一种多步 Q 强化学习方法*)

陈圣磊　吴慧中　韩祥兰　肖　亮
(南京理工大学计算机科学与技术系　南京 210094)

摘　要　Q 学习是一种重要的强化学习算法。本文针对 Q 学习和 $Q(\lambda)$ 算法的不足，提出了一种具有多步预见能力的 Q 学习方法：MQ 方法。首先给出了 MDP 模型，在分析 Q 学习和 $Q(\lambda)$ 算法的基础上给出了 MQ 算法的推导过程，并分析了算法的更新策略和 k 值的确定原则。通过悬崖步行仿真试验验证了该算法的有效性。理论分析和数值试验均表明，该算法具有较强的预见能力，同时能降低计算复杂度，是一种有效平衡更新速度和复杂度的强化学习方法。

关键词　强化学习，MQ 算法，Q 学习，$Q(\lambda)$ 算法

A Multi-step Q Reinforcement Learning Algorithm

CHEN Sheng-Lei　WU Hui-Zhong　HAN Xiang-Lan　XIAO Liang
(Department of Computer Science & Technology, Nanjing University of Science & Technology, Nanjing 210094)

Abstract　Q learning is of great importance in reinforcement learning. MQ algorithm with multi-step predicting capability is proposed to compensate the drawbacks of Q learning and $Q(\lambda)$ algorithmin in this paper . Firsly MDP model is presented. Then based on the analyses of Q learning and $Q(\lambda)$algorithm, MQ algorithm is proposed. The algorithm's update strategy and determination rule of k are also analyzed. The effectiveness of this algorithm is verified through the cliff walking simulation experiments. Theoretical analyses and experiments indicate that better predicting capability and decreased computation complexity can be obtained in this algorithm. So it can balance update speed and complexity effectively.

Keywords　Reinforcement learning, MQ algorithm, Q learning, $Q(\lambda)$algorithm

1　引言

强化学习是人工智能领域中既崭新又古老的课题，其思想来源于生物(特别是人)为适应环境而进行的学习过程。研究者发现[1]，人类进化过程中的学习行为有两个特点：一是人从来不是静止地被动等待而是主动对环境做试探，二是环境对试探动作产生的反馈是评价性的，人们会根据环境的评价来调整以后的行为，也就是我们平时所说的"吃一堑，长一智"。强化学习正是通过试探-评价的迭代逐步获得对环境的认识，从而产生此环境中的最优行为，因此是一种从环境状态到行为映射的学习[2]。强化学习不同于无监督学习和监督学习，无监督学习利用的是生理学中的条件反射原理，监督学习则是一种反馈式学习过程，通过将系统输出与理想输出信号的误差反馈给系统以改善系统输出。

Watkins 在 1989 年提出 Q 学习算法并证明其收敛性以后[3,4]，该算法在强化学习领域受到了普遍关注，针对该算法的改进也层出不穷。其中 Peng 在 1996 年提出了增量式多步 Q 学习[5]，即 $Q(\lambda)$算法，它结合了 Q 学习和 $TD(\lambda)$回报的思想，利用将来无限多步的信息更新当前 Q 函数，并给出了基于资格迹的实现算法。这两种算法存在的问题是：Q 学习利用了一步的信息，更新的速度慢，预见能力不强，而 $Q(\lambda)$算法要对大量的状态-动作对 SAP(State-Action Pair)的 Q 值和相应的资格迹矩阵进行更新，当状态-动作空间规模很大时计算量较大，学习效率不高。为了平衡这两个方面的问题，本文提出采用有限多步信息进行更新的思想，称为多步 Q 学习(MQ, Multi-step $Q(\lambda)$ learning)算法。它利用将来 k 步的信息更新当前的 Q 值，具有多步预见能力，同时能降低计算复杂度。这里的预见能力是指智能体更新当前状态-动作对的 Q 值时，考虑了将来若干个状态-动作对的影响，反映了智能体对未来的思考，从而使当前决策更加理性。

2　Q 学习与 $Q(\lambda)$学习

下面首先给出作为强化学习数学模型的 Markov 决策过程(MDP)模型，然后讨论 Watkins 的 Q 学习和 Peng 的 $Q(\lambda)$学习的思想和算法。

2.1　Markov 决策过程(MDP)模型

现实生产活动中的控制系统、库存管理和调度问题等可以归结为 MDP 问题，而强化学习正是为解决这类问题提出的。一个 MDP 模型可以表示为以下五元组[6]：

$$MDP=\langle S,A(x),P_{xy}(a),r_{xy}(a),V\rangle$$

S 是由系统所有可能的状态组成的非空集合，也称为系统的状态空间。状态用 x,y 表示，x_t 表示 t 时刻的状态。

对状态 $x\in S$，$A(x)$是在状态 x 可用的动作集(也称为决策集)，它是非空的。用 a,b 表示动作。全体状态上的可用动作集 $A=\bigcup_{x\in S}A(x)$。

$P_{xy}(a)$是在状态 x 采用动作 a 从而转移到状态 y 的概率，称为状态转移概率。

$r_{xy}(a)$是在状态 x 采用动作 a 并且转移到状态 y 获得的

*)本文得到国防预研基金项目资助。陈圣磊　博士生，研究方向为智能体理论与应用、强化学习；吴慧中　教授，博士生导师，研究方向为虚拟现实、系统仿真、计算机图形图像理论；韩祥兰　博士生，研究方向为智能决策支持；肖　亮　博士，讲师，研究方向为模式识别与图像处理。

报酬，$r_{x_t,x_{t+1}}(a)$简记为 r_t。

V 为准则函数，它可以有多种取法[6]，这里取为无限阶段的期望总折扣报酬：

$$V(x)=E[\sum_{n=0}^{\infty}\gamma^n\cdot r_{t+n}]$$

其中，γ 为折扣因子。

2.2 Q学习

Q学习是通过维护一个状态-动作空间上的值函数来获得最佳策略[4]。在策略 π 下，状态-动作空间上的值函数 Q 定义为：

$$Q^{\pi}(x,a)=R_x(a)+\gamma\cdot\sum_{y\in S}P_{xy}(a)\cdot V^{\pi}(y)$$

其中策略 π 是一个决策函数的序列，$V^{\pi}(y)$ 是在策略 π 下状态 y 的期望总折扣期望，$R_x(a)=\sum_{y\in S}P_{xy}(a)\cdot r_{xy}(a)$ 为由状态 x 采取动作 a 做出一次转移的期望报酬，称为状态 x 的即时期望报酬。

强化学习的任务就是通过估计 Q 值来得到最优策略 π^*。为简单起见，记 $Q^*(x,a)\equiv Q^{\pi^*}(x,a)\ \forall x,a$。其学习过程是由情节(episode)和步骤(step)组成，其中步骤是指一个确定的状态及该状态下的动作执行和报酬获得，情节是指从起始状态到目标状态的步骤的序列。在第 t 时刻的步骤中，智能体处于当前状态 x_t，选择并执行动作 a_t，观测下一个状态 y_t(即 x_{t+1})和报酬 r_t，按照下式更新 Q_t 的值：

$$Q_t(x,a)=\begin{cases}(1-\alpha)Q_{t-1}(x,a)+\alpha_t\cdot[r_t+\gamma\cdot V_{t-1}(y_t)], & \text{如果 } x=x_t \text{ 且 } a=a_t\\ Q_{t-1}(x,a), & \text{其它情况}\end{cases}$$

其中 $Q_t(x,a)$ 为 t 时刻在状态 x 下执行动作 a 所得的回报，α_t 为学习率，$V_{t-1}(y)=\max_{b\in A(y)}\{Q_{t-1}(y,b)\}$。

文[4]证明了Q学习在一定条件下的收敛性。

定理 1 给定有界强化信号 $|r_t|\leqslant R$，学习率 $0\leqslant\alpha_t\leqslant 1$ 及 $\sum_{t=1}^{\infty}\alpha_t(x,a)=\infty;\sum_{t=1}^{\infty}[\alpha_t(x,a)]^2\leqslant\infty,\ \forall x,a$

则当 $t\to\infty$ 时，$Q_t(x,a)$ 以概率 1 收敛于最优 $Q^*(x,a)$。

2.3 Q(λ)学习

Q学习利用了将来一步的信息更新 Q 值。也就是说，对 $Q_t(x,a)$ 的更新包含了下一状态 y 中可能决策的影响。如果下一决策 a_{t+1} 是一个失败的决策，那么当前的决策也要承担相应的责任，也要把这种影响追加到当前决策上来，因此也就产生了上面的Q学习算法。表面上看，这是一种一步更新算法，但实际上它也考虑了以后所有决策的影响。因为这种影响是递归的，只不过这种影响并没有在更新规则中显式地表达出来。不妨考虑在Q学习值函数的更新中，显式地利用将来所有决策的影响，这就是 Peng 的 $Q(\lambda)$ 学习[5]。

首先定义 t 时刻的校正 n 步截断回报为：

$$r_t^{(n)}=r_t+\gamma\cdot r_{t+1}+\cdots+\gamma^{n-1}\cdot r_{t+n-1}+\gamma^n\cdot V_{t+n-2}(x_{t+n}) \tag{1}$$

考虑到对 $0\leqslant\lambda<1$，有 $(1-\lambda)\cdot(1+\lambda+\lambda^2+\cdots)=1$

所以定义 TD(λ)回报 r_t^{λ} 为 $r_t^{(n)}$，$(n=1,2,3\cdots)$ 的加权平均值，即

$$\begin{aligned}r_t^{\lambda}&=(1-\lambda)\cdot[r_t^{(1)}+\lambda\cdot r_t^{(2)}+\lambda^2\cdot r_t^{(3)}+\cdots]\\&=r_t+\gamma\cdot V_{t-1}(x_{t+1})+\sum_{i=1}^{\infty}(\lambda\gamma)^i\cdot[r_{t+i}+\gamma\cdot V_{t+i-1}(x_{t+i+1})-V_{t+i-2}(x_{t+i})]\\&=r_t+\gamma\cdot V_{t-1}(x_{t+1})+\sum_{i=1}^{\infty}(\lambda\gamma)^i\cdot[r_{t+i}+\gamma\cdot V_{t+i-1}(x_{t+i+1})-V_{t+i-1}(x_{t+i})]+\sum_{i=1}^{\infty}(\lambda\gamma)^i\cdot[V_{t+i-1}(x_{t+i})-V_{t+i-2}(x_{t+i})]\end{aligned}$$

记

$$e_t=r_t+\gamma\cdot V_{t-1}(x_{t+1})-V_{t-1}(x_t) \tag{2}$$

$$e'_t=r_t+\gamma\cdot V_{t-1}(x_{t+1})-Q_{t-1}(x_t,a_t) \tag{3}$$

因此有

$$r_t^{\lambda}-Q_{t-1}(x_t,a_t)=e'_t+\sum_{i=1}^{\infty}(\lambda\gamma)^i\cdot e_{t+i}+\sum_{i=1}^{\infty}(\lambda\gamma)^i\cdot[V_{t+i-1}(x_{t+i})-V_{t+i-2}(x_{t+i})] \tag{4}$$

如果学习率 α 的值比较小，那么 Q 值的调整就比较慢，(4)式右边第二个求和式就很小，可以忽略。

那么 $Q(\lambda)$ 学习的更新规则为：

$$Q_t(x,a)=\begin{cases}Q_{t-1}(x,a)+\alpha_t\cdot[e'_t+\sum_{i=1}^{\infty}(\lambda\gamma)^i\cdot e_{t+i}], & \text{如果 } x=x_t \text{，且 } a=a_t\\ Q_{t-1}(x,a), & \text{其它情况}\end{cases}$$

3 多步Q学习算法

3.1 算法提出

Q学习算法仅显式地利用了将来一步的信息，一步以后的决策对当前的状态影响较小，也就是说它只考虑到当前的效果，预见能力较差。$Q(\lambda)$ 学习算法利用了将来的所有信息更新当前的 Q 值，能够考虑未来的决策对当前 Q 值的影响，预见能力很强，但是它要更新 $|S|*|A|$ 个状态-动作对的 Q 值和迹，其中 $|S|$ 表示状态总数，$|A|$ 表示动作总数，当状态-动作空间规模很大时计算量较大。因此，考虑一种折衷策略，利用将来 k 步的信息更新当前 Q 值，称为多步Q学习算法 MQ。当 $k=1$ 时，它退化为Q学习算法，$k\to\infty$ 时又演变为 $Q(\lambda)$ 算法，因此它是这两种方法的结合，同时在形式上又是两种方法的统一。

t 时刻的校正 n 步截断回报依然如(1)式所示，考虑到对于 $0\leqslant\lambda<1$，

$$(1-\lambda)\cdot\sum_{j=1}^{k-1}\lambda^{j-1}+\lambda^{k-1}=1 \tag{5}$$

令 $\sum_{j=1}^{0}\lambda^{j-1}=0$，则(5)式当 $k=1$ 时也成立。

定义 1 k 步的 $TD(\lambda)$ 回报定义为 $r_t^{(n)}$，$(n=1,2,3\cdots,k)$ 的加权平均值，即 $r_t^{\lambda,k}=(1-\lambda)\cdot\sum_{j=1}^{k-1}\lambda^{j-1}\cdot r_t^{(j)}+\lambda^{k-1}\cdot r_t^{(k)}$。

定理 2 对于定义 1 中定义的 $r_t^{\lambda,k}$，存在

$$r_t^{\lambda,k}=r_t+\gamma\cdot V_{t-1}(x_{t+1})+\sum_{i=1}^{k-1}(\lambda\gamma)^i\cdot[r_{t+i}+\gamma\cdot V_{t+i-1}(x_{t+i+1})-V_{t+i-2}(x_{t+i})] \tag{6}$$

定理 2 可以由数学归纳法证明。对(6)式做以下变换：

$$\begin{aligned}r_t^{\lambda,k}&=r_t+\gamma\cdot V_{t-1}(x_{t+1})+\sum_{i=1}^{k-1}(\lambda\gamma)^i\cdot[r_{t+i}+\gamma\cdot V_{t+i-1}(x_{t+i+1})-V_{t+i-1}(x_{t+i})]\\&\quad\sum_{i=1}^{k-1}(\lambda\gamma)^i\cdot[V_{t+i-1}(x_{t+i})-V_{t+i-2}(x_{t+i})]\end{aligned} \tag{7}$$

与 $Q(\lambda)$ 学习相似，同样忽略(7)式右边第二个求和式，e_t 和 e'_t 的意义如(2)和(3)，因此有

$$r_t^{\lambda,k}=e'_t+\sum_{i=1}^{k-1}(\lambda\gamma)^i\cdot e_{t+i}+Q_{t-1}(x_t,a_t)$$

所以

$$r_t^{\lambda,k}-Q_{t-1}(x_t,a_t)=e'_t+\sum_{i=1}^{k-1}(\lambda\gamma)^i\cdot e_{t+i}$$

那么多步Q学习的更新规则为

$$Q(x,a)=\begin{cases}Q_{-1}(x,a)+\alpha_t\cdot[e'_t+\sum_{i=1}^{k-1}(\lambda\gamma)^i\cdot e_{t+i}], & \text{如果 } x=x_t \text{ 且 } a=a_t\\ Q_{-1}(x,a), & \text{其它情况}\end{cases} \tag{8}$$

多步 Q 学习的算法总结如下。

Initialize $Q(x,a)$, $X[k]$, $A[k]$, $E[k]$ and $E'[k]$
Repeat(for each episode):
  Initialize x, $i=1$
  Repeat(for each step of episode):
    Choose a from x using policy derived from Q(e. g., ε-greedy)
    Take action a, observe r, y
    $e'=r+\gamma\cdot V(y)-Q(x,a)$
    $e=r+\gamma\cdot V(y)-V(x)$
    Update array X,A,E,E'
    If($i<k$)
      $Q(x,a)=Q(x,a)+\alpha_t\cdot e'$ ①
    else
      $\cdot\Delta=E'[0]+\sum_{m=1}^{k-1}(\lambda\gamma)^m\cdot E[m]$
      $\cdot Q(X[0],A[0])=Q(X[0],A[0])+\alpha_t\cdot\Delta$ ②
    $x\leftarrow y$, $i\leftarrow i+1$
  Until x is terminal

3.2 算法分析

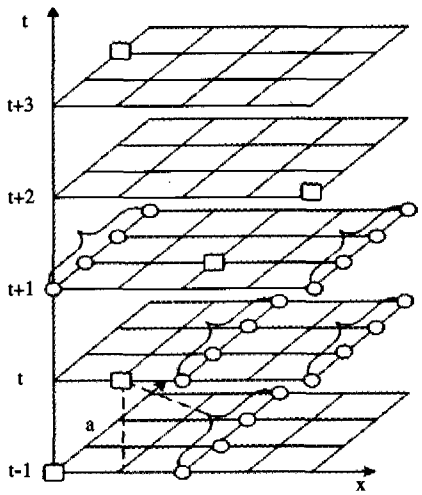

图 1 *MQ* 算法含义

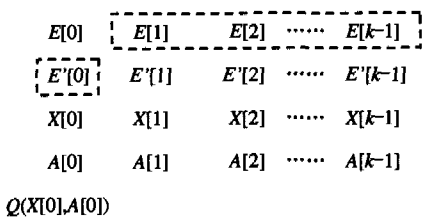

图 2 *MQ* 算法实现示意图

为了说明 *MQ* 学习方法的更新策略，考虑有 5 个状态、4 个动作的 MDP，且 *MQ* 算法中 k 取值 3，如图 1 所示。图中 x 轴表示状态，a 轴表示动作，t 轴表示更新时刻。在一个确定的时刻 t，方格上的点就表示状态-动作空间中的一个点，整个方格就表示状态-动作空间。方框表示 t 时刻状态-动作对的位置，大括号表示在同一状态的不同动作对应的 Q 值中取一个最大值，也就是该状态的 V 值，$V_t(x)=\max_{a\in A(x)}\{Q(x,a)\}$。状态-动作空间中的每一个点都对应一个 Q 值。Q 学习的更新规则的含义是在时刻 t，更新状态-动作对 (x_t,a_t) 的 Q 值时，要用到 (x_t,a_t) 在 $t-1$ 时刻的 Q 值，并且用到下一个时刻的状态 x_{t+1} 在 $t-1$ 时刻的 V 值。而 $k=3$ 时的 *MQ* 方法是在 Q 更新的基础上，再增加使用状态 x_{t+1} 和 x_{t+2} 在 t 时刻的 V 值，以及状态 x_{t+2} 和 x_{t+3} 在 $t+1$ 时刻的 V 值。由此可以看出，与 Q 学习相比，*MQ* 方法考虑了未来多步的决策的影响，因此具有更强的预见能力。$Q(\lambda)$ 学习是 *MQ* 方法的基础上，再增加考虑所有后续时刻状态-动作对的 V 值的影响，也就是时刻每变化一次，都要用当前状态的 V 值重新更新所有经历过的状态-动作对 Q 值。当然这种方法具有最强的预见能力，但是对所有状态-动作对 Q 值的持续更新是以增加计算量为代价的，当状态-动作空间规模很大时学习效率不高。因此，从总体上考虑，*MQ* 方法既能考虑将来 k 步的决策对当前 Q 值的影响，又能减少计算量，是一种有效平衡更新速度和复杂度的强化学习方法。

算法中用 4 个长度为 k 的一维数组 $X[k]$，$A[k]$，$E[k]$ 和 $E'[k]$ 存储 k 个步骤的状态、动作、e' 和 e。对这 4 个数组的更新，是将数据依次移到相邻的低下标的存储单元中，新得到的数据存入最大下标的存储单元中，如图 2 中箭头方向所示。这样，当 $i\geqslant k$ 时，更新语句如多步 Q 学习算法中②所示，总是更新 k 步之前的状态-动作对的 Q 值，对这个 Q 值的更新就考虑了将来 k 步的影响，如图 2 中虚线框所示。由于 Q 值的更新用到了将来 k 步的信息，因此在每个 episode 中的前 i 步 $(i<k)$ 直接使用 Q 学习的更新规则进行更新，如多步 Q 学习算法中①所示。

3.3 k 值的确定

由于每个 episode 中有多少 step 是不确定的，这和算法的收敛性有关，因此仅考虑每个 step 中的计算复杂度。在 *MQ* 算法中，由于在每个 step 中要更新 4 个长度为 k 的数组，因此计算复杂度为 $O(k)$。Q 学习和 $Q(\lambda)$ 算法的计算复杂度分别为 $O(1)$ 和 $O(|S|*|A|)$。由此可见，*MQ* 与 Q 学习和 $Q(\lambda)$ 算法的计算复杂度的比较和 k 的取值有关。

从智能体眼光的角度来讲，k 值越大，智能体能考虑到将来越远的信息，预见能力越强，这时的算法越接近 $Q(\lambda)$ 算法；k 值越小，智能体能利用的将来的信息越少，算法越接近 Q 算法，因此对智能体预见能力的要求使得 $k>1$。但是从计算复杂性角度来看，*MQ* 算法的计算复杂度为 $O(k)$，k 值越小计算复杂度越低。当 $k=1$ 时，具有最低的计算复杂度，也就是 Q 学习的复杂度。为了降低计算复杂度，要求 $k<|S|*|A|$。所以，要保持两方面的优越性，就要求 $1<k<|S|*|A|$。理论分析仅确定了 k 的取值范围，后面通过试验可以确定 k 的具体值。

4 仿真试验

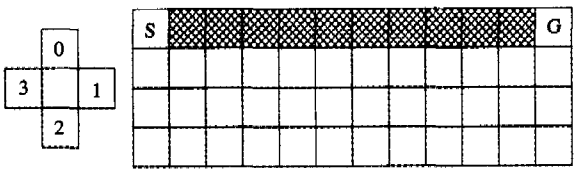

图 3 智能体悬崖步行仿真试验环境

悬崖步行是由 Sutton 提出的一个智能体仿真试验环境[7]，如图 3 所示。智能体的任务是从起始点 S 移动到目标点 G。系统的状态用智能体处于不同的方格来表示。智能体可以采取的动作有 4 个，即向上、下、左、右移动一个方格。智能体执行动作后，系统的状态转移是确定性的。S、G 之间的阴影方格为悬崖，智能体移动到这个区域就有坠崖的危险。因此，如果进入这个区域，就给它一个大的惩罚 $r=-10$，并且智能体回到起始点，该 episode 继续；如果到达 G，就给它一

个大的奖赏 $r=10$，同时该 episode 结束，智能体回到起始点，开始下一个 episode；其它情况，给它一个回报 $r=-0.1$。通过学习，智能体可以找到一条最安全或者最短的路径。

从迭代公式(8)可以看出，程序收敛速度受以下参数的影响：折扣因子 γ、学习率 α、权重参数 λ 和截断步数 k。这里不讨论折扣因子 γ、学习率 α 和权重参数 λ 的影响，分别取值为 $\gamma=0.9$，$\alpha=0.1$，$\lambda=0.85$。首先考虑 k 的变化对 MQ 算法收敛性的影响，然后在取定的 k 值下比较 Q、$Q(\lambda)$ 和 MQ 算法的收敛性。图 4、5、6 中横坐标为 episode 数，纵坐标为每个 episode 中 step 数。程序运行在 PIV3.0G 的计算机上。

4.1 k 的变化对 MQ 算法性能的影响

从图 4 可以看出，在 200 个 Episode 中，当 $k=3$ 时，算法不能收敛；当 $k=80$ 时，算法收敛性能较好。所以，取 $k=80$。

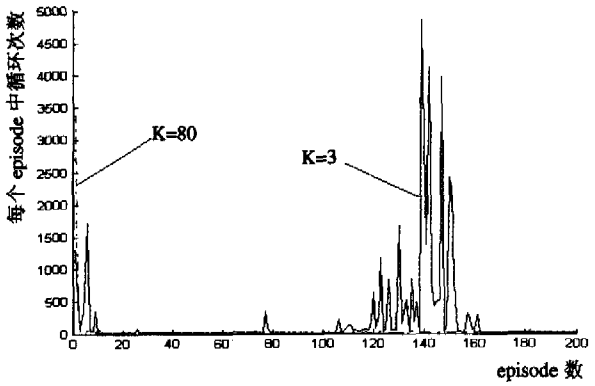

图 4 k 的变化对 MQ 算法性能的影响

4.2 在不同的状态空间规模下 Q、$Q(\lambda)$ 和 MQ 算法的比较

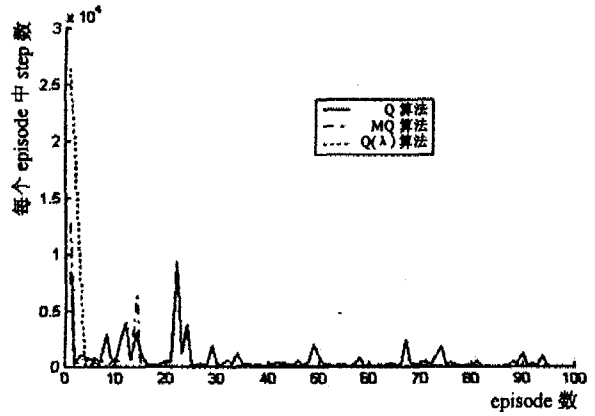

图 5 4×12 的环境网格下 Q、$Q(\lambda)$ 和 MQ 算法的比较

从图 5 和 6 可以看出，在 4×12 规模的环境网格下，3 种算法都可以很快收敛到最安全路径，只是 $Q(\lambda)$ 算法第一次到达目标循环的次数比较大。但是在 8×24 规模的环境下，经历了 100 个 episode 后，$Q(\lambda)$ 和 MQ 算法都收敛到了最安全路径，而 Q 算法仍然有一些振荡，速度不如其它两种算法快。从第一次到达目标循环的次数看，MQ 比 $Q(\lambda)$ 算法大，但是要比 $Q(\lambda)$ 算法小得多。从表 1 所示的程序运行时间看，MQ 比 $Q(\lambda)$ 算法长，但是也比 $Q(\lambda)$ 算法短。因此，总的来说，MQ 算法比 Q 算法收敛速度快，而计算速度又比 $Q(\lambda)$ 快，是对这两种算法的有效改进。

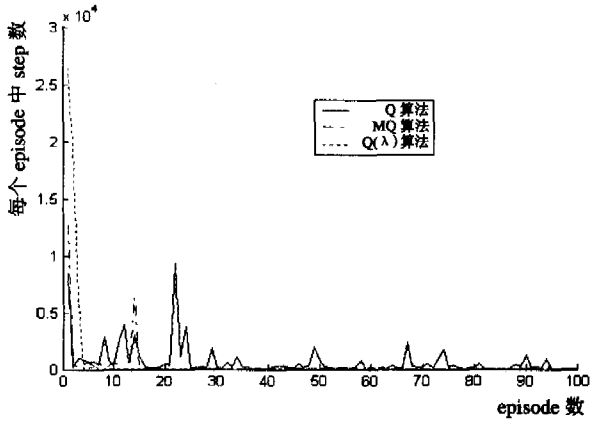

图 6 8×24 的环境网格下 Q、$Q(\lambda)$ 和 MQ 算法的比较

表 1 Q、$Q(\lambda)$ 和 MQ 算法运行时间的比较

算法	CPU 时间/100 个 episode(单位：秒)
Q	0.06
Q(λ)	11.60
MQ	0.76

结论 Q 学习是一种重要的强化学习算法。针对 Q 学习和 $Q(\lambda)$ 算法的不足，提出了 MQ 算法，它利用将来 k 步的信息更新当前的 Q 值，具有多步预见能力，又能降低计算复杂度，从而在更新速度与计算量之间取得了很好的平衡，同时在形式上又是两种方法的统一。试验表明，该算法收敛速度比 Q 算法快，计算速度又比 $Q(\lambda)$ 快，是对这两种算法的有效改进，因此具有良好的应用前景。

参 考 文 献

1 阎平凡. 再励学习-原理. 算法及其在智能控制中的应用. 信息与控制，1996，25(1)：28～34

2 赵志宏，高阳，骆斌，等. 多 agent 系统中强化学习的研究现状和发展趋势. 计算机科学，2004，31(3)：23～27

3 Watkins C J C H. Learning from Delayed Rewards：[PhD thesis]. University of Cambridge，1998

4 Watkins C J C H，Dayan P. Q-Learning. Machine Learning，1992(8)：279～292

5 Peng Jing，Williams R J. Incremental Multi-Step Q-Learning. Machine Learning，1996，22(4)：283～290

6 胡奇英，刘建庸. 马尔可夫决策过程引论. 西安：西安电子科技大学出版社，2000

7 Sutton R S，Barto A G. Reinforcement Learning：An Introduction. Cambridge，MA：MIT Press，1998