simulation environment, we compare the

proposed state-chain sequential feedback Q-learning algorithm with the traditional one-step Q-learning algorithm and the $Q(\lambda)$-learning algorithm. To compare them, performances are evaluated by two criteria, namely the average number of steps to the goal in each episode and reward values in each episode. The two criteria measure how well the learning for path planning task has been performed.

The first simulation environment is represented by a 10×10 occupancy grid map (Fig. 2a). The occupied grid is black. The free grid is white. At time step t, the mobile robot takes one of the five actions, a_t, and moves along the grid. The current state of mobile robot s_t is its position in the grid map. Each state-action pair (s_t, a_t) has a Q-value $Q(s_t, a_t)$. There are 500 Q-values which are initialized as a 1×500 zero matrix. After one action is carried out, the memory matrix $\boldsymbol{M}$ is extended with a new row $[s_t, a_t, r_t, \lambda_t]$. The row number of the memory matrix increases with searching. The task of the robot is to travel from the start point to the goal point. All agents perform 120 episodes with the three Q-learning algorithms respectively. The agents can converge to one of the shortest collision-free paths whose length is 21 with all three Q-learning algorithms. One of the shortest collision-free paths found by the proposed algorithm is shown in Fig. 2a, and Fig. 2b shows the average number of steps from the start to the goal in each episode with the three Q-learning algorithms in the environment. We can see that the proposed Q-learning algorithm converges to one of the shortest paths with fewer episodes than the other two Q-learning algorithms. With the help of the new Q-value update strategy, the proposed Q-learning algorithm can converge quickly. Similar learning performance can be seen from Fig. 2c, in which the reward values of each episode of the three Q-learning algorithms are compared.

The second environment is a 20×20 grid (Fig. 3a). The three Q-learning algorithms can converge to one of the shortest collision-free paths whose length is 28. One of the shortest collision-free paths found by the proposed Q-learning algorithm is shown in Fig. 3a, and the average number of steps to the goal in each episode and the reward value of each episode are different for the three algorithms (Figs. 3b and 3c). We can see that the proposed Q-learning algorithm converges the fastest.

The third environment is a 30×30 grid (Fig. 4a).

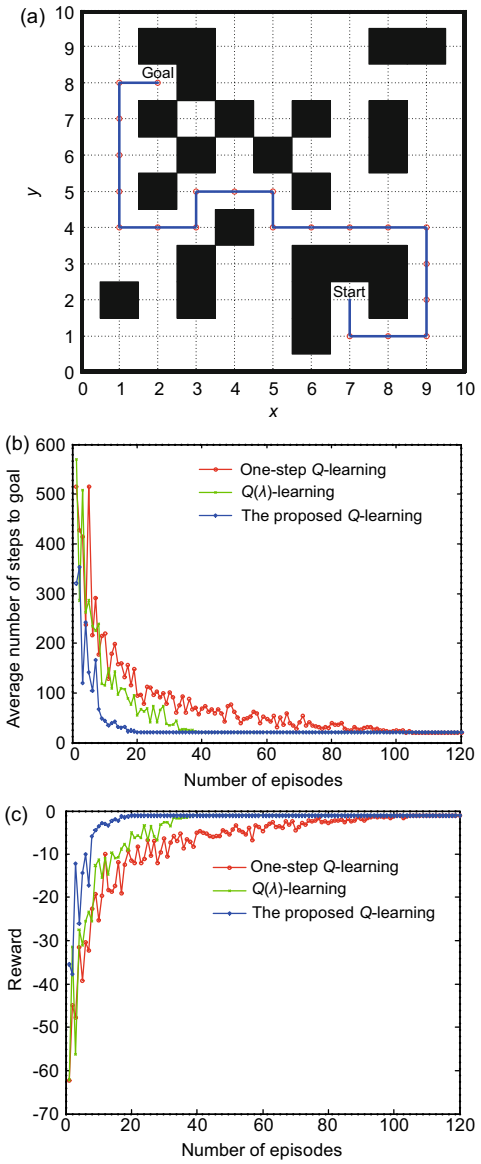

Fig. 2 Comparison of the learning performance of the three Q-learning algorithms in a 10×10 grid. (a) The shortest path found by the proposed Q-learning algorithm; (b) Number of episodes vs. the average number of steps to the goal; (c) Number of episodes vs. reward

The agents perform 500 episodes with the three Q-learning algorithms. The proposed Q-learning algorithm and the $Q(\lambda)$-learning algorithm can converge to one of the shortest collision-free paths within 500 episodes. Fig. 4a shows one of the shortest collision-free paths, whose length is 48, found by the proposed Q-learning algorithm. Figs. 4b and 4c show that with the proposed Q-learning algorithm, the average number of steps to the goal in each episode is smaller than those with the other two Q-learning

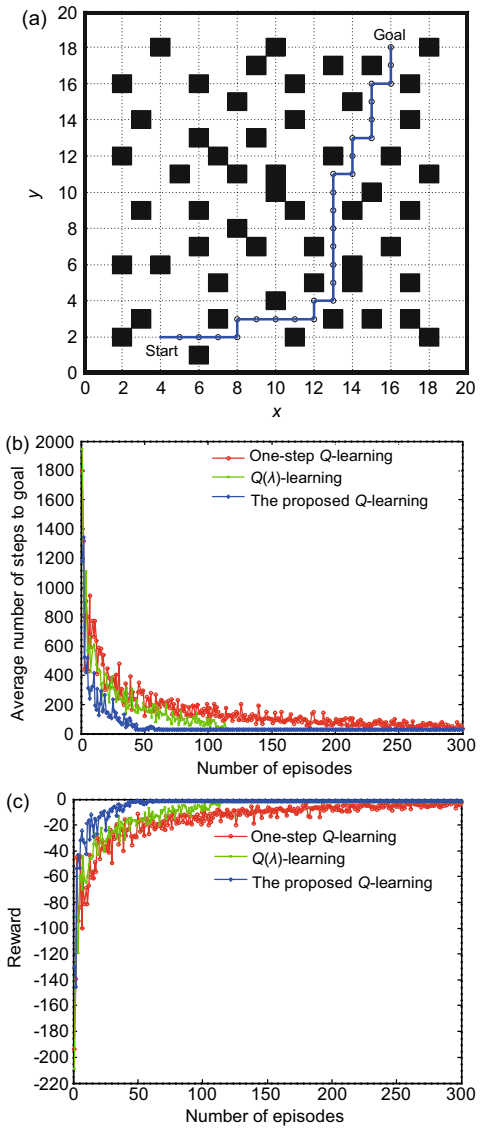

Fig. 3 Comparison of the learning performance of the three Q-learning algorithms in a 20×20 grid. (a) The shortest path found by the proposed Q-learning algorithm; (b) Number of episodes vs. the average number of steps to the goal; (c) Number of episodes vs. reward

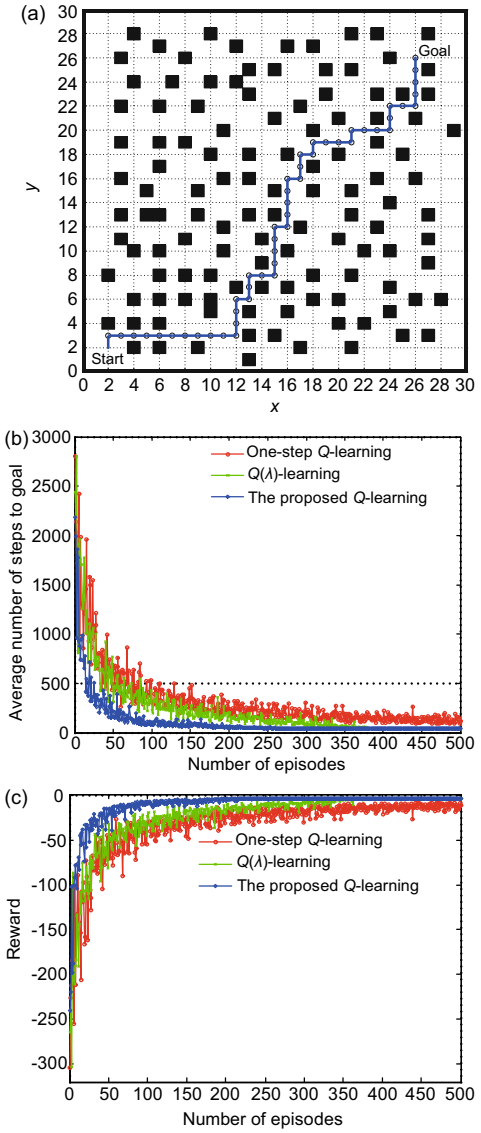

Fig. 4 Comparison of the learning performance of the three Q-learning algorithms in a 30×30 grid. (a) The shortest path found by the proposed Q-learning algorithm; (b) Number of episodes vs. the average number of steps to the goal; (c) Number of episodes vs. reward

algorithms, and the reward value of each episode is higher than that with the other two Q-learning algorithms. Figs. 4b and 4c also show that the one-step Q-learning algorithm cannot converge within 500 episodes, which means that it cannot converge to one of the shortest paths within 500 episodes. Although the $Q(\lambda)$-learning algorithm can find one of the shortest collision-free paths within 500 episodes, the convergence rate is lower than that of the proposed Q-learning algorithm.

Table 3 gives the number of episodes when one of the shortest collision-free paths is first found (N_f), the number of episodes when convergence is reached (N_c), and the total number of steps to convergence (N_ts) with the three Q-learning algorithms in the three different environments. Table 3 shows that all the three Q-learning algorithms can converge to one of the shortest paths whose length is 21 in the 10×10 grid. The proposed Q-learning algorithm needs only 21 episodes for reaching convergence, while the $Q(\lambda)$-learning algorithm needs 43 episodes and the one-step Q-learning algorithm

Table 3 Comparison of the number of episodes for convergence with the three Q-learning algorithms in the three environments

Environment	N_f			N_c			N_{ts}		
	TQ	$Q(\lambda)$	SCSF-Q	TQ	$Q(\lambda)$	SCSF-Q	TQ	$Q(\lambda)$	SCSF-Q
10×10 grid	95	34	13	112	43	21	9163	4792	1939
20×20 grid	314	114	50	410	127	56	52 804	26 163	9875
30×30 grid	>500	328	202	>500	362	242	–	110 210	37 896

N_f: number of episodes when one of the shortest collision-free paths is first found; N_c: number of episodes when convergence is reached; N_{ts}: total number of steps to convergence. TQ: traditional one-step Q-learning; SCSF-Q: state-chain sequential feedback Q-learning

needs 112 episodes. In the 20×20 grid environment, the proposed Q-learning algorithm needs 56 episodes to converge to one of the shortest collision-free paths, whose length is 28. $Q(\lambda)$ and one-step Q-learning algorithms need 127 and 410 episodes, respectively. In the 30×30 grid environment, the proposed Q-learning algorithm can converge to one of the shortest paths at the 242nd episode. $Q(\lambda)$-learning needs 362 episodes for convergence, and the one-step Q-learning algorithm needs more than 500 episodes.

As shown in Table 3, in the 10×10 grid, the total numbers of steps for convergence are 9163, 4792, and 1939 with one-step Q-learning, $Q(\lambda)$, and the proposed Q-learning algorithms, respectively. The proposed Q-learning algorithm needs 21.2% of the steps of the one-step Q-learning algorithm, and 40.5% of the steps of the $Q(\lambda)$-learning algorithm. In the 20×20 grid, the total numbers of steps for convergence are 52 804, 26 163, and 9875 with the above three Q-learning algorithms, respectively. The proposed Q-learning algorithm needs 18.7% of the number of steps of one-step Q-learning, 37.7% of the number of steps of the $Q(\lambda)$-learning algorithm. In the 30×30 grid, the total numbers of steps for convergence are 37 896 and 110 210 with the proposed Q-learning and the $Q(\lambda)$-learning algorithms, respectively. The proposed Q-learning algorithm needs only 34.4% of the number of the steps of the $Q(\lambda)$-learning algorithm. It is shown that the proposed state-chain sequential feedback Q-learning algorithm can improve the learning rate, especially for complex unknown environments.

The results of the experiments validate that the proposed state-chain sequential feedback Q-learning algorithm can converge to one of the shortest collision-free paths with fewer episodes and faster convergence rates compared with one-step Q-learning and $Q(\lambda)$-learning. The improvement is more obvious for complex unknown environments.

6 Conclusions

This paper deals with a state-chain sequential feedback Q-learning algorithm for path planning of autonomous mobile robots in unknown static environments. With the occupancy grid map representation of environments, the state of a mobile robot is its discrete position in the map. A memory matrix is built to remember the state-action pairs during the searching process and a state chain is established. After each action is taken, the Q-values of the state-action pairs are updated sequentially backward with the traditional one-step Q-learning along the state chain. Theoretical analysis proves that the Q-value of each state-action pair is updated many times with the subsequent states in each episode from the start point to the goal with the proposed Q-learning algorithm. As a result, the number of steps for convergence decreases and the learning speed increases.

Different simulated test scenarios are conducted to test the performance of the proposed state-chain sequential feedback Q-learning algorithm. Simulation results show that the proposed state-chain sequential feedback Q-learning algorithm improves the learning speed for finding one of the shortest collision-free paths for autonomous mobile robots, especially in complex unknown static environments.

References

Agirrebeitia, J., Aviles, R., de Bustos, I.F., Ajuria, G., 2005. A new APF strategy for path planning in environments with obstacles. *Mech. Mach. Theory*, **40**(6):645-658. [doi:10.1016/j.mechmachtheory.2005.01.006]

Alexopoulos, C., Griffin, P.M., 1992. Path planning for a mobile robot. *IEEE Trans. Syst. Man Cybern.*, **22**(2): 318-322. [doi:10.1109/21.148404]

Al-Taharwa, I., Sheta, A., Al-Weshah, M., 2008. A mobile robot path planning using genetic algorithm in static

environment. *J. Comput. Sci.*, **4**(4):341-344.

Barraquand, J., Langlois, B., Latombe, J.C., 1992. Numerical potential field techniques for robot path planning. *IEEE Trans. Syst. Man Cybern.*, **22**(2):224-241. [doi:10.1109/21.148426]

Cao, Q., Huang, Y., Zhou, J., 2006. An Evolutionary Artificial Potential Field Algorithm for Dynamic Path Planning of Mobile Robot. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, p.3331-3336. [doi:10.1109/IROS.2006.282508]

Castillo, O., Trujillo, L., Melin, P., 2007. Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots. *Soft Comput.*, **11**(3):269-279. [doi:10.1007/s00500-006-0068-4]

Dearden, R., Friedman, N., Russell, S., 1998. Bayesian Q-Learning. Proc. National Conf. on Artificial Intelligence, p.761-768.

Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J., 2010. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot. Res.*, **29**(5):485-501. [doi:10.1177/0278364909359210]

Framling, K., 2007. Guiding exploration by pre-existing knowledge without modifying reward. *Neur. Networks*, **20**(6):736-747. [doi:10.1016/j.neunet.2007.02.001]

Garcia, M.A., Montiel, O., Castillo, O., Sepulveda, R., Melin, P., 2009. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Appl. Soft Comput.*, **9**(3):1102-1110. [doi:10.1016/j.asoc.2009.02.014]

Ge, S.S., Cui, Y.J., 2002. Dynamic motion planning for mobile robots using potential field method. *Auton. Robots*, **13**(3):207-222. [doi:10.1023/A:1020564024509]

Ghatee, M., Mohades, A., 2009. Motion planning in order to optimize the length and clearance applying a hopfield neural network. *Expert Syst. Appl.*, **36**(3):4688-4695. [doi:10.1016/j.eswa.2008.06.040]

Gong, D., Lu, L., Li, M., 2009. Robot Path Planning in Uncertain Environments Based on Particle Swarm Optimization. Proc. IEEE Congress on Evolutionary Computation, p.2127-2134. [doi:10.1109/CEC.2009.4983204]

Guo, M., Liu, Y., Malec, J., 2004. A new Q-learning algorithm based on the metropolis criterion. *IEEE Trans. Syst. Man Cybern. B*, **34**(5):2140-2143. [doi:10.1109/TSMCB.2004.832154]

Hachour, O., 2009. The proposed fuzzy logic navigation approach of autonomous mobile robots in unknown environments. *Int. J. Math. Models Methods Appl. Sci.*, **3**(3):204-218.

Hamagami, T., Hirata, H., 2003. An Adjustment Method of the Number of States of Q-Learning Segmenting State Space Adaptively. Proc. IEEE Int. Conf. on Systems, Man and Cybernetics, **4**:3062-3067. [doi:10.1109/ICSMC.2003.1244360]

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, **4**(2):100-107. [doi:10.1109/TSSC.1968.300136]

Hwang, H.J., Viet, H.H., Chung, T., 2011. $Q(\lambda)$ based vector direction for path planning problem of autonomous mobile robots. *Lect. Notes Electr. Eng.*, **107**(Part 4):433-442. [doi:10.1007/978-94-007-2598-0_46]

Jaradat, M.A.K., Al-Rousan, M., Quadan, L., 2011. Reinforcement based mobile robot navigation in dynamic environment. *Robot. Comput.-Integr. Manuf.*, **27**(1):135-149. [doi:10.1016/j.rcim.2010.06.019]

Jin, Z., Liu, W., Jin, J., 2009. Partitioning the State Space by Critical States. Proc. 4th Int. Conf. on Bio-Inspired Computing, p.1-7. [doi:10.1109/BICTA.2009.5338123]

Kala, R., Shukla, A., Tiwari, R., 2010. Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning. *Artif. Intell. Rev.*, **33**(4):307-327. [doi:10.1007/s10462-010-9157-y]

Koenig, S., Simmons, R.G., 1996. The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Mach. Learn.*, **22**(1-3):227-250. [doi:10.1007/BF00114729]

Lampton, A., Valasek, J., 2009. Multiresolution State-Space Discretization Method for Q-Learning. Proc. American Control Conf., p.1646-1651. [doi:10.1109/ACC.2009.5160474]

Latombe, J.C., 1991. Robot Motion Planning. Kluwer Academic Publishers. [doi:10.1007/978-1-4615-4022-9]

Oh, C.H., Nakashima, T., Ishibuchi, H., 1998. Initialization of Q-Values by Fuzzy Rules for Accelerating Q-Learning. Proc. IEEE World Congress on Computational Intelligence and IEEE Int. Joint Conf. on Neural Networks, **3**:2051-2056. [doi:10.1109/IJCNN.1998.687175]

Peng, J., Williams, R.J., 1996. Incremental multi-step Q-learning. *Mach. Learn.*, **22**(1-3):283-290. [doi:10.1023/A:1018076709321]

Poty, A., Melchior, P., Oustaloup, A., 2004. Dynamic Path Planning for Mobile Robots Using Fractional Potential Field. Proc. 1st Int. Symp. on Control, Communications and Signal Processing, p.557-561. [doi:10.1109/ISCCSP.2004.1296443]

Saab, Y., VanPutte, M., 1999. Shortest path planning on topographical maps. *IEEE Trans. Syst. Man Cybern. A*, **29**(1):139-150. [doi:10.1109/3468.736370]

Senda, K., Mano, S., Fujii, S., 2003. A Reinforcement Learning Accelerated by State Space Reduction. SICE Annual Conf., **2**:1992-1997.

Song, Y., Li, Y., Li, C., Zhang, G., 2012. An efficient initialization approach of Q-learning for mobile robots. *Int. J. Control Autom. Syst.*, **10**(1):166-172. [doi:10.1007/s12555-012-0119-9]

Still, S., Precup, D., 2012. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory Biosci.*, **131**(3):139-148. [doi:10.1007/s12064-011-0142-z]

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: an Introduction. MIT Press, Cambriage, MA.

Tsai, C.C., Huang, H.C., Chan, C.K., 2011. Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation. *IEEE Trans. Ind. Electron.*, **58**(10):4813-4821. [doi:10.1109/TIE.2011.2109332]

Wang, Z., Zhu, X., Han, Q., 2011. Mobile robot path planning based on parameter optimization ant colony algorithm. *Proc. Eng.*, **15**:2738-2741. [doi:10.1016/j.proeng.2011.08.515]

Watkins, C.J.C.H., 1989. Learning from Delayed Rewards. University of Cambridge, Cambridge, UK.

Watkins, C.J.C.H., Dayan, P., 1992. Q-learning. *Mach. Learn.*, **8**(3-4):279-292.

Wiewiora, E., 2003. Potential-based shaping and Q-value initialization are equivalent. *Artif. Intell. Res.*, **19**:205-208.

Yang, S.X., Meng, M., 2000. An efficient neural network approach to dynamic robot motion planning. *Neur. Networks*, **13**(2):143-148. [doi:10.1016/S0893-6080(99)00103-3]

Yang, X., Moallem, M., Patel, R.V., 2005. A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation. *IEEE Trans. Syst. Man Cybern. B*, **35**(6):1214-1224. [doi:10.1109/TSMCB.2005.850177]

Yun, S.C., Ganapathy, V., Chong, L.O., 2010. Improved Genetic Algorithms Based Optimum Path Planning for Mobile Robot. Proc. 11th Int. Conf. on Control, Automation, Robotics and Vision, p.1565-1570. [doi:10.1109/ICARCV.2010.5707781]