doi:10.3969/j.issn.1000-436x.2014.08.008

基于拓扑序列更新的值迭代算法

黄蔚[1]，刘全[1,2]，孙洪坤[1]，傅启明[1]，周小科[1]
(1. 苏州大学 计算机科学与技术学院，江苏 苏州 215006；2. 吉林大学 符号计算与知识工程教育部重点实验室，吉林 长春 130012)

摘 要：提出一种基于拓扑序列更新的值迭代算法，利用状态之间的迁移关联信息，将任务模型的有向图分解为一系列规模较小的强连通分量，并依据拓扑序列对强连通分量进行更新。在经典规划问题 Mountain Car 和迷宫实验中的结果表明，算法的收敛速度更快，精度更高，且对状态空间的增长有较强的顽健性。
关键词：强化学习；值迭代；拓扑序列；VI-TS
中图分类号：TP181 **文献标识码**：A **文章编号**：1000-436X(2014)08-0056-07

Optimized algorithm for value iteration based on topological sequence backups

HUANG Wei[1], LIU Quan[1, 2], SUN Hong-kun[1], FU Qi-ming[1], ZHOU Xiao-ke[1]
(1. School of Computer Science and Technology, Soochow University, Suzhou 215006, China;
2. Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China)

Abstract: In order to improve the convergence performance, an optimized value iteration based on topological sequence backups, VI-TS, is proposed. The key idea of VI-TS is to circumvent the problem of unnecessary backups by dividing an MDP into strongly-connected components and solving these components in topological sequences after detecting the structure of MDP. The experiment results show that VI-TS has a better convergence performance and robustness for state space growth when applied to classical planning experiment scenarios.
Key words: reinforcement learning; value iteration; topological sequence; VI-TS

1 引言

强化学习(RL, reinforcement learning)是机器学习的一个重要分支，它以在线学习为主要特征，通过 agent 与环境交互，用值函数来评价某个策略下每个状态或动作的好坏，最终确定到达目标的最优策略[1,2]。强化学习的数学理论基础建立在马尔可夫决策过程(MDP, Markov decision process)之上，所有决策都依赖其前面的决策及结果。有穷 MDP 在强化学习中具有非常重要的意义，它占到整个强化学习任务的 90%以上[3]。对于状态和动作空间连续的 MDP，通常都可以量化为有穷 MDP 来近似求解。

动态规划方法是求解模型已知的有穷 MDP 问题的最有效手段之一，其核心思想是用值函数来组织和构建策略搜索，按照“评估值函数→改进策略”交替进行的思想，逐步逼近最优策略。策略迭代和值迭代算法[4]是动态规划的 2 种重要方法，它们之间的主要区别是策略迭代在评估值函数的过程中需要精确计算出值函数后才改进策略，而值迭代算法并未等到值函数收敛就已经改进策略，也就是

收稿日期：2013-05-18；**修回日期**：2013-07-20
基金项目：国家自然科学基金资助项目(61070223, 61103045, 61272005, 61170020)；江苏省自然科学基金资助项目(BK2012616)；江苏省高校自然科学研究基金资助项目(09KJA520002, 09KJB520012)；吉林大学符号计算与知识工程教育部重点实验室基金资助项目(93K172012K04)
Foundation Items: The National Natural Science Foundation of China(61070223, 61103045, 61272005, 61170020); The Natural Science Foundation of Jiangsu Province (BK2012616); High School Natural Foundation of Jiangsu Province (09KJA520002, 09KJB520012); Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University(93K172012K04)

说，在得出值函数大小变化的趋势后就可以改进策略，最终仍能收敛至最优策略[5,6]。

然而，传统的值迭代算法尚存在诸多可以改进的地方，它的执行效率并不高。首先，它是在整个状态空间中穷举搜索，而不去管是否有更新的必要，每一次迭代都要更新所有状态的值。其次，并不是从初始状态出发能到达所有状态，而对这些不可达状态进行更新显然是毫无意义的。因此，有些算法从状态的可达性信息(reach-ability information)入手，利用启发式函数对 MDP 模型进行分析，跳过一些迭代过程中的无意义更新，例如 LAO*[7], LRTDP[8]和 HDP[9]等算法。另外，采用不同的更新顺序也会导致不同的收敛效率，因为 Bellman 公式的计算要依赖后续状态，所以优先计算后续状态的值函数显然是更为合理的选择，这就导致出现了各种不同的优先级扫描算法[7,10,11]，所有这些算法的基本原理都是使用优先级队列，通过设定与值函数变化相关的优先级函数来确定状态值函数的更新顺序。而使用优先级队列无疑增加了额外的时间和空间复杂度，其额外的时间复杂度是 $O(n\log n)$。

本文针对传统值迭代算法更新顺序无序的问题，提出一种不同于优先级扫描算法的新方法 VI-TS (optimized value iteration based on topological sequence backups)，该方法利用 MDP 模型内在的拓扑序列信息，并结合状态的可达性分析，在不增加额外的时间和空间开销的情况下，改善了传统值迭代算法的收敛速度慢、性能不稳定的问题。VI-TS 算法的基本原理是用有向图来表示一个 MDP 模型，然后利用启发式搜索技术剔除部分次优动作，分解出一系列规模较小的强连通分量，再按照拓扑序列计算每个强连通分量中的状态值函数，从而得到最优策略。实验结果表明，VI-TS 的收敛速度和精度都优于传统的值迭代算法及其他一些改进算法，且对状态空间的规模增长也有良好的顽健性。

2　背景知识

2.1　MDP 框架

MDP[5]是求解不确定环境中自主决策问题的模型框架，要求满足马尔可夫性。大部分强化学习任务都可以建模为一个 MDP 模型。

一个有穷 MDP 可用一个五元组 $M=<X, U, f, \rho, \gamma>$来表示，其中，$X$ 和 U 分别表示状态集和动作集，它们都是有穷的；$f: X\times U\times X\to[0,1]$是状态迁移函数，可用 $f(x,u,x')$来表示在状态 x 执行动作 u 后到达后继状态 x'的概率，即 $f(x,u,x')=\Pr\{x_{t+1}=x'|x_t=x,u_t=u\}$；$\rho: X\times U\to R$ 是奖赏函数，可用 $\rho(x,u)$表示在状态 x 执行动作 u 得到的奖赏。$\gamma\in[0,1]$是折扣率。

$V: X\to R$ 是状态值函数，$V(x)$表示 agent 从状态 x 出发能获得的累积奖赏的期望。有穷 MDP 决策问题的目标是使状态值 $V(x)=\sum_{t=0}^{T}\gamma^t R(x_t)$最大，其中，$T$ 是 agent 到达目标状态经历的时间步。

策略 $h: X\times U\to[0,1]$是状态 X 到动作 U 的映射，$h(x,u)$表示在状态 x 选择动作 u 的概率。任意策略 h 下的任意状态值函数 V 都满足

$$\forall x\in X:\quad V^h(x)=\sum_{u\in U}h(x,u)\cdot\left(\rho(x,u)+\gamma\sum_{x'\in X}f(x,u,x')V^h(x')\right)\tag{1}$$

也就是所谓的 Bellman 方程。动态规划的基本原理是利用 Bellman 方程迭代计算状态值函数，并根据值函数的大小来改进策略。

在所有策略中，最优策略用 h^*表示，它能指导 agent 以最短路径迁移到目标状态。最优状态值函数用 $V^*(x)$表示，它满足

$$\forall x\in X: V^*(x)=\max_{u\in U}\left(\rho(x,u)+\gamma\sum_{x'\in X}f(x,u,x')V^*(x')\right)\tag{2}$$

即 h^*可使 agent 从某个状态出发，以最短路径获得最大的累积奖赏，满足$\forall x\in X: V^{h^*}(x)\geqslant V^h(x)$。

2.2　值迭代算法

动态规划的本质是通过迭代计算方式求解 Bellman 方程。理论上，对于一个 MDP 模型已知的强化学习问题，利用式(1)中的 Bellman 方程就可以求解出 MDP 中所有的状态值函数，从而得到最优策略。

值迭代算法的一般流程如算法 1 所示，基本思路为：设置各个状态值函数的初始值，然后利用式(1)迭代计算值函数，逐步逼近真实值，当所有状态的更新误差 Δ 小于阈值 ε 时，停止计算。

算法 1　值迭代算法

1) Input X, ε;

2) Initialize $V(x)$;

3) Repeat:

　　$\Delta\leftarrow 0$;

for all $x \in X$

$v \leftarrow V(x)$;

Backup(x); //计算式(1)

$\Delta \max(\Delta, |v-V(x)|)$;

end for

Until $\Delta < \varepsilon$

从算法 1 可以看出，在更新状态值函数时算法并没有关注更新的顺序，但是由式(1)显然可知，对状态 x 的更新只有在 x 的后继状态更新之后才是有效的。下面以一个简单的 MDP 模型为例。

图 1 是一个 MDP 模型 M 的有向图表示 $G(X,E)$，其中 X 是节点集，E 是有向边集。一个节点代表一个状态，在图中用圆圈来表示；有向边代表 2 个状态之间的迁移关系，用一根带箭头的线来表示。以 x_1 为例，表示在状态 x_1 采取某个动作后可能以一定的概率迁移到状态 x_2、x_3、x_4，Goal 为目标状态。

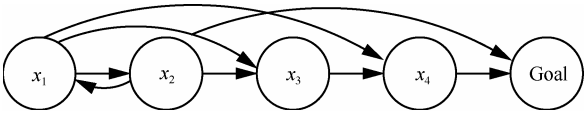


图 1 MDP 模型的有向图

根据式(1)，$V(x)$的计算依赖其后继状态的 $V(x')$，若所有后继状态的 $V^*(x')$已知，则对 x 更新 1 次就能得到 $V^*(x)$。例如在图 1 中，如果先计算 $V(x_4)$，由于 Goal 是已知的，则对 x_4 只需更新 1 次就能得到 $V^*(x_4)$；反之，若先计算 $V(x_3)$，由于 $V(x_3)$的计算依赖其后继状态 x_4，而 $V^*(x_4)$是未知的，因此需要多次迭代才能得到 $V^*(x_3)$。同理，其他节点的计算也是如此。得到 $V^*(x_4)$后，只需对 x_3 更新 1 次就能得到 $V^*(x_3)$，以此类推，因此最优的更新序列是 x_4, x_3, x_2, x_1。

3 VI-TS 算法

3.1 VI-TS 算法描述

由于值函数 $V(x)$的计算依赖其后继状态的值函数 $V(x')$，所以在更新 $V(x)$之前应先更新后继状态的 $V(x')$，而 $V(x')$的计算又依赖其后继状态 $V(x'')$，因此环环相扣，需要找出所有与 x 相关联的状态聚集，这些关联的状态聚集在有向图中正好是一个强连通分量。

下面以图 2 为例说明如何分解与优化一个 MDP 任务模型。图中共有 9 个状态，已标明各状态下可选择的动作及其迁移关系。

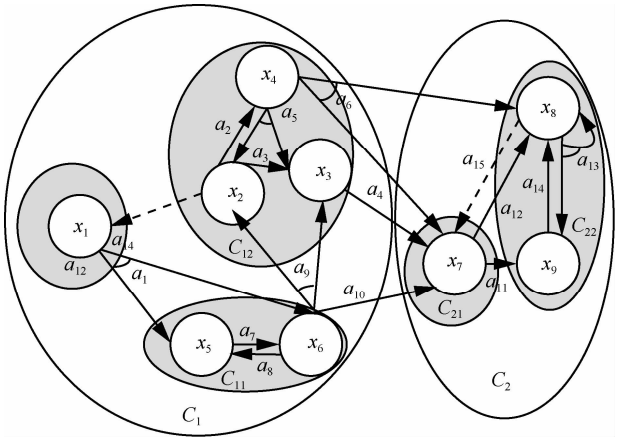


图 2 MDP 模型图结构以及强连通分量集合

首先任务模型被分解为 C_1、C_2 2 个强连通分量。根据实践经验，当模型分解为较多强连通分量且每个分量中的节点数较少时性能较好，因此下一步是继续优化。通常，最优策略很少同时包含 $u(x \rightarrow x')$和 $u(x' \rightarrow x)$，假设已知 $u(x \rightarrow x')$优于 $u(x' \rightarrow x)$，那么预先从策略中剔除 $u(x' \rightarrow x)$可以有助于问题的求解，其理论依据是 Bertsekas 中提出的定理 1。

定理 1 若 $Q_{ub}^*(x,u) < V_{lb}^*(x)$，则 u 不可能是状态 x 处的最优策略[11]。

不等式中的下标 ub 和 lb 分别表示值函数的上、下界。设计算法时，可以通过启发式信息设置值函数的初始值并进行更新。

图 2 中，假设已知 a_{14} 和 a_{15} 分别是状态 x_2、x_8 处的次优动作，那么从动作集中剔除两者后 C_1 分成 C_{11}、C_{12}、C_{13} 3 个分量，C_2 分成 C_{21}、C_{22} 2 个分量，5 个分量的拓扑序列是 C_{22}、C_{21}、C_{12}、C_{13}、C_{11}。依此顺序对所有强连通分量进行值迭代，可以高效地求出所有状态的最优值函数，从而得到最优策略。

VI-TS 算法按照“搜索→分解→计算”这 3 个主要步骤进行，如算法 2 所示，其中强连通分量的计算可用 Kosaraju 等算法实现。

搜索部分：利用启发式信息搜索 M 的状态动作空间，搜索迭代次数 t 的设置可以通过 $V_t(x_0)$的变化率 σ 进行控制，因为如果 t 过大就沦为纯粹的启发式搜索算法。每次迭代，状态值函数最多更新 1 次，同时需要更新值函数的上下界。值函数上届 V_{ub} 和 Q_{ub} 的初始值根据距目标状态的远近而定。例如，如果 x 最少需要 k 步能迁移到目标状态，则 $V_{ub}(x)=C-kr$，C 是根据问题规模而定的一个正数，应确保所有 $V_{ub}(x)>0$。V_{lb} 和 Q_{lb} 的初始值设定比较简单，例如，如果 $x \in G$，则 $V_{lb}(x)=0$；如果 $x \notin G$，则 $V_{lb}(x)=-1$。

分解部分：剔除部分非优动作，形成新的模型 M'。将模型 M 的有向图 G_M 分解为 k 个以拓扑顺序排列的强连通分量 $C_1,C_2,C_3,\cdots,C_k$，若有分量 C_i 包含的状态数量较多，可对该分量再进行 1 次内部搜索。

计算部分：按最后得到的拓扑顺序 $C_1,C_2,C_3,\cdots,C_k$ 计算强连通分量中所有状态的值函数。

算法 2　VI-TS 算法

1) Input $M<X,U,f,\rho,\gamma>,t,\sigma,\varepsilon$

2) Repeat　　//搜索

　　$V_{\text{old}}\leftarrow V_{\text{lb}}(x_0)$;

　　重复 t 次:

　　将每个状态都标记为 unvisited;

　　$x\leftarrow x_0$;

　　Search(x);

　　Until $V_{\text{old}}/V_{\text{lb}}(x_0)>(1-\sigma)$;

3) 将 M'分解成 k 个强连通分量: $C_1,C_2,C_3,\cdots,C_k$;

4) For $i\leftarrow 1$ to k

计算 C_i 中的状态值函数;

Search(x):

　　If $x\notin$ Goal

　　① x 标记为 visited;

　　② $u\leftarrow\arg\max_u Q(x,u)$;

　　③ for all $x'\in SPD(x)$

　　//$SPD(x)$表示 x 的后继状态集

　　Search(x')

　　④Backup(x)

Backup(x):

　　①for all $u\in$Action(x)

　　a) $Q(x,u)\leftarrow\rho(x,u)+\gamma\sum_{x'\in X}f(x,u,x')V_{\text{lb}}(x')$

　　b) If $Q(x,u)<V_{\text{lb}}(x)$

　　将 u 从 Action(x)中剔除

　　②更新 $V(x)$的上界：

　　$V_{\text{ub}}(x)\leftarrow\max_{u\in\text{Action}(x)}Q(x,u)$

　　③更新 $V(x)$的下界：$V_{\text{lb}}(x)\leftarrow\max_u\in$ Action(x)$[\rho(x,u)+\gamma\sum_{x'\in X}f(x,u,x')V_{\text{tb}}(x')]$

针对具体任务，还可以对算法做更进一步的优化，例如忽略模型中 agent 不可达状态区域的值函数更新。可达性分析可用深度优先搜索等算法实现，时间复杂度呈线性。当状态空间中 agent 从开始状态出发只能到达小部分区域时，优化效果尤为明显。

3.2　收敛性分析

VI-TS 算法利用 MDP 模型内在的图结构特征，优化了传统值迭代算法的值函数更新顺序，值迭代算法的最终收敛与状态的更新顺序无关[12]。VI-TS 算法本质上依然是值迭代算法，因此是收敛的。

定理 2　在 VI-TS 算法中，状态值函数以概率 1 收敛至最优。

证明　VI-TS 算法能在有限时间步内结束迭代。因为有限 MDP 模型 M 的状态维度$|X|$是有限的，所以 G_M 的强连通分量个数 k 也是有限的。更新每个分量 C_i 内部的状态值函数用的是优化的值迭代算法，值迭代算法在理论上是收敛的，所以 VI-TS 必然会在有限的时间步内结束迭代。

VI-TS 算法能保证值函数收敛至最优。VI-TS 算法在执行过程的任意时刻，当前更新分量中的状态值函数只依赖已更新过的分量或自身分量。当一个分量收敛后，其中的状态不再受之后更新的状态影响，即该分量中状态的值函数已是最优且不再变化。因此当更新完所有分量时算法收敛。

证毕。

4　实验验证

本文分别以二维迷宫和 Mountain Car 作为实验平台[3]，在参数设置相同的情况下，比较了 VI-TS 和其他对照算法的性能。

4.1　二维迷宫

如图 3 所示，二维迷宫中有一些障碍物(灰色格子)，要求 agent 从起始状态 S 出发，寻找到达目标状态 G 的最短路径。MDP 的状态是迷宫中格子的位置，动作是向上、下、左、右 4 个方向的移动；如果移动后的状态是障碍物或边界，agent 状态保持不变，否则迁移到相邻的格子。在到达目标状态前每一步状态迁移的立即奖赏均为 $r=-1$，迁移到目标状态 G 的立即奖赏 $r=0$。本文所用算法的参数设置为学习率 α=0.1，探索因子 ε=0.1，折扣率 γ=0.95。

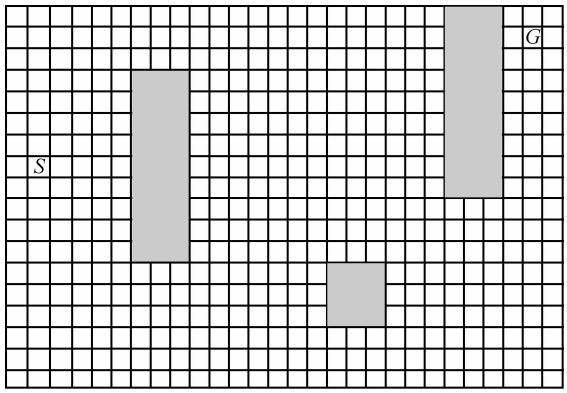


图 3　迷宫实验场景

表 1 分别列出了经典值迭代(VI, value iteration)算法、ILAO*算法、LRTDP 算法和 VI-TS 算法运行迷宫实验花费的平均时间（30 次独立实验的平均值）。使用的计算机 CPU 是 Intel(R) Core(TM) i3-2330 M 2.20 GHz，内存容量 2.0 GB。

表 1 4 种算法在不同规模迷宫实验中花费的时间/ms

算法	$Time_1$	$Time_2$	$Time_3$	$Time_{total}$
VI	—	—	—	1.636
ILAO*	—	—	—	0.472
LRTDP	—	—	—	0.298
VI-TS	0.011	0.007	0.002	0.019

为了统计的清晰，其中 VI-TS 专门列出了在搜索步、分解步和计算步各自所花费的时间 $Time_1$、$Time_2$ 和 $Time_3$。从实验数据可以看出在搜索步花费的时间较多，而计算步花费的时间较少，也就是在每个强连通分量中计算状态值函数花费的时间较少。

4.2 Mountain Car

Mountain Car 实验的目的是以最短的时间从山谷的最低点 S 运动到右端的最高点 G，山谷地形如图 4 所示。由于小车动力不足，无法直接从 S 冲上 G，必须通过反复来回振荡，最终到达最高点 G。MDP 的状态由小车的水平位移 y 和水平速度 2 个连续变量 v 组合而成，其约束条件为

$$\{(y,v)\in R^2 \mid -1.2\leqslant y\leqslant 0.5,-0.07\leqslant v\leqslant 0.07\} \quad (3)$$

其中，左端最高点 A 以及 S 和 G 点的水平位移 y 分别为−1.2、−0.5 和 0.5。动作为小车的油门方向，本文简化为全速向前、零油门和全速向后 3 个动作，分别用+1、0 和−1 表示。其系统动力学特性满足式(4)

$$\begin{cases}\dot{v}=\text{bound}[v+0.001u-g\cos(3y)]\\ \dot{y}=\text{bound}[y+\dot{v}]\end{cases} \quad (4)$$

其中，g=0.002 5 是一个与重力有关的系数，u 为动作值。立即奖赏 $r=\begin{cases}-1, & y<0.5\\ 0, & y\geqslant 0.5\end{cases}$。

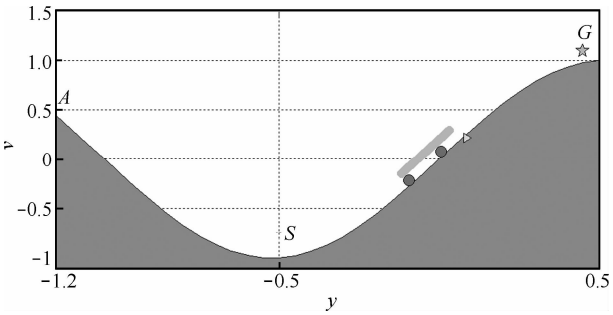


图 4 Mountain Car 实验示意

表 2 是 3 种不同规模的 Mountain Car 模型分别用 VI、ILAO*、LRTDP 和 VI-TS 算法所需的计算时间。

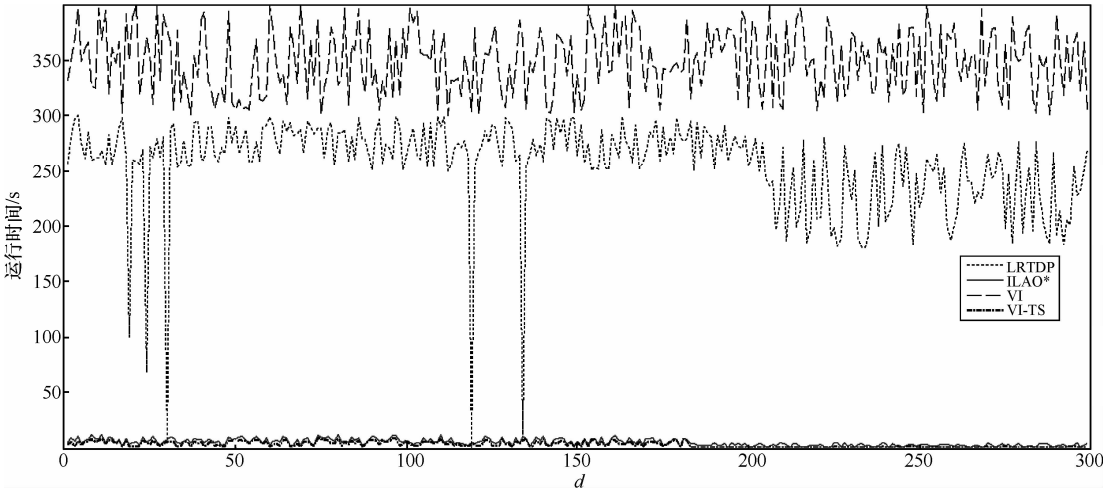
4.3 收敛性能分析

为了检测搜索深度对算法性能的影响，本文在 Mountain Car 实验平台上随机抽取了不同的起始状态位置。从起点到目标状态的最短路径距离就是距离目标状态的最短搜索深度，记为 d，单位是 agent 迁移的步数。图 5 描述了 4 种算法在状态空间为 300×300 的 Mountain Car 实验平台上的收敛性能曲线，其中图 5(a)是 4 种算法运行时间的全貌图，可以看出 VI 和 LRTDP 的收敛时间明显多于 ILAO*和 VI-TS 的收敛时间。为了清晰地比较 ILAO*和 VI-TS 的性能，图 5 (b)中放大显示了 ILAO*和 VI-TS 的性能曲线，从图中可以看出 VI-TS 算法的运行时间最短，运行速度比 VI 和 ILAO*快 1 个数量级，比 LRTDP 快 2 个数量级。

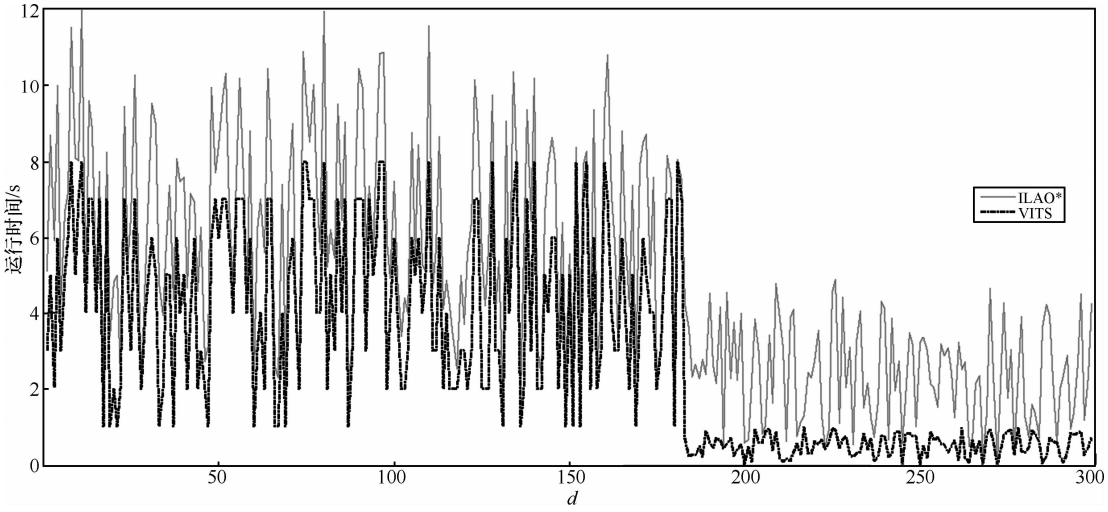
在搜索深度方面，VI 算法不受起始点设置的影响，因为 VI 没有搜索过程。LRTDP 算法在 d 较小时运行时间较短，当 d 变大时则收敛速率快速下降。ILAO*算法在 d 较小时收敛时间变化不大，而当 d 变大时则收敛时间也增加了。实验结果表明，启发式搜索算法 ILAO*和 LRTDP 的性能受 d 值的影响较大，运行时间随着 d 的增大而增加了至少 2 个数量级；VI-TS 算法则受 d 值的影响相对较小，运行时间仅仅随 d 的增大而增加了 1 个数量级左右。综合来看，VI-TS 算法的收敛性能优于其他 3 种对照算法。

表 2 4 种算法在不同规模 Mountain Car 问题中的运行时间/ms

状态空间维度	VI	ILAO*	LRTDP	VI-TS			
				$Time_1$	$Time_2$	$Time_3$	$Time_{total}$
100×100	10.25	1.95	1.31	0.21	0.01	0.02	0.24
300×300	>300	12.15	233.5	2.23	0.13	0.01	2.38
700×700	>300	102.55	>300	12.32	0.75	0.18	13.25



(a) 4 种算法的运行时间比较



(b) ILAO*和 VI-TS 的细节放大

图 5　Moutain Car 平台上 4 种算法的收敛性能比较

5　结束语

本文提出一种基于拓扑序列更新的值迭代算法 VI-TS，将 MDP 模型表示为一个有向图并分解出一系列强连通分量，然后按照拓扑序列计算各强连通分量内部的状态值函数，进而得到整个 MDP 模型的最优策略。虽然 VI-TS 算法搜索的计算时间较多，但是搜索过程可以分解任务模型的状态空间，降低问题的维度，且搜索过程利用启发式搜索技巧剔除了策略空间中的一部分次优动作，使得 VI-TS 在后续过程能高效计算每个分量中的状态值函数，从而改进传统值迭代算法的收敛速度。将 VI-TS 算法应用于二维迷宫和 Mountain Car 等经典强化学习规划问题的实验结果表明，VI-TS 算法的收敛速度更快，收敛精度也更高，且对状态空间的增长有较强的顽健性。如何优化搜索步骤以进一步提高收敛速度将是下一步工作的研究方向。

参考文献：

[1] 刘全, 傅启明, 龚声蓉等. 最小状态变元平均奖赏的强化学习方法[J]. 通信学报, 2011, 32(1): 66-71.
LIU Q, FU Q M, GONG S R, *et al*. Reinforcement learning algorithm based on minimum state method and average reward[J]. Journal on Communications, 2011, 32(1): 66-71.

[2] SZEPESVARI C. Algorithms for Reinforcement Learning[M]. San Rafael: Morgan Claypool, 2010.

[3] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction[M]. Cambridge: MIT Press, 1998.

[4] HOWARD R. Dynamic Programming and Markov Processes[M]. Cambridge, MA: MIT Press, 1960.

[5] BERTSEKAS D P. Dynamic Programming and Optimal Control[M]. Belmont, MA: Athena Scientific,2000.

[6] POWELL W B. Approximate Dynamic Programming: Solving the Curses of Dimensionality[M]. New York: John Wiley & Sons, 2007.

[7] HANSEN E, ZILBERSTEIN S. Lao*: a heuristic search algorithm that finds solutions with loops[J]. Artificial Intelligence, 2001, 129(1/2): 35-62.

[8] BONET B, GEFFNER H. Labeled RTDP: Improving the convergence of real-time dynamic programming[A]. Proc of 13th ICAPS[C]. Trento, Italy, 2003.12-21.

[9] BONET B, GEFFNER H. Faster heuristic search algorithms for planning with uncertainty and full feedback[A]. International Joint Conference on Artificial Intelligence[C]. 2003.1233-1238.

[10] MOORE A W, ATKESON C G. Prioritized sweeping: reinforcement learning with less data and less time[J]. Machine Learning, 1993, 13(1):103-130.

[11] ANDRE D, FRIEDMAN N, PARR R. Generalized prioritized sweeping[A]. Proc of the 10th Conference on Advances in Neural Information Processing Systems[C]. Cambridge, 1997.1001-1007.

[12] CORMEN T H, LEISERSON C E, RIVEST R L, *et al*. Introduction to Algorithms[M]. Cambridge, MA: MIT Press, 2001.

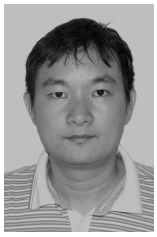
作者简介：

黄蔚（1970-），女，江苏海门人，苏州大学讲师，主要研究方向为机器学习。

刘全（1969-），男，内蒙古牙克石人，苏州大学教授、博士生导师，主要研究方向为强化学习、智能信息处理和自动推理。

孙洪坤（1988-），男，江苏淮安人，苏州大学硕士生，主要研究方向为强化学习。

傅启明（1985-），男，江苏淮安人，苏州大学博士生，主要研究方向为强化学习、贝叶斯推理和遗传算法。



周小科（1976-），男，江西上饶人，苏州大学讲师，主要研究方向为机器学习。