参考文献:

[1] CHONGJIE Z, LESSER V, SHENOY P. A multi-Agent learning approach to resource sharing across computing clusters[R]. Computer Science Department, University of Massachusetts Computer Science Amherst UMass, UM-CS-2008-035, 2008.

[2] KO P C, LIN P C, YOU J A, et al. Multi-layer allocated learning based neural network for resource allocation optimization[C]// Proceedings of the 9th Joint Conference on Information Sciences(JCIS 2006). Taibei, China, 2006: 35-41.

[3] TESAURO G. Online resource allocation using decompositional reinforcement learning[C]//Proceedings of AAAI 2005. Pittsburgh, USA, 2005: 886-891.

[4] LITTMAN M L, STONE P. Leading best-response strategies in repeated games[C]//The 17th Annual International Joint Conference on Artificial Intelligence Workshop on Economic Agents, Models, and Mechanism. Seattle, Washington, USA, 2001: 745-756.

[5] HU J, WELLMAN M P. Multiagent reinforcement learning in stochastic games[OL]. Citeseer. ist. psu. edu/hu99multiagent. Html, 1999.

[6] BUSONIU L, De SCHUTTER B, BABUSKA R. Multiagent reinforcement learning with adaptive state focus[C]//Proceedings of the 17th Belgium-Netherlands Conference on Artificial Intelligence. Brussels, Belgium, 2005: 35-42.

[7] KOK J R, VLASSIS N. Collaborative multiagent reinforcement learning by payoff propagation[J]. Journal of Machine Learning Research, 2006, 7: 1789-1828.

[8] 杨佩,陈兆乾,陈世福. 机器学习在RoboCup中的应用研究[J]. 计算机科学, 2003, 30(6): 118-121.
YANG Pei, CHEN Zhaoqian, CHEN Shifu. RoboCup multi-Agent system machine-learning[J]. Computer Sciences, 2003, 30(6): 118-121.

[9] 王醒策,张汝波,顾国昌. 基于强化学习的多机器人编队方法研究[J]. 计算机工程, 2002, 28(6): 15-16.
WANG Xingce, ZHANG Rubo, GU Guochang. Research on multi-Agent team formation based on reinforcement learning[J]. Computer Engineering, 2002, 28(6): 15-16.

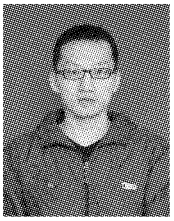
[10] HU J, WELLMAN M P. Nash Q-learning for general-sum stochastic games[J]. Journal of Machine Learning Research, 2003, 4: 1039-1069.

[11] ALPAYDM E. 机器学习导论[M]. 范明,等译. 北京: 北京工业出版社, 2009: 244-255.

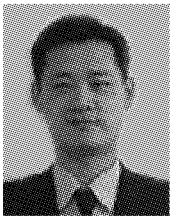
[12] LAGOUDAKIS M G, PARR R. Least-squares policy iteration[J]. Journal of Machine Learning Research, 2003(4): 1107-1149.

[13] XU X, HU D W, LU X C. Kernel based least-squares policy iteration[J]. IEEE Transactions on Neural Networks, 2007, 18(4): 973-992.

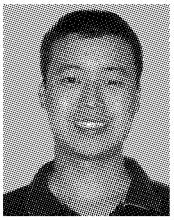
作者简介:



连传强,男,1986年生,硕士研究生,主要研究方向为模式识别与机器学习.



徐昕,男,1974年生,研究员,博士,主要研究方向为增强学习、自适应动态规划理论和算法、智能移动机器人、智能系统.



吴军,男,1980年生,博士研究生.主要研究方向为多机器人系统、机器学习与智能系统.

doi:10.3969/j.issn.1673-4785.2011.02.001

面向资源分配问题的Q-CF多智能体强化学习

连传强,徐昕,吴军,李兆斌
(国防科技大学 机电工程与自动化学院,湖南 长沙 410073)

摘 要:多智能体强化学习算法在用于复杂的分布式系统时存在着状态空间大、学习效率低等问题.针对网络环境中的资源分配问题对多智能体强化学习算法进行了研究,将Q-学习算法和链式反馈(chain feedback,CF)学习算法相结合,提出了Q-CF多智能体强化学习算法,利用一种称为信息链式反馈的机制实现了多智能体之间的高效协同.仿真结果表明,和已有的多智能体Q-学习算法相比,该方法具有更加快速的收敛速度,同时保证了协同策略的性能优化.

关键词:多智能体系统;强化学习;资源分配;协同控制
中图分类号:TP391.1 **文献标识码**:A **文章编号**:1673-4785(2011)02-0095-06

Q-CF multi-Agent reinforcement learning for resource allocation problems

LIAN Chuanqiang, XU Xin, WU Jun, LI Zhaobin
(College of Mechatronics and Automation, National University of Defense Technology, Changsha 410073, China)

Abstract: When a multi-Agent reinforcement learning algorithm is used in complex distributed systems, problems such as huge state space and low learning efficiency arise. In this paper, a multi-Agent reinforcement learning algorithm was studied for the resource allocation problem in a network environment. By combining the Q-learning algorithm and the chain feedback learning mechanism, a novel Q-CF multi-Agent reinforcement learning algorithm was presented. In the Q-CF algorithm, multi-Agent cooperation was realized based on the mechanism of information chain feedback. Simulation results show that compared with the multi-Agent Q-learning algorithm in existence, the proposed algorithm in this paper has a faster convergence speed while at the same time ensures the performance optimization of cooperation policy.

Keywords: multi-Agent system; reinforcement learning; resource allocation; cooperation control

近些年来,在网络环境中的资源分配问题因为其广泛的应用,如网络服务、传感器网络等,所以受到越来越多的关注.它的特点在于,在网络环境中,资源实现了共享,因而能够满足更多的应用需求.然而随着需求的不断增大,网络环境中的资源分配问题规模也越来越大,因此如何合理分配资源以优化系统的性能,提高系统的效率是亟待解决的问题.许多学习算法已经被应用到资源分配问题中[1-3],其中机器学习在资源分配问题中的应用是当前的一个研究热点.

强化学习(reinforcement learning, RL)又称为增强学习或再励学习,是与监督学习和无监督学习并列的一大类机器学习方法.作为一种以环境的状态和评价性反馈为输入的机器学习方法,强化学习通过与环境交互,不断改进策略,最终获得最优行为策略.而由多个并发的强化学习主体组成的多智能体系统近年来受到了越来越多的关注.Littman基于随机决策理论框架提出了零和策略下的多智能体强化学习算法[4],Hu和Wellman将这种方法扩展到非零和决策[5],这种算法可以看作是单个智能体Q-学习的扩展;L. Busoniu等人提出适应性的状态聚焦Q-学习算法[6],其特点在于状态空间由简变繁直至学习的收敛性满足要求,提高了学习效率;J. R. Kok等人将整体的行为值函数分解并利用决定性的传播算法,使得问题的规模仅为线性[7].

收稿日期:2010-03-25.
基金项目:国家自然科学基金资助项目(60774076,90820302).
通信作者:连传强. E-mail:wzdslcq@163.com.

多智能体强化学习在多机器人系统中的应用最为广泛,如多机器人足球赛[8]、多机器人编队控制[9]等,此外在其他领域如资源分配、游戏、网络路由选择、口语对话系统等也有了一定的应用,目前已成为机器学习领域研究的热点之一.

目前的多智能体强化学习算法基本还是以Q-学习算法为主,由于Q-学习算法是一种表格式的算法;因此在面临大规模空间的分布式优化决策问题时,往往存在学习效率低、计算收敛性差等缺点.因此如何改进多智能体强化学习算法就显得尤为重要.本文通过网络环境中的资源分配问题来研究多智能体强化学习算法.虽然一些学习算法已经被应用到资源分配问题,但这些学习算法的效率往往不能令人满意;因此通过资源分配问题来研究多智能体强化学习算法对优化资源分配问题、改进多智能体强化学习算法具有重要的意义.

本文研究的网络环境中的资源分配问题模型是一个由若干个共享簇组成的网络结构,每个簇拥有一定数量的计算节点,每个节点都拥有一定的资源.针对其特点,提出了Q-CF多智能体强化学习算法,它将资源分配问题的决策分解成2步决策:本地分配决策和任务传递决策,其中分布式Q-学习算法的学习任务是学习本地分配策略,而利用CF学习算法来学习任务的传递决策.仿真结果表明,和已有的多智能体Q-学习算法相比,该方法具有更加快速的收敛速度,同时保证了协同策略的性能优化.

1 资源分配问题描述与多智能体强化学习

1.1 资源分配问题描述

对于一个网络环境中的资源分配系统,每个簇可以从外部接收任务,也可以从邻居簇接收任务.在每个时间步,簇都必须做出决策任务在本地如何分配以及未分配的任务该传递给哪个邻居簇.每个任务都有自己的效用率,当任务在本地被成功地分配,那么系统将在每个时间步都获得相应的效用值直到任务完成;反之,如果一个任务在它的最大等待时间内没有被分配,那么这个任务将从系统中清除.当任务完成以后,它所占用的资源将被返还给系统.而研究这个问题的目的就是最大化整个系统的平均效用率.

用集合(C,A,T,D,R)来描述资源分配系统,其中:

$C=\{C_1,C_2,\cdots,C_n\}$是所有簇的集合.

$\boldsymbol{A}=[a_{ij}]$是任务在相邻簇之间的传递时间矩阵,其中a_{ij}表示任务在相邻簇C_i和C_j之间传递所花费的时间.

$T=\{t_1,t_2,\cdots,t_m\}$是所有任务类型的集合.

$D=\{d_{ij}\}$是任务类型到达的分布,其中d_{ij}表示任务类型t_i到达簇C_j的分布.

$R=\{R_1,R_2,\cdots,R_k\}$是每个簇所能提供资源类型的集合.

每个簇都有自己的一些计算节点,每个节点都拥有若干种资源类型以及相应的资源量.一般来说,一个任务可以被几个节点共同完成,但是为了简化问题的复杂度,这里假设一个任务只能被一个节点完成,因此对任务类型t_i可用集合$(D_i^{rd},D_i^s,D_i^u,D_i^w)$来描述,其中:$D_i^{rd}=\{(R_i,D_{1d}),\cdots,(R_p,D_{pd})\}$表示的是一个任务类型所需求的资源分布,其中$D_{pd}$表示的是对资源类型$R_p$的需求量的分布;$D_i^s$表示的是任务的服务时间分布;$D_i^u$表示的是任务的效用率分布;$D_i^w$表示的是任务最大等待时间的分布.

通过以上的定义,不难得出系统的平均效用率为

$$\mathrm{AUR}=\frac{\sum_{t=1}^{t}\sum^{n}{}_{t_i=1}\sum_{x\in T_{t_i}}u(x)}{T}.$$

式中:T为仿真总时间,T_{t_i}为t时刻在簇C_i上所有正在运行的任务集合,$u(x)$为任务x的效用率,而该文的目的就是通过学习使得系统的AUR最大化.

1.2 多智能体强化学习

学习的过程通常是一个马尔可夫决策过程(MDP),可用集合$(S,A,\boldsymbol{P},r)$来描述这一过程,其中:S为状态空间,A为行为空间,$\boldsymbol{P}$是状态转移矩阵,$\boldsymbol{P}(s_{t+1}|s_t,a_t)$表示的是在$t$时刻状态为$s_t$时执行动作$a_t$使得在$t+1$时刻状态为$S_{t+1}$的概率,$r$为回报函数,$r(s'|s,a)$表示为在状态$s$执行$a$动作使得状态转移到$s'$时所得到的立即回报值.而智能体的学习目标就是最大化如下的折算累计回报:

$$V=\sum_{i=0}^{\infty}\gamma^i r(s_{j+1}\mid s_j,a_j). \tag{1}$$

式中:$\gamma\in(0,1)$为折算因子.

作为一种强化学习算法,Q-学习算法中的评估函数$Q(s,a)$定义为从状态s执行动作a的立即回报加上以后遵循最优策略的值(用γ折算),其更新

规则为

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma\overline{Q} - Q(s,a)]. \tag{2}$$

式中：$\overline{Q} = \sum_{s'} p(s' \mid s,a) \sum_{a'} \pi(s',a') Q(s',a')$，$\pi(s',a')$ 为在 s' 状态下执行 a' 动作的概率. $\alpha \in (0,1)$ 为学习率，$\gamma \in (0,1)$ 为折算因子，r 为在 s 状态下执行 a 动作的立即回报.

在状态 s 上的最优动作 a 为

$$\pi^*(s) = \arg\max_a Q(s,a).$$

Hu 和 Wellman 将单智能体的 Q-学习算法推广到了多智能体系统[10]. 在他们的算法中，假设一个智能体可以感知其他智能体的行动和从环境中得到的回报，所有智能体的 Q 值表都被保存. 在假设有 2 个智能体的系统中，当智能体 i 在状态 s 选择动作 (a_1,a_2) 到达状态 s' 时，智能体得到回报 (r_1,r_2)，Q 值表通过下式进行更新：

$$Q_i(s,a_1,a_2) \leftarrow (1-\alpha) Q_i(s,a_1,a_2) + \alpha\{r_i + \gamma V_i(s')\}.$$

式中：$V_i(s')$ 是智能体 i 在双矩阵决策 $\langle A_1, A_2, Q_1(S), Q_2(S)\rangle$ 的平衡点.

2　Q-CF 多智能体强化学习算法

Q-CF 学习算法是一个异步的强化学习算法，即通过 Q-学习算法和 CF 学习算法相结合实现异步学习，其中利用 Q-学习算法来学习任务的本地分配策略，而通过 CF 学习算法来学习任务的传递对象选择策略.

2.1　本地策略的分布式 Q-学习算法

每个簇均独立地使用 Q-学习算法学习任务在本地的分配策略. 当一个簇接收到一定数量的任务时，如何根据簇的资源状态合理地分配这些任务使得平均效用率最大，这是学习的目的. 首先定义集合 $(S,A,\boldsymbol{p},r)$.

状态空间 S：为了便于对比，采用了文献[1]中的问题模型. 原问题模型有 2 种资源，即 cpu 资源和 network 资源，将每种资源按照数量的多少分成 n 个层次，再将每个节点各资源的层次联合起来，便成为描述这个节点的资源状态. n 越大，对节点状态的描述能力就越强，但同时状态空间也急剧变大，严重影响了 Q-学习的效果. 因此在这里每种资源只分为 2 个层次，设定一个阈值，比如 60，当资源低于这个阈值的时候，定义为 low，反之定义为 high，那么一个计算节点的资源状态可用 (low, low)、(low, high)、(high, low)、(high, high) 中的一种来表示. 因此，定义 $s = (n_1,n_2,n_3,n_4)$，其中 n_1 表示一个簇中节点状态为 (low,low) 的数量，n_2 表示 (low,high) 的数量，n_3 表示 (high,low) 的数量，n_4 表示 (high,high) 的数量.

动作空间 a：a = (action1, action2)，其中 action1 表示选取的任务类型，action2 表示为选取的任务分配到哪一种类型的节点. 例如 $a = (3,4)$，表示将第 3 种任务类型的任务分配到状态为 (high,high) 的节点. 如果任务中有多个相同的任务类型，将优先选取效用率高的任务.

状态转移矩阵 $\boldsymbol{p}$：通过实验可以实现对状态转移矩阵 $\boldsymbol{p}$ 的估计.

回报函数 r：如果相应的任务完成了，$r = u$，其中 u 为任务的效用率；反之，$r = 0$.

定义好 $(S,A,\boldsymbol{p},r)$ 之后，就可以利用 Q 值更新规则 (1) 以及搜索策略 (ε-greedy) 进行 Q-学习：

$$\pi(s,a_i) = \frac{e^{Q}(s,a_i).\tau}{\sum_j e^{Q}(s,a_j)/\tau}.$$

式中：e、τ 均为大于 0 的常量.

本地策略的分布式 Q-学习算法流程为

对簇 i 每个 s，a 初始化表项 $Q_i(s,a)$ 为 0，t 时刻簇 i 的状态为 s，等待本地分配的任务数为 n；

循环 (循环次数为 n)：

1) 利用 ε-greedy 方法选择 1 个动作 a 并执行它；

2) 接收到立即回报 r；

3) 观察新状态 s'；

4) 对 $Q_i(s,a)$ 按照式 (3) 进行更新；

5) $s \leftarrow s'$；

6) 若 $r = 0$，则将该任务放入等待传递队列，否则移除该任务；

结束.

2.2　智能体之间任务传递策略的 CF 学习算法

CF 学习算法学习的是本地未分配的任务该如何选择传递对象，以便让其他簇完成该任务. 它的学习算法思想来源于 K 臂赌博机问题[11]，对 $(S,A,\boldsymbol{p},r)$ 中的各项定义如下：

状态空间 s：状态只有 1 个，那就是 k 个簇.

动作空间 a：$a = (j,i)$，表示将任务类型为 t_j 的任务传递给编号为 i 的簇.

状态转移矩阵 $\boldsymbol{p}$:学习环境为单状态,状态转移矩阵 $\boldsymbol{p}(s|s)=1$.

回报函数 r:这里的回报函数有2个,第1个是立即回报 $r_1=-1$,表示的是将任务传递给簇的立即回报,或者说是惩罚,它存在的意义在后面将做出分析;第2个是延迟回报 $r_2=\beta^n$,其中 $\beta\in(0,1)$ 为折算率,n 为任务经过 n 次传递才被分配,$n=0$ 表示为任务在本地就被分配了,如果在任务的最大等待时间内没有被分配,则 $n=\infty$,即 $r_2=0$.

定义好了以上各项之后,再定义值函数 Q 的更新规则. 它的更新规则是异步的,其中当任务 j 到达簇 i 时,更新规则为

$$Q(a)\leftarrow Q(a)+r_1.$$

当任务 j 被成功分配以后,此任务经过的所有簇则按照更新规则:

$$Q(a)\leftarrow Q(a)+r_2$$

来更新 Q 值.

簇 i 对任务 j 的传递策略为

$$\pi^*=\arg\max_{\text{neighbor}}Q(j,\text{Neighbors}). \tag{3}$$

式中:j 为任务类型为 t_j 的任务,Neighbors 为第 i 个簇的所有邻居簇.

智能体之间任务传递策略的 CF 学习算法流程如下:

t 时刻,簇 i 有 n 个任务等待传递.

循环(循环次数为 n):

1)对任一任务 j 按策略(3)传递到邻居簇 i';

2)对簇 i' 进行此任务的 Q 值更新,更新规则为:$Q(a)\leftarrow Q(a)+r_1$;

3)若任务 j 在簇 i' 被成功分配,则对任务 j 之前经过的所有簇进行 Q 值更新,更新规则为:$Q(a)\leftarrow Q(a)+r_2$;反之,则将任务 j 放入簇 i' 的等待传递队列;

4)将任务 j 从簇 i 的等待传递队列中移除.

由图1所示,假如任务 task 按照策略(3)如图中实线方向进行传递,直到簇5才被成功分配,那么成功分配的信息将会按照虚线的方向反馈回给每个此任务经过的簇,此信息具体的表现形式为 β^i,也就是相应簇的延迟回报 r_2. 不难看出,只有当任务在经历了 n 次传递直到被分配以后,β^i 的值才会反馈回给每一个它经历过的簇,因此这个回报是有延迟的,也就是说经过一段时间才会对 Q 值有所影响. 引进 r_1 立即惩罚函数以后,在某个时刻 Q 值会随着传递给它的任务增多而立即减小,从而引起传递策略的改变将任务传递给别的邻居簇,这样就避免了任务的堆积,整体上平衡了任务的分布.

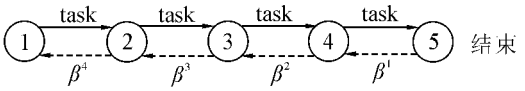


图1 信息链式反馈示意

Fig. 1 Information chain feedback sketch map

因为任务最后的分配情况会沿着它所经过的所有簇将信息逐级地反馈回去,因此称之为链式反馈学习算法. 从算法流程容易看出,此算法的特点在于值函数的更新是异步的,这对平衡任务分布,提高学习速度和效果起到了重要作用.

3 实验结果及分析

3.1 实验设计

为了便于比较,实验完全采用文献[1]中的实验模型,模型如图2所示,图中圆圈内的数字表示为簇的编号,相应的圆圈外的数字表示该簇所拥有的节点数,每个节点都有2种资源:cpu 和 network,它们的初始值范围均为[50,150],其中显示为带阴影的4个簇6、7、10和11可以接受外部输入的任务,所有的簇均可以接收内部簇之间传递的任务.

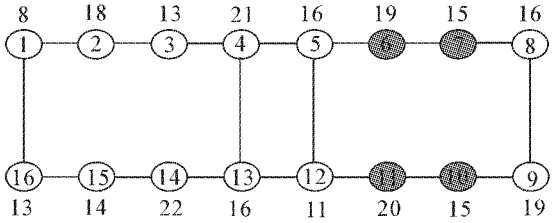


图2 实验模型

Fig. 2 Experiment model

假设所有任务的最大等待时间均为10,每种任务类型可用集合 (d_c,d_n,u,t_s) 表示,其中 d_c 表示为对 cpu 的需求量;d_n 表示对 network 的需求量;u 表示任务的效用率,t_s 表示任务的服务时间. 假设 d_c、d_n、u 服从泊松分布,t_s 服从指数分布,实验的任务类型有4种:

$$t_1=(9,8,1,20),\ t_2=(15,48,6,35),$$

$$t_3=(45,8,5,30),\ t_4=(47,43,25,50).$$

另外,外部任务的到达也服从参数如下的泊松分布:簇6:(4.5,0.5,2.5,1),簇7:(3.5,0.5,2,0.5),簇10:(4,2,0.5,1),簇11:(1.5,2.5,0.5,0.5).

最后,实验采用了3种方法进行效果的对比,具

体如表 1.

表 1　3 种决策方法

Table 1　Three decision-making methods

决策方法	G-R	Q-R	Q-CF
第 1 步决策算法	贪婪	Q	Q
第 2 步决策算法	随机	随机	CF

3.2　实验结果及分析

与文献[1]中的实验结果进行了比较，文献[1]中采用的是 FAL 多智能体强化学习算法，其本质上也是多智能体 Q-学习算法的一种，而本文采用的为 Q-CF 多智能体强化学习算法.

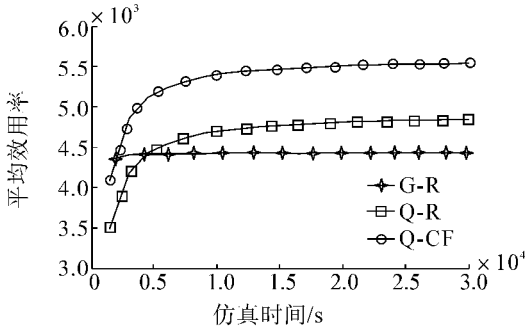


图 3　平均效用率

Fig. 3　Average utility rate

从图 3 中不难看出，Q-CF 曲线在 t = 25 000 s 左右趋于平稳，而文献[1]的方法在 t = 60 000 s 左右才趋于平稳，2 种方法的学习效果几乎相当. 因此本文的 Q-CF 算法在保证了学习效果的同时，具有更快的收敛速度，下面对其原因进行分析.

对分布式 Q-学习算法的分析：文献[1]在学习本地分配策略时也用到了 Q-学习算法，但因为其在对状态的定义中还包含了对任务状态的定义（其实只是非常粗略的定义），这使得状态空间为本文的 16 倍，因此影响了其学习速度. 而本文在 Q-学习中对任务的到达情况进行了统计，将这种统计用到学习中，在一定程度上弥补了状态定义的精简，因此学习的效果没有太大变化.

对 CF 学习算法分析：因为引入了立即惩罚函数 r_j，使得 Q 值的更新能够更加准确地反映实际情况，而延迟回报 r_2 至多在任务的最大等待时间内就能够反馈给相应的 Q 值，因此 CF 学习算法的时效性非常强，大大提高了学习速度；而文献[1]中的 Q-学习因为状态空间特别大，而且需要不断地统计邻居簇在各种状态下各任务的完成情况，这需要一定的仿真时间才能保证其准确性，学习速度因此变慢.

图 4 对 Q-R 与 Q-CF 各簇的平均效用率进行了比较，容易看出中间部分 Q-R 各簇的平均效用率略高于 Q-CF，这是因为 Q-R 第 2 步分配采取了随机策略，任务很难传递得很远，因此几乎都堆积在这些簇中而使得这些簇的值较高；而在两边的那些簇中，Q-CF 要远远好于 Q-R，这是因为 Q-R 第 2 步决策采取随机策略使得任务的传递很难达到远离能够接受外部任务的那些簇，而 Q-CF 则因为第 2 步决策采用 CF 学习算法而能够将任务量在系统中很好的平衡，因此各簇平均效用率的值分布得比较好.

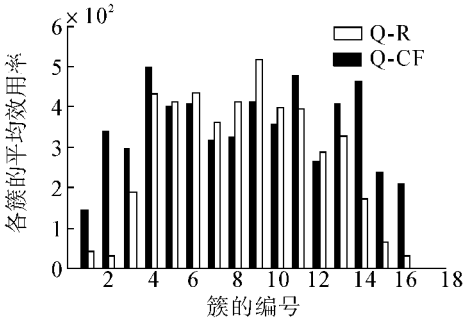


图 4　Q-R 与 Q-CF 各簇的平均效用率

Fig. 4　Average utility rate of clusters of Q-R and Q-CF

4　结束语

本文通过网络环境中的资源分配问题对多智能体强化学习算法进行了深入的研究，提出了 Q-CF 学习算法，实现了智能体之间的高效协同，达到了学习的目的. 实验结果表明，本文方法和已有的多智能体 Q-学习算法相比，具有更加快速的收敛速度，同时保证了协同策略的性能优化. 这对以后相关问题的研究以及多智能体强化学习算法在分布式系统中的应用具有一定的意义.

由于 Q-学习算法是一种表格式的算法，因此当状态动作空间较大时学习速度和效果都不能令人满意，而本文也是将状态空间进行了离散化之后才将 Q-学习算法用于其中，这样势必会影响学习的效果. 而更高级的强化学习算法，如最小二乘策略迭代（LSPI）[12]、基于核的最小二乘策略迭代（KLSPI）[13]等均采用逼近的方式，解决了状态动作空间连续或者巨大的问题，因此用更高效的算法来替代 Q-学习算法应用到多智能体系统中将是以后要继续的工作.