文章编号: 1001-0920(2002)03-0292-05

基于递阶强化学习的多智能体 AGV 调度系统

李晓萌, 杨煜普, 许晓鸣
(上海交通大学 自动化研究所, 上海 200030)

摘 要: 递阶强化学习是解决状态空间庞大的复杂系统智能体决策的有效方法。具有离散动态特性的 AGV 调度系统需要实时动态的调度方法, 而具有 MaxQ 递阶强化学习能力的多智能体通过高效的强化学习方法和协作, 可以实现 AGV 的实时调度。仿真实验证明了这种方法的有效性。

关键词: 递阶强化学习; MaxQ 方法; 多智能体协作; AGV 调度
中图分类号: TP 18 **文献标识码**: A

Multiagent AGV dispatching system based on hierarchical reinforcement learning

LI Xiao-meng, YANG Yu-pu, XU Xiao-ming
(Institute of Automation, Shanghai Jiaotong University, Shanghai 200030, China)

Abstract: Hierarchical reinforcement learning is an effective method of solving decision problems for complex systems with enormous number of states. AGV dispatching system needs dynamic dispatching rules because of its discrete and dynamic properties. Multiagent with the capacity of MaxQ hierarchical reinforcement learning is implemented in real-time AGV dispatching by high performance learning and cooperation. The simulation testifies the efficiency of this method.

Key words: hierarchical reinforcement learning; MaxQ method; cooperative multiagent; AGV dispatching

1 引 言

强化学习是近年来机器学习和智能优化控制的一个重要研究方向, 许多学者对此进行了研究[1, 2]。但强化学习应用到复杂系统决策中却遇到了很大困难。原因在于该方法需要搜索的状态空间通常非常庞大, 求得优化的行动策略非常缓慢, 所以很难将其用于实时决策问题。递阶强化学习正是为了解决这一问题而提出的[3]。由 Dietterich 最近提出的 MaxQ 方法[4]是基于任务分解来设计的递阶强化学习方法。本文以该方法为基础, 研究具有分布式特性的多智能体强化学习, 并以一个多 AGV (Automated Guided Vehicles) 调度的仿真来证明这种多智能体递阶强化学习的有效性。

2 MaxQ 递阶强化学习

Dietterich 提出的 MaxQ 方法一方面确保了各个子任务都是 MDP (Markov Decision Problem), 一方面可将强化学习的奖赏函数与值函数进行相应分解, 并与各子任务 MDP 相关联, 使得各子任务都能

收稿日期: 2000-12-26; **修回日期**: 2001-05-28

作者简介: 李晓萌(1975—)。男, 四川绵阳人, 博士生, 从事分布式智能控制、智能交通系统等研究; 许晓鸣(1957—), 男, 上海人, 副校长, 教授, 博士生导师, 从事复杂系统的智能控制等研究。

通过强化学习得到最优策略。因此MaxQ 方法是一种适合于大规模复杂调度问题的方法。

MaxQ 方法的原理是: 根据复杂MDP 问题中各子问题的相互独立性和状态空间的可分割性, 将问题和状态空间分解成各个层次的子MDP 问题和相应子状态集合, 并利用上述分解构成递阶学习结构。根据这一递阶结构, 智能体可以通过强化学习寻找到每一个子MDP 问题的最优策略。由这些优化的策略递阶地构成整个问题的优化策略。下面我们从递阶结构、递阶策略的优化和学习算法 3 个方面分析MaxQ 方法。

2.1　递阶学习结构

图 1 给出了MaxQ 递阶结构图, 其中三角形的节点称为Max 节点。当该节点是叶节点时, 它表示整个问题域内的基本动作, 如图中的$a_i(i=1,2,3)$; 否则, 该节点表示一个子任务(根节点表示为整个MDP), 如Max-$M_i(i=1,2,3)$。Q 节点表示为完成父节点任务所采取的动作, 即由Q 值确定的完成上一级子任务的最佳动作。

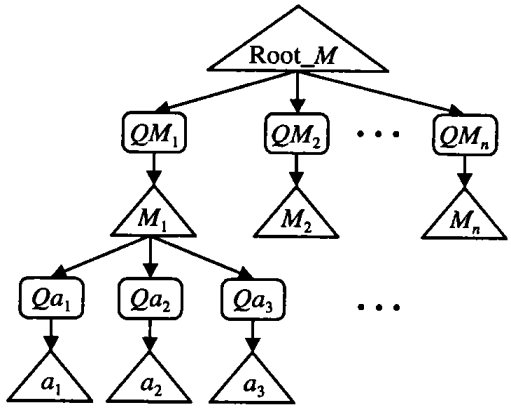

图 1　MaxQ 递阶结构示意

MaxQ 方法是一个通用的强化学习方法, 其形式化描述为: 假定一个MDP M 是定义在状态集合S、具有奖赏函数$R(s'|s,a)$ 的动作集合A 和状态转移概率函数$P(s'|s,a)$ 上的优化决策问题。将M 分解成相互独立的若干个子问题, 由这些子问题构成集合$\{M_i, i=1,2,\cdots,n\}$。其中每个M_i 都是一个MDP, 将它定义为一个 4 元组(S_i, T_i, A_i, R_i)。其中, S_i 为M_i 的状态集合; T_i 为M_i 的终止状态集, 当状态变迁到集合T_i 中时M_i 的求解过程结束; A_i 为M_i 问题允许的行动集合, A_i 既可以是M 的基本行动集合, 也可以是子任务集合; $R_i(s'|s,a)$ 为从状态$s\in S_i$ 变迁到状态$s'\in T_i$ 的奖赏函数, 它表明了上级子任务对在状态s 采取行动a 对于到达终止状态s'的期望程度, 即该子任务对各个终止状态的偏好。

2.2　递阶策略的优化

MaxQ 图的递阶策略是指策略集合$\pi=\{\pi_0,\pi_1,\cdots,\pi_n\}$。其中$\pi_i$ 是M_i 的策略, 它从根节点任务M_0 开始递阶地扩展, 直至基本动作被执行。若每个策略π_i 在M_i 的所有策略中都是优化的, 则策略π 是优化的策略[4]。MaxQ 方法将MaxQ 图解释为递阶策略的值函数表示。定义$V_i^{\pi}(s)$ 为执行策略π 从状态s 到M_i 的某个终止状态的期望累积奖赏, 即策略的值函数。对一个固定的递阶策略π, 节点i 执行动作a 的即时奖赏, 即节点i 从当前状态依据策略π_i 到达T_a 所得到的期望奖赏, 可表示为

$$V_i^{\pi}(s)=V_a^{\pi}(s)+\sum_{s'}P(s'|s,a)V_i^{\pi}(s') \tag{1}$$

其中, $a=\pi_i(s)$, $P(s'|s,a)$ 表示M_i 执行a 后状态从s 迁移到s'的概率。式(1) 给出了策略值函数的递阶分解, 这样根节点的值函数便表达了整个M 的值函数, 而每个M_i 都是一个MDP。但值函数是未知的, 因此需要构造对值函数的逼近。依据Q 学习算法的迭代结构[5], 定义$Q_i^{\pi}(s,a)$ 为M_i 在状态s 执行a, 并随着执行递阶策略π 后的期望累积奖赏。由此迭代结构并由式(1) 可得

$$Q_i^{\pi}(s,a)=V_a^{\pi}(s)+C_i^{\pi}(s,a) \tag{2}$$

$$V_i^{\pi}(s)=\begin{cases}Q_i^{\pi}(s,\pi_i(s)), & i\text{ 为非叶节点}\\ \sum_{s'}P(s'|s,a)R(s'|s,a), & i\text{ 为叶节点}\end{cases} \tag{3}$$

$$C_i^{\pi}(s,a)=\sum_{s'}P(s'|s,a)V_i^{\pi}(s') \tag{4}$$

其中, $C_i^{\pi}(s,a)$ 为完成函数, $R(s'|s,a)$ 为定义在叶节点上的即时奖赏。式(2)～(4) 给出了MaxQ 递阶结构的值函数逼近。每个Q 节点(其父节点为i, 子节点为a) 存储了S_i 中每个状态的信息$C_i^{\pi}(s,a)$, 而每个Max 节点i 则根据π_i 所得的Q 值返回子节点。

2.3　学习算法

学习算法的目的在于从局部优化的策略中选取对全局目标最为有效的策略构成递阶策略。算法要保证子策略的终止状态都在目标状态集内, 因此应对到达非目标的终止状态的子策略进行惩罚。较好的办法是对每个合成Max 节点的M_i, 对其能够到达目标状态的策略和可以到达终止状态但非目标状态的策略进行区别, 也就是对后者增加惩罚函数项。所以每个合成Max 节点需要维护两张表$C_i(s,a)$ 和$\tilde{C}_i(s,a)$。算法选择要执行的动作, 根据当前的探索策略执行动作a, 观察结果状态s'和奖赏$R_i(s'|s,$

a), 求出下一步最优动作

$$a^* = \arg\max_{a'}[\widetilde{C}_i(s', a') + V_{a'}(s')] \tag{5}$$

$$\widetilde{C}_i(s,a) = (1-\alpha(i))\widetilde{C}_i(s,a) + \alpha(i)\cdot[\widetilde{R}_i(s') + R_i(s'|s,a) + \widetilde{C}_i(s,a^*) + V_{a^*}(s')] \tag{6}$$

$$C_i(s,a) = (1-\alpha(i))C_i(s,a) + \alpha(i)\cdot[R_i(s'|s,a) + C_i(s,a^*) + V_{a^*}(s')] \tag{7}$$

其中, a^* 为根据当前 $\widetilde{C}$ 值和 V 值确定的在状态 s 的最优动作, $\widetilde{R}_i(s')$ 为附加的负奖赏(惩罚)函数, $R_i(s'|s,a)$ 为将整个奖赏函数分解到各个 Q 节点中的奖赏, $\alpha(i)$ 为节点 i 在每一步 t 的学习率。

对于节点 i, 执行策略后一旦到达终止状态(但非目标状态), 则 M_i 结束, 需要将 C_i 值向上传递, 但显然此时的策略对于 M_i 并非优化策略, 所以需要增加负奖赏 $\widetilde{R}_i(s')$ 以求得能够到达目标状态的策略。由 $\widetilde{C}_i$ 来求得优化策略, 而 $\widetilde{R}_i(s')$ 仅是 M_i 的局部函数, 所以在向上级节点传递完成函数值时, 仍然传递 C_i。

3 多AGV 调度问题

多AGV 系统是一个离散事件动态系统, 它是FMS 中一个重要的组成部分。AGV 的主要工作流程是将零件和原材料从仓库运送到车间的加工台缓冲区, 并将加工后的零件运送到仓库。AGV 的调度就是在AGV 空闲或加工台发出运送请求时, 合理地安排各AGV 的工作流程。多AGV 调度面临两个主要问题[6]: 1) 加工台存放加工后零件的缓冲区已满发出运送请求时, 如果此时空闲的AGV 不止一辆, 则需要选择完成该服务的AGV; 2) 当一台AGV 同时接收到多个运送请求时, 它需要决定首先应完成哪个运送服务。此外, 调度还会受到AGV 运送能力、当前状态、加工台缓冲区零件数、加工过程等条件约束。因此, 多AGV 调度问题需要建立动态的调度规则。在已有的研究中, 通常采用各种启发式方法及其组合来解决多AGV 的调度问题[7, 8]。但当AGV 的动态变化因素较多时, 这些方法的效果很不理想。将智能体的强化学习方法用于为各个AGV 建立动态的调度规则, 动态变化的环境即为新的状态空间, 智能体通过递阶强化学习能在动态变化的系统环境下建立新的调度策略。

这里采用Jim Lee 给出的AGV 调度实例。如图2所示, AGV 从仓库的Parts 处向4台加工机器(M_i, i = 1, 2, 3, 4) 运送4种待加工零件 $Part_i$。其中 $Part_i$ 送往 M_i, 并将加工完的零件 $Assembly_i$ 送回仓库的Assemblies 处。AGV 的运送路径是单向的, 如图中箭头所示。AGV 调度的主要目标是最大化加工完零件的产出, 即单位时间内Assemblies 处的零件输出量。

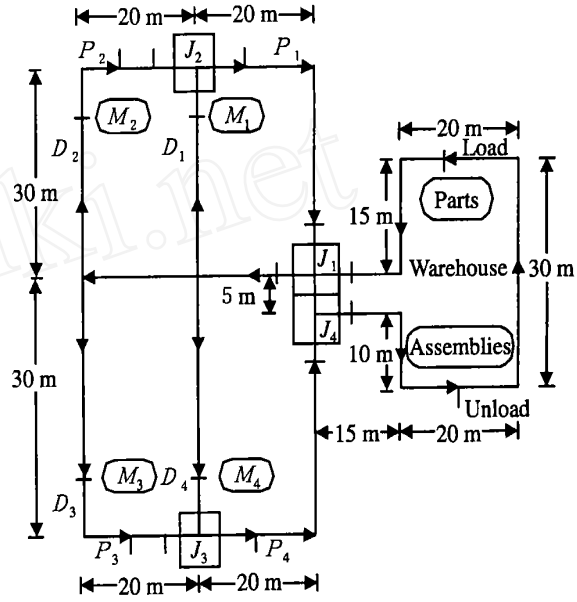

图2 加工车间的AGV 调度路径

4 多智能体递阶强化学习方法

我们将递阶强化学习多智能体系统应用于多AGV 优化调度。智能体采用递阶强化学习具有显著的优点: 1) 通过对动作粒度的划分, 使智能体能对各个层次的子任务进行学习, 建立完成当前子任务的优化策略, 而对上层的策略学习则可以忽略下层子任务的学习过程, 从而使下层的子任务可以被重用; 2) 因为对当前子任务策略的学习, 可以忽略与该子任务无关的状态特征, 所以需要学习的状态空间显著减小, 学习速度加快, 并将递阶强化学习引入多智能体系统; 3) 通过智能体间的协调和协作, 一方面共享相互间的优化任务策略, 加快系统的决策调度过程; 另一方面, 智能体只需通过了解其它智能体当前从事的高层子任务, 就可以调整自身的任务策略, 以避免任务冲突。上述两方面确保了分布式系统的实时优化调度。

在递阶强化学习多智能体系统中, 所有智能体均具有相同的递阶学习结构, 都采用MaxQ 方法进

行策略学习。智能体之间只需要在高层子任务上进行相互协作而屏蔽各自为完成高层子任务的学习过程, 因而可以增加智能体在高层子任务的协作, 共享底层子任务的状态和策略等信息, 并且避免在底层策略协调时出现冲突。比如图 1 中, 假设智能体 1 已具有完成子任务M_1的优化策略, 则当智能体 2 需要完成M_1时, 便可从智能体 1 中直接得到优化策略, 而不必重新进行学习。此外, 当智能体 1 得知智能体 2 正在完成任务M_1时, 它可以调整自己的任务策略而不会与智能体 2 的任务发生重叠。

上节给出的AGV 调度问题是一个MDP 问题。第 1 级子任务分解为ST1 = {TM$_i$, TA$_i$, i = 1, 2, 3, 4}, TM$_i$ 和 TA$_i$ 分别表示将零件 i 送到机器M_i和送到Assemblies 处, 第 1 级的各个子任务分别需要完成 4 个第 2 级子任务ST2 = {GoToLoad, GoToPut, Load, Put}; 而第 2 级的子任务由基本动作序列构成, 这些基本动作即为MaxQ 图的叶节点。基本动作定义为AGV 在每个位置处的动作{Forward, Right, Left, Stay}。

在这个调度问题中, 多智能体的协作及递阶强化学习过程如图 3 所示。智能体的递阶强化学习分为两级: 任务级和行动级。

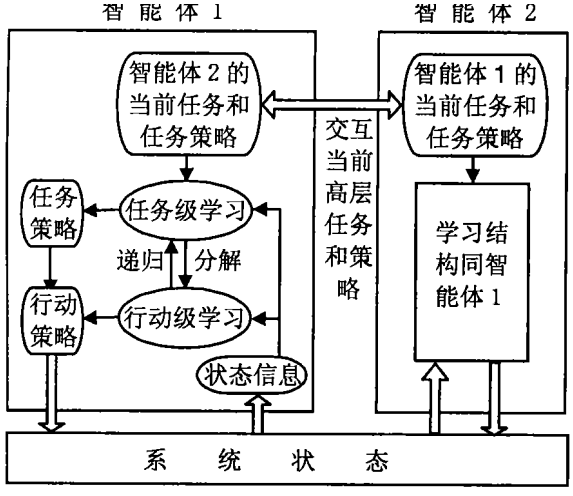

图 3　多智能体协作学习

图中智能体 1 和智能体 2 的高层任务和策略交互是指：两个智能体相互告知对方自己正在进行的子任务和任务策略, 而将完成各自子任务的细节屏蔽。智能体在任务级的学习需要根据对其它智能体的任务信息来排除一些不合理的任务策略, 以避免与其它智能体的任务冲突。在行动级的学习中, 智能体学习的结果是得到完成第 2 级子任务的行动策略, 即到达目标位置的最短路径。

5　仿真研究

这里对图 2 所示的AGV 调度问题进行仿真研究。表 1 给出了AGV 调度系统的各项参数。仿真中状态空间的每个状态由下列元素组成: AGV 位置、装载零件状况、加工机器的输入、输出缓冲区以及加工状况、Parts 处的 4 种零件到达状况。我们对图 2 中的路径设置 20 个AGV 的位置点, AGV 装载状态有 9 种情况, 对于缓冲区的状态, 仅以满和空来表示, 则每台机器的状态有 4 种情况, Parts 的零件缓冲区有 2^4 = 16 种状态。这样, 每个智能体所需探索的状态数约为$2^{19}(20 \times 9 \times 4^4 \times 2^4)$ 个, 这对于强化学习是一个巨大的状态数。而采用MaxQ 递阶方法, 每个子任务 ST1 所需探索的状态数仅为 $20 \times 9 \times 4 \times 2$ = 1 440 个, 大大减少了探索的状态数。对于 ST1 的选择, 只需根据得到的完成函数 C 的值以及其它智能体当前进行的 ST1 任务来选择将执行 ST1 的任务。

表 1　AGV 调度问题参数

Parts 处的零件到达平均时间 /s	8
到达零件比例(P_1 : P_2 : P_3 : P_4)	20 : 28 : 22 : 30
机器的零件缓冲区(输入; 输出)	10; 10
机器加工时间(M_1;M_2;M_3;M_4)/s	15; 24; 24; 30
零件装载时间 /(s/个); 下载时间 /(s/个)	6; 3
AGV 的最大装载量 /个	10

我们用 4 个AGV (相应于 4 个智能体) 完成上述调度任务, 并对两种情况的多智能体系统进行仿真。一种是具有协作能力的多智能体, 每个智能体采用MaxQ 方法进行递阶强化学习, 在进行学习前智能体需要了解其它智能体的当前行为; 另一种是独立学习的多智能体, 采用一般的Q-学习算法建立行动策略, 但相互间不关心其它智能体的当前行动。分别对两种情况进行 10 次仿真, 并对单位之间加工零

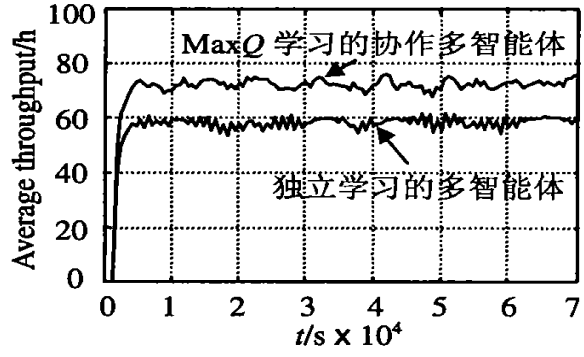

图 4　多智能体AGVs 调度仿真

件后的产量求平均值,得出如图 4 所示的产量曲线。比较两种方法对加工零件产量的影响可以看到,具有递阶强化学习的协作多智能体系统具有更好的调度效果。

6 结 语

采用递阶强化学习方法研究状态空间庞大的复杂系统的智能优化控制是一种比较有效的方法。具有离散动态特性的AGV 调度系统需要实时动态的调度方法,而具有MaxQ 递阶强化学习能力的多智能体通过高效的强化学习方法和协作,可以实现AGV 的实时调度。关于如何建立与高层子任务相应的抽象状态空间,建立基于递阶强化学习的AGV 调度仿真平台等问题,还有待于进一步研究。

参考文献(References):

[1] T G Dietterich. Machine learning research: Four current directions[J]. Artificial Intelligence Magazine, 1997, 18(4): 97-136.

[2] C Claus, C Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems[A]. Proc of the 10th AAAI[C]. Wisconsin: Madison, 1998. 746-752.

[3] L Kaelbling. Hierarchical reinforcement learning: Preliminary results[A]. Proc of the 10th ICML[C]. San Francisco: Morgan Kaufmann, 1993. 167-173.

[4] T Dietterich. The MAXQ method for hierarchical reinforcement learning[A]. Proc of the 15th ICML[C]. San Francisco: Morgan Kaufmann. 1998. 118-126.

[5] C J Watkins. Learning from delayed rewards[D]. Cambridge: Kings College, 1989.

[6] J Bartholdi, L Platzman. Decentralized control of affixed route automatic guided vehicle system[J]. IIE Transactions, 1989, 21(1): 76-81.

[7] J Lee. Composite dispatching rules for multiple-vechile AGV system[J]. Simulation, 1996, 66(2): 121-130.

[8] C Klein, J Kim. AGV dispatching[J]. Int J of Production Research, 1996, 34(1): 95-100.

(上接第 291 页)

7 结 语

本文首先提出设计连续变增益H_∞控制器的新方法。对应不同运动区域的线性化系统,利用具有极点配置的状态反馈H_∞技术,设计了满足H_∞性能和动态特性的状态反馈增益,并通过泰勒级数展开拟合成与运动点有关的连续函数。随着系统状态的变化,拟合成的增益函数可使控制器获得连续的增益,克服了传统变增益控制器的不足,使得所设计的控制器同样适合系统状态变化快的对象。

因为单纯连续变增益H_∞控制器不能使其阶跃响应既具有快速特性又具有良好的阻尼特性,所以本文又提出了模糊连续变增益H_∞控制器,通过模糊控制的引入,使控制器在误差较大时具有快速特性而误差较小时又具有良好的阻尼特性,从而使系统随状态变化始终具有很高的动态性能。在控制器设计中,利用LMI技术得到了满足要求的解。仿真与实验验证了本文设计的模糊连续变增益H_∞控制器的有效性和先进性。

参考文献(References):

[1] Shamma J S, Athans M. Analysis of nonlinear gain scheduled control systems[J]. IEEE Trans on Automatic Control, 1990, 35(8): 898-907.

[2] Shamma J S, Athans M. Gain scheduling: Potential hazards and possible remedies[J]. IEEE Control System Magazine, 1992, 12(3): 101-107.

[3] Jiang J. Optimal gain scheduling controller for a diesel engine[J]. IEEE Control System Magazine, 1994, 14(4): 42-48.

[4] Chilali M, Gahinet P, Apkarian P. Robust pole placement in LMI regions[J]. IEEE Trans on Automatic Control, 1999, 44(12): 2257-2269.

[5] Gahinet P, Apkarian P. A linear matrix inequality approach to H_∞ control[J]. Int J of Robust and Nonlinear Control, 1994, 4(3): 421-448.

[6] 虞忠伟, 陈辉堂. 基于滑模观测器的摩擦力自适应补偿方案[J]. 机器人(Robot), 1999, 21(7): 562-568.

[7] Gahinet P, Arkadii N, Laib A J, et al. The LMI control toolbox[A]. Proc of 33rd Conf on Decision and Control[C]. Florida, 1994. 2038-2041.