基于混合遗传蚁群算法的多 Agent 动态任务分配研究

张 晋 曹耀钦
(第二炮兵工程学院 西安 710025)

摘 要 在多Agent系统中,由于任务的复杂性和Agent之间的异构,Agent的动态任务分配问题实际上是一个NP难优化问题。针对MAS的任务分配问题的动态特性,首先建立任务分配数学模型,建立任务分配优化的目标函数;其次提出了一种混合遗传蚁群算法。利用遗传算法快速迭代和蚁群算法正反馈信息、分布式求解的特点实现任务分配的组合优化。实验仿真的结果分析表明,该算法具备较好的全局收敛效率和求解精度,可明显提升多Agent系统的性能。

关键词 多Agent系统(MAS),动态任务分配,混合遗传蚁群算法
中图法分类号 TP181 **文献标识码** A

Research on Dynamic Task Allocation for MAS Based on Hybrid Genetic and Ant Colony Algorithm

ZHANG Jin CAO Yao-qin
(The Second Artillery Engineering University, Xi'an 710025, China)

Abstract In multi-agent systems(MAS), because of the complexity and the difference between respective agents, the dynamic task allocation for multi-agent systems is a NP-hard combinatorial optimization problem. According to the dynamic characteristic of task allocation, first this paper established the mathematical model of task allocation and the target function. And then the hybrid genetic and ant colony algorithm which possesses the trait such as rapid iteration, positive reaction and distribution, was put forward to achieve combinatorial optimization of task allocation. Finally, the simulation experiments demonstrated that, the algorithm is accomplished in convergence efficiency and solution precision can significantly enhance the performance of MAS.

Keywords MAS, Dynamic task allocation, Hybrid genetic and ant colony algorithm

在分布式计算环境下,多 Agent 任务分配问题关乎整个系统的性能,已有大量的研究机构和个人致力于多 Agent 任务分配策略问题。文献[1]提出了混合蚁群算法对多 Agent 任务分配进行了组合优化。文献[2-4]将蚁群算法和遗传算法进行动态融合,在组合求解精度和时间上有较好的效果。文献[5]以 TSP 问题为例,提出混合遗传模拟退火算法,改善了算法早熟问题。然而这些研究工作只考虑了静态任务环境,应用在动态任务环境下已没有明显的优势。

在复杂环境下,动态性是多 Agent 系统(MAS)的显著特征。其动态性体现在:Agent 节点的加入或退出、应用环境急剧变化、任务流之间的交叉影响、通信带宽限制和任务优先级差别等。MAS 的动态任务分配问题即在极短的时间内有效地获得最优 Agent 组合来执行任务。

MAS 系统中的任务分配问题是一个典型的 NP-难优化问题。蚁群算法和遗传算法都是基于群体的仿生算法,所使用的算子来源于种群演化而对解空间进行搜索。本文针对蚁群算法或者遗传算法都存在过早或过晚、容易陷入局部最优解的问题,对遗传算法和蚁群算法进行动态融合以提高求解精度和速度,解决 Agent 之间的动态任务分配问题。

1 问题描述

任务分配问题可定义为:任务集 $I=\{1,2,\cdots,m\}$ 和 Agent 集 $J=\{1,2,\cdots,n\}$,任务集中的任务 $i(i\in I)$ 分配给一组 $\text{Agent}_j(j\in J)$ 来完成。每一个 Agent_j 的资源能力为 a_j,而分配给 Agent_j 的每一项任务 i 都会消耗其 r_{ij} 的能力资源。系统把任务 i 分配给 Agent_j 的代价为 c_{ij}。

$$c_{ij}=\omega_1 t_{ij}+\omega_2\sum_{k=1}^{s}\theta_{ij} \tag{1}$$

式中,t_{ij} 为时间开销,θ_{ij} 为硬件资源开销、通信资源开销和其他资源开销,s 为资源数,ω_1、ω_2 为对应权重且 $\omega_1+\omega_2=1$。任务分配问题求解的目标是以最低的资源代价找出可行的任务分配方案。建立目标函数如下,其中 x_{ij} 为 0-1 决策变量。当任务 i 被分配给 Agent_j 时 x_{ij} 的值为 1,否则为 0。

$$\min F(x)=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij} \tag{2}$$

满足约束条件:

$$\sum_{i=1}^{m}r_{ij}x_{ij}\leqslant a_j,(j=1,\cdots,n) \tag{3}$$

$$\sum_{j=1}^{n}x_{ij}=1,(i=1,\cdots,m) \tag{4}$$

$$x_{ij}\in\{0,1\} \tag{5}$$

张 晋(1985—),男,硕士生,主要研究方向为计算机网络及应用,E-mail:jimyeah4016@yahoo.com.cn;曹耀钦(1963—),男,博士,教授,主要研究方向为计算机网络及应用。

式(3)表明 Agent 的资源能力有限,式(4)、式(5)表明一个任务只能分配给一个 Agent,同一个任务不能分割给多个 Agent,式(5)中 $i=(1,2,\cdots,m)$,$j=(1,2,\cdots,n)$。

2 混合遗传蚁群算法

2.1 相关算法简介

遗传算法[6](Genetic Algorithm,GA)由美国 Michigan 大学 J. Holland 教授于 1975 年提出,是近年来迅速发展起来的一种全新的随机搜索与优化算法,其基本思想是基于 Darwin 生物进化论和 Mendel 遗传学说。蚁群算法[7](Ant Colony Optimization,ACO)由意大利学者 Marco Dorigo 在意大利米兰理工大学于 1991 年首次提出,是一种模拟蚂蚁觅食过程的仿生算法。

GA 和 ACO 都具备全局搜索、概率随机搜索、自适应好、鲁棒性和易于与其他搜索算法结合的优点。但是,GA 和 ACO 都可能陷入局部最优。另外,GA 搜索快速,ACO 可看作分布式算法且具备正反馈机制。GA 的主要缺点是算法后期易做大量冗余迭代导致求解精度和效率低。ACO 的主要缺点是初期信息素缺乏,导致搜索初期积累信息素占用的时间较长。

2.2 算法基本思想

本文将遗传算法和蚁群算法的优势进行融合。混合遗传蚁群算法能够弥补遗传算法的信息反馈以及提高求解精度,而遗传算法又能适当弥补蚁群算法的求解速度慢的问题。由于蚁群算法在缺乏初始信息素分布时,求解速度慢。而遗传算法能快速收敛,因此,在算法的前一阶段,利用遗传算法的随机性和快速迭代特性,生成任务分配的次优解作为蚁群算法的初始信息素分布,再利用蚁群算法的全局收敛能力、并行正反馈对问题的解进行优化。

2.2.1 基于遗传算法的次优任务分配

(1)生成初始种群:设置系统解集 $S=\{s_1,s_2,\cdots,s_n\}$。在某系统时段内 MAS 的 Agent 先验知识库为空的情况下,随机生成初始种群;若不为空,则根据先验知识即系统前次搜索出的解集 S 为初始种群。

(2)编码:采用二进制编码。Agent 的忙碌空闲情况决定了编码只需取 0 或者 1。编码 0 或 1 对应 Agent 集 J 中的 $Agent_j$ 是否在解空间中。

(3)适应度函数即为本文的目标函数式(1)。算法运行初期,个体差异明显,优势个体也明显,适应度高,易造成局部收敛。算法运行后期,遗传算子变化少,个体之间差异小,算法效率和求解精度下降。因此,将模拟退火算子 Metropolis 分布加入遗传算法:

$$P_{accept}(s,s',T)=\begin{cases}1, & \text{if } f(s')<f(s)\\ \exp\left(\dfrac{f(s)-f(s')}{T}\right), & \text{else}\end{cases} \tag{6}$$

温度更新函数如下:

$$T(t)=\frac{T_0}{1+\mu t} \tag{7}$$

算法开始时,温度下降较快,而在算法运行后期,温度下降的速率减小,所以可以改善算法后期的求解精度。设置一个较小温度值 ε,当 $T\leqslant\varepsilon$ 时遗传算法迭代结束。

(4)选择:采用轮盘赌。

(5)交叉:采用 Syswerda 提出的基于基因位置的杂交。遗传算子是搜索的最大驱动力,在算法运行初期,为了增强全局搜索能力,让遗传算子尽量大。

(6)变异:采用交换变异。随机选择两个位置并交换其内容。

2.2.2 基于蚁群算法的优化任务分配

(1)路径构建

在遗传算法求得次优解 $X=\{x_1,x_2,\cdots,x_n\}$ 以后,把次优解作为蚁群算法的初始信息素分布即蚂蚁的初始位置。其中 x_i 代表 $Agent_i$。在次优解空间中,蚁群算法将在次优解中进一步寻优。根据任务集 $I=\{1,2,\cdots,m\}$ 和 Agent 集 $J=\{1,2,\cdots,n\}$,将蚂蚁数量为 m 在 n 个 Agent 中寻找最优的 Agent 组合。定义一个资源代价矩阵 $C_{n\times n}$,$Agent_i$ 和 $Agent_j$ 之间的路径为 (i,j),蚂蚁 k 访问下一个 $Agent_j$ 的概率如下:

$$P_{ij}^k=\frac{[\tau_{ij}]^\alpha[\beta_{ij}]^\beta}{\sum_{l\in N_i^k}[\tau_{il}]^\alpha[\beta_{il}]^\beta},\ \text{if } j\in N_i^k \tag{8}$$

式中,$\eta_{ij}=1/c_{ij}$ 是一个预先设定的启发式信息,代表 $Agent_i$ 和 $Agent_j$ 之间的资源代价信息,τ_{ij} 为路径 (i,j) 上的信息素,α 和 β 决定信息素和启发式信息对路径选择的影响力。定义一个记忆存储结构 M_k,按照路径构建的先后顺序记录蚂蚁 k 已经访问过的 Agent 序号,并由此得出与 $Agent_j$ 相邻的 Agent 集合 N_i^k。

随着算法的运行,信息素浓度在已构建的路径上不断被强化,正反馈作用愈加明显。然而已构建的路径不一定最优,于是路径构建在式(8)的基础上采用伪随机比例作如下调整:

$$j=\begin{cases}\arg\max\{\tau_{il}[\eta_{il}]^\beta\}, & \text{if } q\leqslant q_0\\ J, & \text{else}\end{cases} \tag{9}$$

式中,q 是均匀分布在区间[0,1]中的一个随机变量,q_0($0\leqslant q_0\leqslant 1$)是一个参数,$J$ 是根据式(8)给出的概率分布产生的一个随机变量。调整的目的为:根据信息素的积累和启发式信息得出了 q_0,当蚂蚁 k 以 q_0 的概率访问下一 Agent 节点的同时,蚂蚁以($1-q_0$)的概率探索新路径。

(2)信息素更新

定义算法从开始到目前为止找到的最优路径实施额外的信息素强化,记为 T_{ts}。信息素强化额度如下(其中 C_{ts} 为目前最优路径的资源代价,由代价矩阵 $C_{n\times n}$ 和 T_{ts} 得来):

$$\Delta\tau_{ij}^{ts}=\begin{cases}1/C_{ts}, & \text{if } path(i,j) \text{ belong to } T_{ts}\\ 0, & \text{else}\end{cases} \tag{10}$$

在此,本文规定信息素的挥发和释放都只在构成 T_{ts} 的路径上进行。定义全局信息素更新规则如下:

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}+\rho\Delta\tau_{ij}^{ts},\ \forall(i,j)\in T^{ts} \tag{11}$$

除了全局信息素更新规则外,此处再采用局部信息素更新规则。在路径构建过程中,蚂蚁 k 每经过一条路径 (i,j),都将对该路径 (i,j) 进行局部信息素更新,目的在于减少该路径的信息素浓度。这将增加探索新路径的机会,避免算法陷入停滞。局部更新规则如下:

$$\tau_{ij}(t+1)=(1-\xi)\tau_{ij}+\xi\tau_0 \tag{12}$$

式中,ξ 满足($0<\xi<1$),τ_0 取为 $1/nC^{nn}$,是信息素初始值,C^{nn} 代表最邻近资源探索代价。

3 实验仿真分析

本文以大区域雷达功能仿真 MAS 为应用背景,并在此

背景下对本文动态任务分配机制的有效性进行实验分析。设MAS系统即时任务数为10，任务处理Agent初始数目为30。为模拟任务分配过程的动态性，定义一个整数变量$\varphi(-10\leqslant\varphi\leqslant10)$作为每次任务分配过程中加入和退出MAS任务分配的Agent数目，确定进行该动作的Agent由系统随机确定。参数设置：遗传算法编码位数与即时Agent数目一致，交叉概率P_c、变异概率P_m分别是均匀分布在区间(0.7,0.9)和(0.01,0.2)上的一个随机变量，模拟退火算子中初始温度T_0为100，温度更新参数μ为0.1，迭代结束温度ε为0.1。蚁群算法信息素因子α为2，启发因子β为3，信息素挥发系数ρ为0.1，q_0为0.9，ξ为0.1。因为系统消耗的时间代价是关键因素，所以将ω_1取为1，ω_2取为0。现将文献[1]中的HA[1]算法和文献[2]中HAGA[2]算法与本文的HAGA动态任务分配机制进行比较分析，表1是任务和Agent在不同规模时3种机制的优化比较，T为优化时间代价，G为迭代次数。图1为规模为10×30时的3种机制的进化曲线，采用样本无限逼近的方法得出，表示了优化时间和精度的关系。

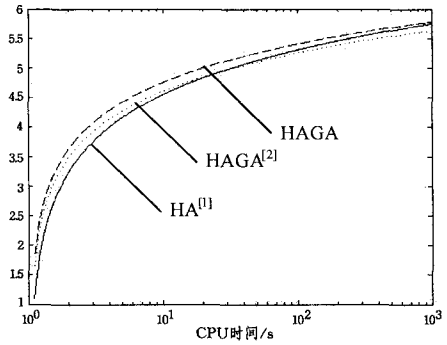

图1 优化进化曲线

表1 HA[1]、HAGA[2]和HAGA优化比较

	HA[1]		HAGA[2]		HAGA	
	T	G	T	G	T	G
10×20	38.19	451	32.59	367	30.13	321
10×24	40.13	479	32.93	385	30.87	335
10×27	43.93	529	36.89	476	31.19	369
10×31	47.85	539	37.61	519	32.49	487
10×38	55.03	545	45.32	527	33.57	495

由表1可知，从运行时间上来看，HAGA在Agent规模动态变化时的优势更明显。文献[1]的HA[1]算法Agent规模较大时其运行时间和精度已经不再适用。文献[2]的HAGA算法存在一个遗传和蚁群动态衔接的问题，只存在实验经验而没有理论衔接点。

结合图1优化进化曲线综合分析，在Agent规模动态增加并且算法迭代接近最优值的情况下，HAGA[2]算法运行时间大量增加，说明该机制不能保证在求解时间上的可靠性。两种算法的动态衔接问题成为影响算法性能的关键因素。在Agent规模为10×30的情况下，文献[1]的HA[1]算法在搜索初期要花费大量的时间，而文献[2]中HAGA[2]算法在算法运行后期性能明显下降。由此，可以综合得出，不论Agent规模是否动态变化，本文的HAGA动态任务分配机制适应性良好，保持了良好的时间效率和求解精度。

结束语 本文提出了MAS中动态任务分配问题的数学描述，建立了动态任务分配数学模型和相应的目标函数。应用了混合遗传蚁群算法对Agent动态任务分配进行优化组合求解。该算法的遗传算法部分加入了模拟退火算子，蚁群算法部分把全局信息素更新和局部信息素更新结合起来。仿真实验表明，该机制克服了传统遗传算法、蚁群算法易陷入局部最优和求解效低的问题，实现了两种算法的动态融合，提高了搜索速度和全局优化能力，对动态任务分配问题有效，并且在求解时效和精度上有良好的表现。但是，本文的仿真实验环境并非严格的分布式计算环境，任务分配的动态因素设置有限。在今后的研究工作中，设想将其他资源代价因素考虑在内。

参考文献

[1] 严建峰，李伟华，刘明. 基于混合蚁群算法的MAS任务分配[J]. 计算机应用研究，2009，26(1)：68-70

[2] 梁军，程显毅. 基于混合蚁群遗传算法的Agent联盟求解[J]. 计算机科学，2009，36(4)：227-231

[3] 赵玉兰. 基于混合蚁群遗传算法求解Agent联盟[J]. 信息与电脑，2010，6：111-113

[4] 徐金荣，李允，刘海涛，等. 一种求解TSP的混合遗传蚁群算法[J]. 计算机应用，2008，28(8)：2084-2088

[5] 杜宗宗，刘国栋. 基于混合遗传模拟退火算法求解TSP问题[J]. 计算机工程与应用，2010，46(29)：40-46

[6] 玄光男，程润伟. 遗传算法与工程优化[M]. 于歆杰，周根贵，译. 北京：清华大学出版社，2004

[7] 刑文训，谢金星. 现代优化计算方法[M]. 北京：清华大学出版社，2005

(上接第259页)

即发送补偿，而NORM协议则在一个时间片后集中发送补偿包，不利于数据的及时恢复。当补偿包数目较大时，会造成一定的抖动。

4)本文算法采用滑动窗口机制，根据媒体数据的Deadline决定是否对丢失补偿，而不是保证绝对可靠。可以避免无限制的回滚造成的延迟。

实验表明，结合网络编码的反馈轮机制更能适应无线网络环境高误码率和丢失率的特点，更适用于流媒体多播的传输。

参考文献

[1] Adamson B, Bormann C, Handley M, et al. NACK-Oriented Reliable Multicast Transport Protocol[Z]. rfc 5740, 2009

[2] Rizzo L. Effective erasure codes for reliable computer communication protocols[J]. ACM Sigcomm Computer Communication Review, 1997, 27(2): 24-36

[3] Sisalem D, Wolisz A. MLDA: a TCP_friendly congestion control framework for heterogeneous multicast environments[C]// Proc. of the 8th International Workshop on Quality of Service. Pittsburgh: [s. n.], 2000: 65-74

[4] Li S Y R, Yeung R W, Cai N. Linear Network Coding[J]. IEEE Transactions on Information Theory, 2003, 49(371)