第35卷 第14期 计 算 机 工 程 2009年7月
Vol.35 No.14 Computer Engineering July 2009
·人工智能及识别技术· 文章编号：1000—3428(2009)14—0230—03 文献标识码：A 中图分类号：TP301

基于DFS的多Agent动态任务分配算法

陈 凤，先晓兵
(常熟理工学院信息化办公室，常熟 215500)

摘 要：针对任务分配算法应用于不确定动态环境时存在的不足，研究具有动态模糊特性的任务环境，借助动态模糊集理论，给出相关的多Agent动态任务分配算法并进行实例测试。测试结果表明，该算法模型可以合理地模拟MAS系统中任务分配的运行过程，并获得最优的任务分配策略与良好的任务实现效果。

关键词：多Agent系统；动态任务分配；动态模糊集

Multi-Agent Dynamic Task Allocation Algorithm Based on Dynamic Fuzzy Set

CHEN Feng , XIAN Xiao-bing
(Information Office, Changshu Institute of Technology, Changshu 215500)

【Abstract】 There are some problems when task allocation algoritm is dynamic and uncertainty environments. This paper describes the task allocation environment with Dynamic Fuzzy Set(DFS) and presents a dynamic task allocation method that is used to improve veracity and reduce the error. It is proved that this model can simulate the process of task allocation to get the optimal task allocation strategy. Example test results indicate the rationality and validity of the algorithm.

【Key words】 Multi-Agent System(MAS); dynamic task allocation; Dynamic Fuzzy Set(DFS)

1 概述

多Agent系统(Multi-Agent System, MAS)是近年来人们普遍关心的问题之一。过去的任务分配主要是根据任务固有属性的不同作简单划分，以达到任务分配的目的。在这个过程中假设任务是不可分割的最小单位，用单个Agent独立地实现各个任务。Smith分析了合同网(CNP)在Agent中如何满足任务要求的作用，其主要思想是将任务分成多个子任务和子合同，然后通过相应的竞拍机制分配给每个Agent。合同网将任务分配给单个Agent完成，具有一个合理有效的任务划分算法是该方法的关键。Gasser主要讨论了多Agent系统如何应用Agent知识和行为的社会性，很多分布式人工智能研究学者用博弈论解决这个问题，目的是设计一个合理的协调分配算法。Sandholm和Lesser给出了一个有限制的、理性的Agent协调分配模型，并提出了一般的分类算法。该算法主要依赖计算时间，其不足是所有Agent分配和组合都是在假定计算一个稳定任务的基础上进行的。文献[1]提出了一个在分布合作问题上求解环境中任务分配的算法，该算法放弃了限制Agent合作数量的易操作特性，选用贪心启发方式进行Agent组队，其原因是：在给定的最有可能的解决方案中，对解决问题的Agent数目的限制是可以被计算证明的。

任务分配问题属于非确定性多项式时间问题，当算法可以观测所有组队Agent个体的值而并非所有Agent集体的值时，任务分配的复杂性就降低了。文献[2]在这个基础上提出了动态规划算法，该算法的时间复杂度为$O(3^n)$，明显优于无限穷举算法的时间复杂度$O(n^n)$，然而，该时间复杂度是有限制条件的，在实际的任务分配中不可行，因为实际任务分配中的任务都是通过Agent组队完成的。文献[3]提出用集合划分和集合覆盖的理论解决任务分配问题的方法，其中的贪心近似算法能得到一个好的最有效任务分配。若可行的子集空间有限，启发式算法可以很好地解决该问题。为了实现多Agent任务分配，集合划分、覆盖理论要求应用的可能组合的空间是自然有限环境，且任务是通过异构Agent实现的。拍卖技术是多Agent系统进行任务分配常用的，它的主要原理是随机生成竞拍顺序，各个Agent按照顺序竞拍到自己要执行的任务，一轮完成后得到各自的任务分配方案。文献[4]用线性规划处理该问题，将任务分配映射到0-1整数规划上，在最坏情况下输入是指数级的，在具体问题中输入一般都是多项式级的，且运行效果良好。其不足是单一的方法显得过于集中，在解决动态的任务分配中，逼近矩阵则显得力不从心。文献[5]通过蚁群的适应机制构造动态任务分配问题，引导系统对能量进行最优配置，但在该过程中单个个体的感知能力和社会交互能力都受到了限制，且不能确定自适应机制的蚁群方法与能量最优化配置之间关联的密切程度，得到的最优值也不能得到任何理论模型的证明。与此同时，为使任务分配达到能量配置最优，蚁群在自动调节适应动态的环境和进化策略上还需进一步加强。文献[6]针对分布式计算中多值任务分配问题提出了A*算法和遗传算法，引入多实例机器学习的概念处理并行问题，算法为分布式控制系统提供了良好的可测性并减少了系统运行时间。

综上所述，面对动态模糊环境，目前的动态任务分配算法都存在一些不足。因此，本文选择动态模糊集(Dynamic

作者简介：陈 凤(1980－)，女，硕士，主研方向：机器学习，动态模糊逻辑；先晓兵，讲师、硕士

收稿日期：2008-12-26 **E-mail：**hhitcf@cslg.edu.cn

Fuzzy Set, DFS)和 Agent 技术共同解决这个问题。

2 基于 DFS 的多 Agent 动态任务分配的基本概念

在 MAS 中，根据多 Agent 协调工作的原则，动态任务分配可以看作：某个 Agent 在一个关键时间点上选择一个合适的行为，通过团队合作完成全局任务。如何将多个 Agent 进行合理组合，调整各自的行为，最大程度地实现系统和各子系统的目标是本文要解决的一个重要的问题。

从整个任务的角度看，多 Agent 协调的行为选择可定义为被组合 Agent 状态空间到被组合 Agent 行为空间的映射。

定义 1 动态模糊 Agent 行为选择 $(\bar{C},\vec{C})$ 是指在被组合 Agent 任务分配环境状态空间 $(\bar{S},\vec{S})$ 中的一个输入数据集 $(\bar{I},\vec{I})$ 到一个输出数据集 $(\bar{O},\vec{O})$ 的映射，可表示为 $(\bar{C},\vec{C})$ ： $(\bar{I},\vec{I})\rightarrow(\bar{O},\vec{O})$ 。

如图 1 所示，映射过程分成下面 3 个步骤：

(1)每个 Agent 根据自身感知适应性来竞争任务的执行。

(2)通过相应的拍卖决策机制决定 Agent 的任务实现者。

(3)竞胜者 Agent 执行一个或多个动作完成任务。

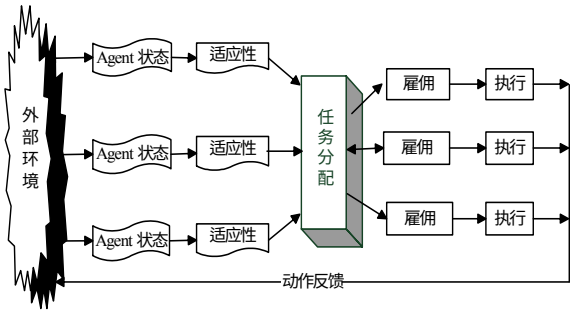

图 1 动态任务分配过程

定义 2 动态模糊任务分配过程模型：$\{(\bar{S},\vec{S}),(\bar{L},\vec{L}),(\bar{u},\vec{u}),(\bar{y},\vec{y}),(\bar{L},\vec{L}),(\bar{I},\vec{I}),(\bar{O},\vec{O})\}$，其中，$(\bar{S},\vec{S})$ 是所有 Agent 所要完成的任务集合；$(\bar{L},\vec{L})$ 是已动态分配给 Agent 执行的任务；$(\bar{u},\vec{u})$ 是 $(\bar{S},\vec{S})$ 到 $(\bar{L},\vec{L})$ 的输出；$(\bar{y},\vec{y})$ 是 $(\bar{L},\vec{L})$ 到 $(\bar{S},\vec{S})$ 的动态反馈；$(\bar{p},\vec{p})$ 是系统分配的性能指标；$(\bar{I},\vec{I})$ 是外界环境对动态模糊多 Agent 动态分配系统的输入；$(\bar{O},\vec{O})$ 是系统对外界的输出。

定义 3 动态模糊任务分配过程模型描述为

$$(\bar{x},\vec{x})(k+1)=G_1((\bar{x},\vec{x})(k),(\bar{u},\vec{u})(k),\xi_1(k)) \tag{1}$$

$$(\bar{y},\vec{y})(k)=G_2((\bar{x},\vec{x})(k),(\bar{u},\vec{u})(k),\xi_2(k)) \tag{2}$$

$$(\bar{p},\vec{p})(k)=\sum_{i=1}^{k}P((\bar{y},\vec{y})(i)) \tag{3}$$

其中，$(\bar{x},\vec{x})(k)$ 是 $(\bar{S},\vec{S})$ 在时刻 k 的状态变量；$(\bar{u},\vec{u})(k)$ 是 $(\bar{S},\vec{S})$ 的动态输出；$\xi_1(k)$ 是状态方程中的随机干扰；$(\bar{y},\vec{y})(k)$ 是 $(\bar{L},\vec{L})$ 的动态反馈输出；$\xi_2(k)$ 是观察随机误差；k 表示时刻，只取整数值。假定其中的向量全部为有限维状态变量。$(\bar{P},\vec{P})$ 表示系统执行性能指标，$P(i)$是一个标量函数，表示时刻 i 时系统执行的性能。

3 基于 DFS 的多 Agent 动态任务分配分析

3.1 多 Agent 动态任务环境

根据环境要求的不同将任务分成 3 种类型：(1)紧急任务：允许剥夺正在执行的标准任务，一般是通过给每个 Agent 高于一般的奖赏值，这种任务要求立即响应。(2)标准任务：这类任务只有在 Agent 资源十分丰富而是任务的效用足够时有利于完成分配。(3)不被剥夺的任务：类似于标准任务，但其执行是不能打断的。

根据任务的特点，采取 2 种分配方式：(1)及时分配：假定环境中有一系列的任务，每个任务的分配方法必须是最优的。在这种分配方式下，要经过多重的任务拍卖过程，目的是要求资源能够获得最优的使用状态，减少 Agent 的重复冗余使用。(2)随机分配：假定环境中有一些紧急任务要完成，这些任务随机分配给各个 Agent 只需获得高的奖赏值，对于任务分配是否是资源最优条件并不很苛刻。

3.2 多 Agent 动态任务分配过程

多 Agent 动态任务分配过程是一个受激励机制控制的有序过程，其一般流程如下：

(1)对需要执行的目标任务 $(\bar{So},\vec{So})$ 中的某一个相关子集 $(\bar{So}^{+},\vec{So}^{+})$，激励机制 $(\bar{G},\vec{G})$ 依据 $(\bar{So}^{+},\vec{So}^{+})$ 的公共特性激活一个操作机制 $(\bar{Op},\vec{Op})$ 的子集 $(\bar{Op}^{+},\vec{Op}^{+})$，在 $(\bar{Op}^{+},\vec{Op}^{+})$ 的作用下，形成一个任务目标 $(\bar{Y},\vec{Y})$ 的子集 $(\bar{Y}^{+},\vec{Y}^{+})$ 。

(2)对于任务目标 $(\bar{Y},\vec{Y})$ 中的元素 $(\bar{y}_0,\vec{y}_0)$，在执行算法的作用下有一个偏差信息 $E(\bar{y}_0,\vec{y}_0)$，激励机制 $(\bar{G},\vec{G})$ 依据激活评价机制 $(\bar{V},\vec{V})$ 的一个子集 $(\bar{V}^{+},\vec{V}^{+})$，并作用于任务环境，环境响应 $N(\bar{y}_0,\vec{y}_0)$ 与偏差信息 $E(\bar{y}_0,\vec{y}_0)$ 在 $(\bar{V}^{+},\vec{V}^{+})$ 的作用下，对 $(\bar{y}_0,\vec{y}_0)$ 进行修正。

(3)对于需要执行的任务 $(\bar{So},\vec{So})$ 中的某个子集 $(\bar{So}^{+},\vec{So}^{+})$，其操作与评价的结果为：$(\bar{V},\vec{V})\,[\,(\bar{Op},\vec{Op})\ (\bar{So},\vec{So})\,]\Rightarrow(\bar{Y}^{+},\vec{Y}^{+})\subseteq(\bar{Y},\vec{Y})$。

(4)重复上述过程直到任务执行过程达到精度要求。

3.3 多 Agent 动态任务分配算法的实现

3.3.1 多 Agent 动态任务分配算法描述

多 Agent 动态任务分配执行有下面 4 个主要成分：要分配的执行的任务状态空间，任务分配的执行的 Agent 团队，每个 Agent 所产生的动作集，任务分配的系统响应。

多 Agent 动态任务分配算法执行步骤如下：

(1)在执行开始阶段，每个 Agent 根据自己的感知信息构造自身的优先级，从而确定 Agent 的执行顺序。

(2)团队中的 Agent $k\in N$，$R^{k}(s)=0$，在状态 $s\in S$ 时，平均奖赏值表示为 $\rho^{k}=0$，初始化输入参数 (α_m,β_m)，探测系数为 γ_m。

(3)假设现在系统所处的周期是 m，那么确定下一个周期 m+1 的情况。对于每个 Agent 团队 $k\in B$，选择 $a^{k}\in A^{k}(s)$ 为 $1-\gamma_m$ 的概率，当 $R^{k}(s,a^{k})$ 为最大值；选择其他随机的动作 $a^{k}\in A^{k}(s)-a^{k}$ 为 γ_m 的概率。

(4)计算 Agent 团队中的所有 Agent 的奖赏值 $r^{k}(s,s',a^{k})$，第 k 个 Agent 团队采取动作 $(a^{1},a^{2},\cdots,a^{N})$。

(5)根据式(4)计算 Agent 团队中所有 Agent 的平均奖赏值。

$$\rho_{m+1}^{k}=(1-\beta_m)\rho_m^{k}+\beta_m\times(m\rho_m^{k}+r^{k})\Big/_{m+1} \tag{4}$$

$$R_{new}^{k}(s,a^{k})\leftarrow(1-\alpha_m)R_{old}^{k}(s,a^{k})+\sum_{a^{k}\in A^{k}(s)}P^{k}(s',a^{k}) \tag{5}$$

更新 $R^{k}(s,a^{k})$，当 $p^{k}(s',a^{k})=1-\gamma_m$， a^{k} 使用贪心策略，$p^{k}(s',a^{k})=\gamma_m/A^{k}(s')-1$， a^{k} 为其他的策略。

(6)重复上述步骤直至整个任务完成。

(7)在执行算法的作用下，获得一个任务目标输出结果 $(\bar{Y},\vec{Y})$ 中的元素 $(\bar{y}_0,\vec{y}_0)$，与任务目标的要求存在偏差信息 $E(\bar{y}_0,\vec{y}_0)$，系统根据环境响应 $N(\bar{y}_0,\vec{y}_0)$ 与偏差信息 $E(\bar{y}_0,\vec{y}_0)$ 通过评价机制 $(\bar{V}^{+},\vec{V}^{+})$ 对 $(\bar{y}_0,\vec{y}_0)$ 进行修正。

(8)重复步骤(7)，直到任务执行过程偏差信息 $E(\bar{y}_0,\vec{y}_0)$ 达

到精度要求为止。

3.3.2　多 Agent 动态任务分配算法分析

在算法步骤(1)中需要确定 Agent 的执行次序，因为在任务执行的初期每个 Agent 选择动作是随机的。在 Agent 团队中多个 Agent 同时执行目标任务的情况下，任务的执行过程将变得不稳定。首先让每个 Agent 轮流执行，Agent 根据任务的适应度确定自身优先级，然后对优先级计算排序，选择优先级最大的 Agent。在一个周期内，由被选择的 Agent 执行目标任务，同时保持其他 Agent 按照上个周期获得的策略执行一个已有的动作，然后下一个周期按照优先级的次序，后面的 Agent 用同样的方法执行。

根据 Markov 过程无后效性的特点，每个 Agent 动作 $(\overline{a_i},\overline{a_i})$ 的平均奖赏值与动作 $(\overline{a_i},\overline{a_i})$ 相关的短期奖赏值有关，在此基础上，在算法步骤(5)中，原先的评价机制短期奖赏函数 $Rr:(\overline{S_0},\overline{S_0})\times(\overline{A},\overline{A})\to(\overline{V},\overline{V})$ 用平均奖赏函数 $Ar:(\overline{S_0},\overline{S_0})\times(\overline{A},\overline{A})\to(\overline{V},\overline{V})$ 替代，则 $\left|(\overline{G},\overline{G})(\overline{So},\overline{So})\right|=\left|(\overline{G},\overline{G})\right|\times M\times\left|(\overline{A},\overline{A})\right|$，由此可见，用于 Agent 团队执行的目标任务特征空间和动作集合之间的复杂度 $O(n\times m\times a)=O(k\times a)$ 明显优于原来的几何级数。

4　实例仿真

下面通过 2 个实例验证本文所提理论的可行性。

实例 1

用动态模糊任务分配过程模型处理 Hermit 多项式逼近的简单问题，从而验证该模型在处理一般问题上也不失一般性。图 2 示出了整个任务执行的过程效果，图中“＋”号为目标任务中一些离散的样本点，即随机给定的离散样本点。

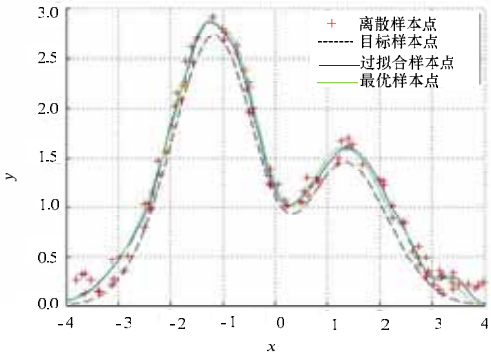

图 2　任务执行过程效果

本文在给定这些点时充分考虑到实际问题中出现的观测误差和随机干扰噪声，即上文提到的 $\xi_1(k)$ 是状态方程中的随机干扰，$\xi_2(k)$ 是观察随机误差，因此，这些样本点取值与实际点取值存在误差。在任务执行的过程中，若当前测试误差不再下降，则测试误差达到最小点，停止任务的执行。为了保证算法更可靠地停止在测试的最优点，以连续多个测试误差的均值不再下降作为停止条件。在执行过程中，算法在运行了 334 次迭代后测试误差开始上升，此时的测试误差为 2.982 9，得到最好的执行效果。若继续运行直至达到最大的迭代值，执行效果便出现过拟合现象。

实例 2

假定系统中有 4 类异构的 Agent 团队，它们要处理 3 类不同的任务，分别在标准数据集 Iris data set 中能根据提供的属性找到各自样本。该数据集共有 150 个样本点，每个样本点有 4 个属性：萼片长度，萼片宽度，花瓣长度，花瓣宽度，把花瓣长度和花瓣宽度作为异构 Agent 团体的初始训练集的属性，萼片长度和萼片宽度作为测试集的属性。分类效果如图 3 所示。

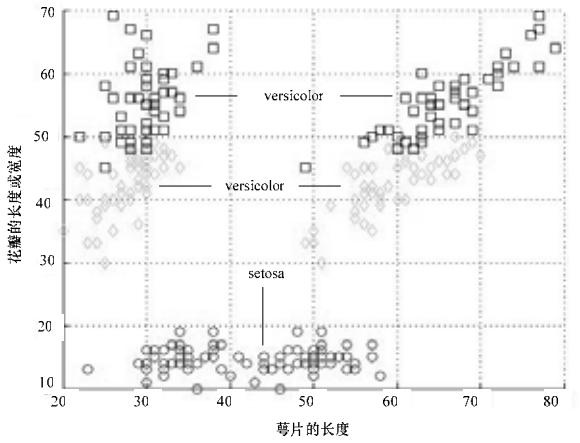

图 3　Iris 的数据集分布情况

在任务执行过程中统计的 4 个异构 Agent 团队协调合作的总次数为 3 000 次，在任务样本训练过程中，假定有 l 个样本，每组 Agent 每次取一个样本进行测试，其余的 $l-1$ 个样本用于训练，共有 l 种取法，对于每一轮次都按 l 种取法训练 l 次，当测试数据总和最小时停止训练。选出这 4 个 Agent 团队中执行时间最短的 Agent 团队的实验数据，如图 4 所示。随着训练时间的增加，测试误差和训练误差呈现单调递减，可根据任务所需的精度要求停止训练。

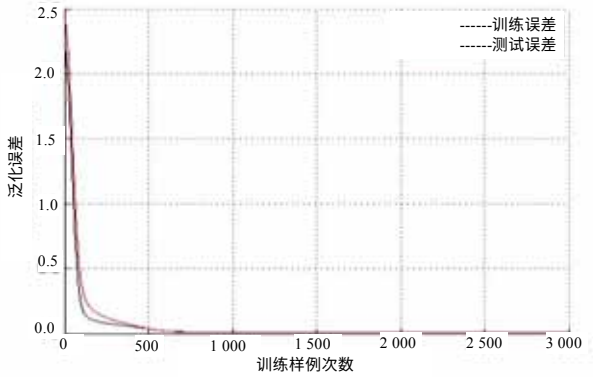

图 4　测试误差和训练误差的变化

5　结束语

本文针对目前多 Agent 的任务分配算法在处理动态模糊环境上的不足，运用动态模糊集解决多 Agent 的任务分配问题，根据任务出现的环境状况对任务进行分类处理，主要的分配方式有及时分配和随机分配。根据任务执行的动态性，认为多 Agent 动态任务分配动态过程是一个受激励机制控制的有序过程，并给出了相应的流程。根据任务出现的动态性，提出了基于动态模糊集的多 Agent 动态任务分配模型。深入分析了 Agent 团队所处的动态模糊环境，比较了多种动态任务分配算法的优缺点，提出了基于动态模糊集的动态任务分配算法。通过对目标任务空间的有效划分以及对评价函数的修改，降低了目标任务的搜索复杂度，提高了任务分配的效率。

本文中的任务分配都是在集中式管理的情况下进行的，下一步工作是如何让这些 Agent 团队在分布式情况下合理地协调，进一步提高 Agent 团队自身的适应管理能力。

(下转第 235 页)