基于混合蚁群遗传算法的Agent联盟求解

梁 军 程显毅
(江苏大学计算机科学与通信工程学院 镇江 212013)

摘 要 针对混合蚁群遗传算法容易融合时机过早或过晚、种群进化经历的代数过多、效率低等问题,首先改进了蚁群算法,并将改进的蚁群算法和遗传算法结合,应用于Agent联盟求解。提出了基于混合蚁群遗传算法的Agent联盟求解算法(Hybrid Ant Colony and Genetic Algorithm,HAGA),算法的核心是动态寻找两个算法的衔接点,在该点左侧使用遗传算法,右侧使用蚁群算法。与其他传统算法的实验比较,证明了该算法在求解联盟的最优解的时间和精度上都有较高的效果。把HAGA应用于RoboCup 2D龙队客户端程序中,使用比赛分析工具软件SoccerDoctor对比赛结果进行了统计分析,结果显示龙队在诸多技术参数方面均占有明显优势。

关键词 Agent联盟,蚁群算法,遗传算法,机器人足球比赛
中图法分类号 TP273+.22 **文献标识码** A

Solving Method of Agent Coalition Problem Based on Hybrid Ant Colony and Genetic Algorithm

LIANG Jun CHENG Xian-yi
(School of Computer Science & Communication Engineering, Jiangsu University, Zhenjiang 212013, China)

Abstract Aiming at such problems as too early or too late fusion of the hybrid ant colony, too many generations of the species evolution and low efficiency, the ant colony algorithm, first of all, got improved. And the connection of the improved ant colony algorithm and the genetic algorithm is applied to the problem-solving of the Agent-coalition. The algorithm of the agent coalition was put forward. The core to the algorithm dynamically searches for the joint point of these two algorithms in a dynamic way. The genetic algorithm is used in the left of this point, the ant colony algorithm in the right. Compared with the traditional algorithm experiments, these algorithms is highly efficient in the time and precision of the algorithm of the coalition. The HAGA was used in client end of RoboCup 2D Dragon Team in Jiang Su University. The match result was analyzed statistically by the analysis tool software—SoccerDoctor in China University of Science and Technology. The result shows that the Dragon Team has evident advantage in many technology parameters.

Keywords Agent coalition, Ant colony algorithm, Genetic algorithm, Robo cup

目前在MAS中,联盟生成的基本理论是N人合作对策理论,如Shapley值、核、核心等。Sandholm证明了有n个Agent的MAS系统的联盟结构总数是一个NP-Hard问题[1]。很多学者根据实际需要,对这个NP问题进行各种算法的探讨,研究了各种实际条件限制下的求解近似的最优联盟结构。

文献[2]研究在高层模式下执行任务的Agent的合作行为,提出了一种基于市场的传输协议,用于管理在群体中各个Agent的行为以及紧急情况处理。文献[3-5]提出了一些具有界限或任意时间等条件下的联盟生成算法,根据实际问题需要找出近似解,效率较高。

近年来很多学者不再满足于求得联盟的近似解,开始研究最优联盟的求解。郑金华[6]等运用遗传算法实现对Agent联盟的求解,求解精确解效率较高;文献[7]提出基于蚁群算法的单任务Agent联盟求解,在时间效率上比较有优势。文献[8]提出了遗传算法与蚁群算法融合的一般性优化问题求解策略,在对TSP问题的应用实验中取得了良好效果,该策略将遗传算法设置为运行固定的迭代次数,具有简单易行的优点,但这样容易造成融合时机过早或过晚。文献[9]将蚁群算法与遗传算法反复交叉,利用蚁群算法不断生成种群个体进行同步时序电路的初始化,这种优化方法能够尽可能多地初始化触发器,但每次种群进化都要经历几代运行,效率较低。

本文在上述研究的基础上,提出一种新的基于混合蚁群遗传算法的Agent联盟求解算法,主要创新之处为:(1)动态实现蚁群算法和遗传算法的衔接;(2)改进了蚁群路径选择规则和信息素更新规则。

1 联盟问题的描述

定义1 设一个MAS中有n个Agent,记$MAS=\{A_1, A_2, \cdots, A_n\}$。向量$B_i=(b_i^1, b_i^2, \cdots, b_i^j)(b_ij\geqslant 0, 1\leqslant i\leqslant n, 1\leqslant$

到稿日期:2008-05-13 本文受国家自然科学基金(60702056)资助。
梁 军(1976-),男,讲师,主要研究方向为模式识别、多Agent理论与应用,E-mail:liangjun@ujs.edu.cn;程显毅(1956-),男,教授,博士生导师,主要研究为模式识别、多Agent理论与应用。

$j \leqslant r$)表示Agent A_i 执行某一特定动作的能力。Agent 完成任务 t 所具有的能力需求表示为 $E_t=\langle b_t^1, b_t^2, \cdots, b_t^r\rangle$，完成任务获得的收益记为 $P(t)$。MAS 中的一个非空子集 C 称为一个 Agent 联盟，MAS 的一个完全划分 CS 称为该 MAS 的一个联盟结构。

定义 2 设联盟结构 $CS_k=\{C_1, C_2, \cdots, C_k\}$，联盟 $C_i \subseteq MAS$，$\bigcup_{i=1}^{k} C_i = MAS$，$C_i \cap C_j = \varnothing$，$i, j=1, 2, \cdots, k$，且 $i \neq j$。其中，联盟 C_i 中 Agent 个数定义为联盟 C_i 的大小，记为 $|C_i|$，联盟结构 CS_k 的大小为该联盟结构内含有的联盟个数，记为 $|CS_k|=k$。

定义 3 设所有大小为 k 的联盟结构的集合为 $L_k=\{CS_k\}$，所有的联盟结构为 $L=\bigcup_{k=1}^{n} L_k$，记向量 $B_c=\langle b_c^1, b_c^2, \cdots, b_c^r\rangle$ 是联盟中所有 Agent 能力向量的总和，联盟 C 可以完成任务 t 的必要条件是 $\forall 1 \leqslant i \leqslant r, b_t^i \leqslant b_c^i$。

定义 4 设 $U(C_i)$ 为联盟 C_i 的一个收益，则联盟结构 CS_k 的总收益可以表示为 $V(CS_k)=\sum_{i=1}^{k} U(c_i)$。

定义 5 设在 Agent 联盟结构中 $\forall C_i, C_j, 1 \leqslant i, j \leqslant n, i \neq j$，如果有 $U(C_i+C_j) \geqslant U(C_i)+U(C_j)$，则称联盟收益满足超可加性；如果有 $U(C_i+C_j) \leqslant U(C_i)+U(C_j)$，则称联盟收益满足次可加性。

定义 6 设有联盟结构 $CS_k=\{C_1, C_2, \cdots, C_k\}$，$CS_{k-1}=\{C_1, \cdots, C_{i-1}, C_i, \cdots, C_{j-1}, C_{j+1}, \cdots, C_k\}$。每个 CS_k 可以生成 $k(k-1)/2$ 个 CS_{k-1}。如果 $\{C_1, \cdots, C_{i-1}, C_i, \cdots, C_{j-1}, C_{j+1}, \cdots, C_k\}$ 是由 $\{C_1, C_2, \cdots, C_k\}$ 生成的，则记为 $CS_k \rightarrow CS_{k-1}$。如果有 $CS_k \rightarrow CS_{k-1}$，$CS_{k-1} \rightarrow CS_{k-2}$，则 $CS_k \rightarrow CS_{k-2}$，则该联盟生成满足传递性。

定义 7 将 n 个 Agent 组成的联盟结构用一个 n 层的联盟结构图表示，其中第 k 层是 L_k 中所有的 Agent 联盟结构。在相邻的两层 $k, k-1$ 之间，如果 $CS_k \rightarrow CS_{k-1}$，则在 CS_k 与 CS_{k-1} 间画一条线。从高层(L_n)到低层(L_1)可以看成是联盟的合并过程，从低层到高层可以看成是联盟的分解过程。联盟的形成与演化正是由联盟的合并与分解组成。

给定所有联盟 C_i 的收益，联盟结构优化问题就是找出收益最大的 Agent 结构，即最优联盟结构 CS^*，$V(CS^*)=\arg\max_{CS \in L} V(CS)$。联盟 C_i 的效用 $U(C_i)$ 取决于完成任务所获得的利益以及联盟中各成员完成任务所付出的代价和其他花费，可表示为：$U(C_i)=P(t)-F(C_i)-C(C_i)$，其中 $P(t)$ 表示完成任务 t 所获得的利益，$F(C_i)$ 表示联盟 C_i 中各个成员 Agent 完成任务 t 所付出的总代价，$C(C_i)$ 表示联盟 C_i 求解任务 t 花费的额外开销，如联盟中各 Agent 之间的通信开销。由此，Agent 联盟求解问题就转换成了优化问题。鉴于进化算法在解决优化问题中的良好表现，所以本文考虑使用遗传算法和蚁群算法。

2 HAGA 算法

2.1 相关算法简介

1)遗传算法(Genetic Algorithm，GA)

美国 Michigan 大学 J. Holland 教授于 1975 年提出的遗传算法，以达尔文生物进化理论“适者生存，优胜劣汰”和孟德尔遗传变异理论“生物遗传进化主要在染色体上，子代是以父代遗传基因在染色体上的有序排列”为基础，模拟生物进化过程。GA 在自适应控制、组合优化、模式识别、机器学习、规划策略、信息处理等领域的应用中展示了明显的优越性。

GA 的主要优点是：(a)具有领域无关的群体性全局搜索能力，可避免陷入局部最优；(b)搜索使用评价函数启发，过程简单；(c)使用概率机制进行迭代，具有随机性；(d)可扩展性强，易于介入到已有模型中，且易于与其他优化技术结合。

GA 的主要缺点是：(a)没有充分利用系统反馈信息，使搜索具有盲目性；(b)算法求解到一定范围时往往做大量冗余迭代，向最优解收敛的速度大大降低，导致求最优解效率较低。

2)蚁群算法(ant colony optimization，ACO)

意大利学者 M. Dorigo 在 1992 年首次提出了蚁群算法。研究表明，几乎不具备视觉的蚁群之所以能够找到蚁巢到食物之间的最短路径，是靠其在走过的路上释放一种挥发性分泌物(即信息素，pheromone)来实现的。后来的蚁群选择该路径的概率与当时这条路径上信息素的强度成正比。当某路径上通过的蚁群越来越多时，在此路径上留下的信息素也越来越强，使得后来的蚁群选择此路径的概率更高，从而更增加了此路径上的信息素强度，可吸引更多的蚁群选择此路径。这样就形成了一种正反馈机制，使蚁群最终可发现最短路径。蚁群算法在许多组合优化问题上取得了较好的效果。如 TSP 问题、一次分配问题和车间作业调度问题等。

ACO 的主要优点是：(a)具有正反馈机制，通过信息素的不断更新，高效收敛到最优解；(b)具有通用随机优化特性并在蚁群中融入人类智能；(c)是一种分布式优化方法，有利于并行计算；(d)是一种全局优化方法，单目标优化和多目标优化问题均可求解。

ACO 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 算法改进

1)改进蚁群路径选择规则和信息素更新规则

ACO 的路径转移概率使用固定的公式：

$$p_{ij}^k(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in U}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}}, & j \in U \\ 0, & \text{other}\end{cases} \tag{1}$$

在算法运行初期，如果问题的规模较大，则蚁群很难在众多路径中找出一条较好的路径。为了加快算法的收敛，蚁群路径选择的概率值应当取比较大的数，这样较好的路径被选择的概率就较高，较优联盟中的 Agent 容易找到。随着算法的运行，正反馈的作用越来越明显，一些路径上的信息素明显高于其他路径，但这些路径不一定是最优的，因而容易停滞。为了缓解正反馈过程中容易陷入局部最优解，进化过程中做了适当改进和调整。这时，应该减少路径选择的概率，使解的搜索更多样化，才有利于找到全局最优解。因而，可设第 k 只蚂蚁在节点 i 选择下一个节点 J 的规则是：

$$J=\begin{cases}\arg\max_{u \in U}\{[\tau_{iu}(t)]^{\alpha}[\eta_{iu}(t)]^{\beta}\}, & \text{if } q \leqslant q_0 \\ J, & \text{other}\end{cases} \tag{2}$$

其中，q 是 0 到 1 中均匀分布的一个随机数，q_0 是预先设置的一个阈值，$q_0 \in (0,1)$。J 是服从概率分布 $p_{ij}^k(t)$ 的一个随机变量。

而且，ACO 采用信息素随时间挥发的方程为：

$$\tau_{ij}(t+1)=\rho \cdot \tau_{ij}(t)+\Delta\tau_{ij} \tag{3}$$

其中信息素挥发因子为[0,1]固定常量，这样在算法运行初期，信息素比较小，挥发因子相对比较大，这样信息素浓度比较大的边消逝得更快，容易削弱它们的优势，造成算法行进的速度慢；而在算法运行后期，信息素比较大，挥发因子就相对比较小，信息素浓度比较大的边更容易保持其信息素，正反馈的优势更明显，算法容易过早收敛于非全局最优解。为此，在算法运行初期，应当尽量减小挥发因子，采用较小的更新系数；在算法运行后期，应当尽量增大挥发因子，采用较大的更新系数。

因此，信息素更新规则可以采用：

$$\tau_{ij}(t+1)=\rho\cdot\tau_{ij}(t)+\lambda\Delta\tau_{ij} \tag{4}$$

其中，$(1-\rho)$是挥发因子，$\rho=0.9^{1+\frac{1}{100}N}$；$\lambda$是信息素更新系数，$\lambda=1.001^N$；$N$为蚁群算法循环的次数。

同时借鉴 Tomas Stutzle 提出的 MMAS (max-min ant system)算法的思想，将各路径的信息素大小控制在$[\tau_{\min},\tau_{\max}]$之间，如式(5)所示，这样可以比较有效地避免算法早熟问题。

$$\tau_{ij}(t+1)=\begin{cases}\tau_{\min}, & \text{if } \tau_{ij}(t)\leqslant\tau_{\min}\\ \tau_{ij}(t), & \text{if } \tau_{\min}\leqslant\tau_{ij}(t)\leqslant\tau_{\max}\\ \tau_{\max}, & \text{if } \tau_{ij}(t)\geqslant\tau_{\max}\end{cases} \tag{5}$$

开始取 $N=0$，$\rho=0.9$，$(1-\rho)=0.1$，$\lambda=1$。此时 ρ 比较大，挥发因子$(1-\rho)$比较小，信息素更新系数 λ 比较小，这样既有利于信息素浓度大的路径脱颖而出，又能避免过早收敛。

随着算法的运行，ρ逐渐变小，$(1-\rho)$逐渐变大，λ逐渐变大。

在算法运行后期，ρ比较小，挥发因子$(1-\rho)$比较大，信息素更新系数λ比较大，能改善 GA 和 ACO 后期算法容易停滞等缺点，加快算法收敛速度。

2)蚁群算法和遗传算法的动态衔接

文献[10]提出的蚁群算法和遗传算法的动态融合策略方法是首先设置最小遗传迭代次数 $Gene_{\min}$ 和最大遗传迭代次数 $Gene_{\max}$；然后统计遗传算法迭代过程中子代群体的进化率，并以此设置子代群体最小进化率 $Gene_0$；最后，如果在设定的迭代次数范围内，连续子代群体的进化率都小于 $Gene_0$，则遗传算法过程结束进入蚂蚁算法。这种方法的主要缺点是：a)$Gene_{\min}$，$Gene_{\max}$和 $Gene_0$ 不容易预先设定或设定不够准确，仅凭经验或实验估计缺乏理论依据；b)未真正实现蚁群算法和遗传算法的动态衔接，未体现连接的自主性。本文通过分析文献[10]，并在其基础上考虑到遗传算法进化到一定阶段以后，适应度变化缓慢，提出采用样条曲线对遗传算法的进化曲线建模，通过发现曲线的拐点而自动搜索到动态衔接的最优位置，以实现蚁群算法和遗传算法的动态衔接，体现了衔接的自主性。由于篇幅原因，这里不作介绍。

2.3 HAGA 描述

```
step 0  初始化遗传算法，随机生成初始种群 P(0)，t=0；
step 1  while (t≤T and 不是拐点)
        do { for i=1 to M //适应度函数
             do {计算当前群体 P(t)每个个体 Ci 的适应度函数 f(Ci)；}
           for i=1 to M //选择
             do {根据当前种群 P(t)的每个个体适应度函数 f(Ci))选择生成中间群体；}
           for i=1 to M //交叉
             do {以概率 Pc 选择两个个体进行交叉并将新个体替换原个体插入到种群 P(t)中；}
           for i=1 to M //变异
             do {以概率 Pm 选择某个个体进行变异并将新个体替换原个体，插入到种群 P(t)中；}
           for i=1 to M
             do P(t+1)=P(t)；
           t=t+1；}
step 2  if(是拐点 and t≤T) 生成中间优化解向量 X=P(t+1)；
step 3  初始化蚁群算法，t=0；N=0；
        for 每条边(i,j)
          do{ 根据中间优化解向量 X 得到初始信息素分布 τij(t)并 Δτij^k=0；}
step 4  设置当前尚未访问的 Agent 集合 Jk={a1,a2,…,an}，随机放置 m 只蚂蚁到 n 个 Agent 上；
          for k=1 to m
            do {记第 k 只蚂蚁所在的 Agent 为 ai 并将 ai 从 Jk 中删除，计算初始联盟能力向量 BCk；}
step 5  for k=1 to m //路径选择
        do while(BCk<Bt)
          { 根据式(2)选择下一个 Agent j，并将该蚂蚁移到 Agent j，
            将 aj 从 Jk 中删除，计算当前联盟能力向量 BCk；}
step 6  for k=1 to m //计算当前最优联盟
        do { 根据式 u(Ci)=P(t)-F(Ci)-C(Ci)计算第 k 只蚂蚁所形成的联盟的效用，求出 m 个联盟中效用最大的联盟作为当前找到的最优联盟；}
step 7  for 每条边(i,j) //信息素的更新
          do{ 根据公式 Δτij=Σ_{k=1}^{m} Δτij^k 和 Δτij^k=u(Ck)/Σ_{k=1}^{m} u(Ck)计算 Δτij 和 Δτij^k；}
        for 每条边(i,j)
          do {根据式(4)计算 τij(t+1)；}
step 8  t=t+1；N=N+1；
        for 每条边(i,j) Δτij=0；
        do {if (N< Nmax并且算法无停滞)
             { 转 Step 6；}
           else 输出最优联盟及其效用值，终止程序；}
```

3 算法分析

3.1 复杂性分析

HAGA 的运行时间 T_{HAGA} 主要由遗传算法运行时间 T_{GA}、蚁群算法运行时间 T_{ACO} 和衔接部分运行时间 T_J 组成，表示为 $T_{HAGA}=T_{GA}+T_{ACO}+T_J$。HAGA 中的遗传算法与蚁群算法过程采用相同的适应值计算规则，因此可以使用一致的时间值 T_{cf} 表示这两个算法中计算每个划分方案适应值所需的时间。下面分别对 T_{GA}，T_{ACO} 和 T_J 的大小进行分析。

1)T_{GA} 包括遗传算法初始设置时间 $T_{GA\text{-}init}$、终止条件判断时间 $T_{GA\text{-}termi}$ 和种群进化迭代时间。假设遗传算法进化了 I_{GA} 代，种群规模为 N；种群中每个个体完成一次遗传操作的过程所需的时间为 $T_{GA\text{-}op}$ 那么种群迭代进化时间就是 $I_{GA}\times N\times(T_{cf}+T_{GA\text{-}op})$。这样 $T_{GA}=T_{GA\text{-}init}+T_{GA\text{-}termi}+I_{GA}\times N\times(T_{cf}+T_{GA\text{-}op})$，而 $T_{GA\text{-}init}+T_{GA\text{-}termi}$ 远小于 $I_{GA}\times N\times(T_{cf}+T_{GA\text{-}op})$。因此 $T_{GA}\approx I_{GA}\times N\times(T_{cf}+T_{GA\text{-}op})$。

2)T_{ACO} 包括蚁群算法初始设置时间 $T_{ACO\text{-}init}$、终止条件判断时间 $T_{ACO\text{-}term}$ 和蚁群算法迭代时间。假设蚁群算法迭代

I_{ACO}次。每次迭代中使用 m 只蚂蚁，每只蚂蚁完成一次任务图着色的时间为 $T_{ACO\text{-}op}$，那么蚁群算法的迭代时间就是 $I_{ACO}\times m\times(T_{ACO\text{-}op}+T_{cf})$。这样，$T_{ACO}=T_{ACO\text{-}init}+T_{ACO\text{-}termi}+I_{ACO}\times m\times(T_{ACO\text{-}op}+T_{cf})$，而 $T_{ACO\text{-}init}+T_{ACO\text{-}termi}$ 远小于 $I_{ACO}\times m\times(T_{ACO\text{-}op}+T_{cf})$。因此，$T_{ACO}\approx I_{ACO}\times m\times(T_{ACO\text{-}op}+T_{cf})$。

3) T_J 包括构造优化解集合所需的时间 $T_{J\text{-}better}$ 和计算信息素初始所需的时间 $T_{J\text{-}ip}$。由于采用样条曲线拟合遗传算法进化曲线，不涉及大量的数据运算，所以 $T_{J\text{-}better}$ 的时间有限。假设任务图节点数为 K，边数为 H，每次计算一条边上的信息素值的时间为 $T_{J\text{-}ep}$，那么，信息素初值计算时间 $T_{J\text{-}ip}$ 就是 $N\times H\times T_{J\text{-}ep}$。这样 $T_J\approx N\times H\times T_{J\text{-}ep}$。

由上述分析可知，$T_{HAGA}\approx I_{GA}\times N\times(T_{cf}+T_{GA\text{-}op})+I_{ACO}\times m\times(T_{ACO\text{-}op}+T_{cf})+N\times H\times T_{J\text{-}ep}$。我们还发现 T_{cf} 的复杂度为 $O(K\times H)$，远大于 $T_{GA\text{-}op}$，$T_{ACO\text{-}op}$ 以及 $N\times H\times T_{J\text{-}ep}$。因此可以简化得 $T_{HAGA}\approx I_{GA}\times N\times T_{cf}+I_{ACO}\times m\times T_{cf}=(I_{GA}\times N+I_{ACO}\times m)\times T_{cf}$。

在运行过程中遗传算法部分所需的存储空间约为 $2\times N\times O(K)$，蚁群算法部分所需的存储空间约为 $m\times O(K+H)$，算法衔接部分构造优化解集合所需的存储空间为 $N\times O(K)$。当然不同的程序实现对算法运算时间与存储空间也有一定的影响。

3.2 运行效率分析

上述复杂性分析表明，HAGA 的运行时间 T_{HAGA} 主要取决于 $I_{GA}\times N+I_{ACO}\times m$ 与 T_{cf} 的乘积。其中，T_{cf} 是对某个划分解进行评价所需的时间，与所用的优化算法无关。对于特定的组合优化问题，T_{cf} 本质上是评价问题空间中一个解所需的时间。因此 HAGA 效率改进的关键在于最小化 $I_{GA}\times N+I_{ACO}\times m$，其中 N，m 是控制参数，分别表示遗传算法的种群规模和蚁群算法的蚂蚁数，一般是预先设定的固定值。当遗传算法迭代次数 I_{GA} 为 0 时，HAGA 就退化为蚁群算法；当蚁群算法迭代次数 I_{ACO} 为 0 时，HAGA 则退化为遗传算法。

4 实验结果及分析

实验中主要采用的是集中式计算，没有模拟真实的分布式计算，集中在一起计算所有潜在联盟的联盟值，并不影响系统总收益的值，但会对算法执行的时间和通信复杂度造成一定的影响。

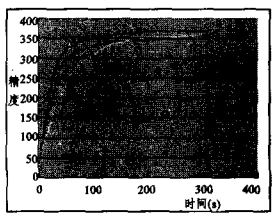
首先对 Shehory 等人的算法[3]（简记为 SHA）和 HAGA 两种算法的系统总收益进行比较，以检验这两种算法的优劣，结果如表 1 所列，其中：$V(CS)$ 表示总收益，T_{SHA} 表示 SHA 运行时间，T_{HAGA} 表示 HAGA 运行时间。实验部分参数为：No. 1：$n=10$，$m=4$，Agent 个数 10；No. 2：$n=20$，$m=8$，Agent 个数 20；能力 r 均为 10；联盟大小 $k=3$；算法各运行 5 次。

表 1 SHA 和 HAGA 两种联盟形成机制的结果对比

No.	第一次				第二次			
	SHA		HAGA		SHA		HAGA	
	V(CS)	T_{SHA}	V(CS)	T_{HAGA}	V(CS)	T_{SHA}	V(CS)	T_{HAGA}
1	29	0.71	66	1.17	106	16.00	295	28.41
2	9	0.82	127	1.39	83	16.47	227	28.25
3	20	0.72	57	1.27	69	14.83	197	27.37
4	19	0.71	104	1.22	66	15.43	190	27.03
5	13	0.89	98	1.29	89	16.33	243	28.55

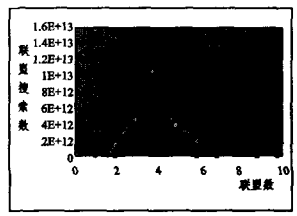
从运行时间上看，SHA 时间很短约为 HAGA 的一半。通过分析，我们认为主要原因是 SHA 能够形成的联盟比较少，有些任务不能完成，所以系统总收益也较少。当然，我们还要考虑到，因为没有进行模拟真实的分布式计算，运行时间不一定准确，所以只能作参考。但 HAGA 能够形成更多的联盟，能够完成更多的任务，所以运行时间长一些，系统总收益也必然较大。从表 1 可以看出，不论 $n=10$ 还是 20，SHA 的系统总收益均比 HAGA 系统总收益小得多。

接着对 HAGA，GA[5] 和 ACO[6] 求解 Agent 联盟问题的比较。实验参数：$n=30$，采用 30 个 Agent，遗传编码位数为 30 位，遗传算法种群规模取为 50，遗传迭代次数为 30，交叉概率 $P_c=0.8$，变异概率 $P_m=0.05$。蚁群算法 $\alpha=2$，$\beta=3$，$\rho=0.9^{1+\frac{1}{100}N}$，$\lambda=1.001^N$，$\tau_{\min}=10$，$\tau_g=30$，取 $\tau_{\max}=600$，迭代次数为 20。图 1 显示了这 3 种算法的优化解进化曲线，结果显示 HAGA 在解的精度和时间上，都明显优于 GA，ACO 两种算法。图 2 中 x 轴对应每层 Agent 联盟的结构图，y 轴是需要搜索的联盟结构数。在 x 轴从 20 到 1 的变化过程中，开始时，Agent 联盟结构中具有同构关系的 Agent 联盟结构较少，因此，HAGA 算法与其他两个算法相差不大，但是在变化到 3 到 7 层时，每一层的 Agent 联盟结构中，具有同构关系的 Agent 联盟结构非常多，因此，HAGA 经过挥发因子 $(1-\rho)$ 抑制以后，可以显著减少需要搜索的 Agent 联盟结构个数。这说明 HAGA 能较高效地找到 Agent 联盟的优化解，是多 Agent 联盟合作求解的一种很有效方法。



(a)

图 1 最优进化曲线比较



(b)

图 2 Agent 联盟结构图和需要搜索的结构数比较

最后将 HAGA 应用于 RoboCup 仿真比赛客户端程序中。实验使用 3 台机器，一台机器用于运行 SoccerServer 和 Monitor，另外两台机器分别运行在 Linux 下，对应的球队分别为江苏大学龙队（含 HAGA 算法的程序）和江苏大学超新星队（曾获得 2004，2005 年全国机器人足球赛二等奖，三等奖）。RoboCup 仿真比赛在 6000 * 10 个仿真周期中进行，然后使用中国科技大学比赛分析工具软件 SoccerDoctor（http://robocup.ai.ustc.edu.cn/sim/tools/SoccerDoctor.exe），对比赛结果进行了统计分析，如表 2 所列。结果显示前者面对超新星队时，无论在 Shoot-Success 还是在 Defensive-Success 等重要方面均占有明显优势。

表 2 实验统计数据

	龙队	超新星队
射门次数	8	3.5
进球数	4.6	1.2
射门成功率	57.5%	34.2%
控球时间	42.5%	23.1%
传球成功率	64.7%	38.2%
体力值	75.5	67.8
防守成功率	73.1%	64.5%

结束语 本文的创新之处是提出了 HAGA 来求解 Agent 联盟问题，该算法融合并改进了遗传算法和蚁群算法两种仿生算法，实现了对这两种算法的动态衔接。HAGA 在遗传算法运行过程中加入了子代优化解改进效率检测机制，使得当遗传算法对优化解改进效率降低到一定程度时（即进化拟合曲线拐点处），自动终止遗传过程，从而避免 I_{GA} 无谓的增长。并且把优化解转换为较为准确的初始信息素开始蚁群算法的执行，使蚁群算法大大降低了用于形成最优解上信息素所需的迭代次数。这样利用蚁群算法正反馈、高效收敛的优势，可以在 I_{ACO} 很小的情况下迅速找到最优解。因此，HAGA 通过抑制 I_{GA} 与 I_{ACO} 无谓的增长，从而最小化 $I_{GA}\times N+I_{ACO}\times m$ 来达到提高搜索效率的目的。通过对 HAGA 复杂性分析和运行效率分析进一步说明了 HAGA 在 Agent 联盟求解问题上的优势。对比实验以及 HAGA 在 RoboCup 中的应用也都表明 HAGA 在联盟求解问题上取得较满意效果。值得一提的是本文得到的是一种通用的联盟算法，可以通过参数变化适应不同的联盟对象（比如企业动态联盟中供应商选择联盟与零售商选择联盟有很大的不同），从而充分体现出具体 Agent 联盟的特点；第二，样条逼近曲线需要大量的计算时间，必将降低遗传算法的效率，实验中我们改进了样条逼近曲线方法，由于篇幅原因不作介绍；第三，我们多次实验证明了得出的曲线拐点就是最佳的两种算法的交接点，但缺乏理论依据。

参 考 文 献

[1] 程显毅. Agent 计算 [M]. 哈尔滨：黑龙江科学技术出版社，2003

[2] Cornforth D, Kirley M, Bossomaier T. Agent Heterogeneity and Coalition Formation: Investigating Market-Based Cooperative Problem// IEEE Computer Society AAMAS'04. July 2004: 556-563

[3] Sandholm T W, Larson K, Andersson M R, et al. Anytime coalition structure generation with worst case guarantees // Proceedings of the 15th National conference on Artificial Intelligence. Menlo Park, AAAIPress, 1998: 46-54

[4] Shehory O, Kraus S. Methods for task allocation via Agent coalition formation. Artificial Intelligence, 1998, 101(1-2): 165-200

[5] Jennings N R, Dang V D. Generation coalition structures with finite bound from optimal guarantees // Proc. of the AAMAS. 2004, 2: 572-579

[6] Zheng Jin-hua, Chen Zhen-zhou, Cai Zi-xing. Multi-Agent Coalition Formation Based on Genetic Algorithms. Computer Engineering & Science, 2004(6)

[7] Xia Na, Jiang Jianguo, Wei Xing, et al. Searching for Agent Coalition for Single Task Using Improved Ant Colony Algorithm. Journal of Computer Research and Development, 2005, 42(5): 734-739

[8] Ding J L, Chen Z Q, Yuan Z Z. On the combination of genetic algorithm and ant algorithm. Journal of Computer Research and Development, 2003, 40(9): 1351-1356

[9] Li Z, Xu C P, Mo W, et al. Initialization for synchronous sequential circuits based on ant algorithm&genetic algorithm. Acta ElectronicaSinica, 2003, 31(8): 1276-1280

[10] Xiong Zhi-hui, Li Si-kung, Chen Ji-hua. Hardware/Software Partitioning Based on Dynamic Combination of Genetic Algorithm and Ant Algorithm. Journal of Software, 2005, 16(4): 503-512

(上接第 217 页)

lar 为 0.9050；当要求服务下载 a，c 类信息时，peer1 与 B 的内容相似度 *Similar* 为 0.7525；当要求服务下载 d，f 类信息时，peer1 与 B 的内容相似度 *Similar* 为 0.9524；同一节点 B 由于要求服务的内容不同与节点 peer1 的内容相似度也不同，内容相似度的大小直接影响到推荐节点推荐值。

通过以上关于节点间内容相似度的分析，应用式(5)和式(6)计算 peer2 的全局信任值。当需要下载 d，f 类信息时 peer2 全局信任值为 0.4212，如果不考虑部分相似度 peer2 的全局信任值仅为 0.3376；交易结束后 peer1 修改 B，C 的推荐可信值$\langle p,c\rangle$，同时根据文献[10]peer2 对 peer1 所提供的服务做出评价。

结束语 本文在已有 P2P 模型的基础上提出了基于内容相似度和推荐反馈计算节点推荐值的对等网络信用模型 IPBS。通过计算该节点间的内容相似度来评价节点提供推荐服务的能力，节点间相似度是根据每次交易的内容不同改变节点的相似度值；IPBS 通过历史交易时间权值来调整相应的推荐值，根据交互的结果对推荐者的推荐分配不同的权重，基于上述 3 方面的考虑可以加强模型的动态适应能力和搜索服务的效率。

参 考 文 献

[1] Shaw M. Everyday dependability for everyday needs // Supplemental Proceedings of the 13th International Symposium on Software Reliability Engineering. IEEE Computer Society, 2002: 7-11

[2] Stephen M. Formalising trust as a computational concept. Ph. D. dissertation. University of Stirling, Scotland, 1994

[3] Xiong L, Liu L. Peer Trust: Supporting reputation-based trust in peer-to-peer communities. IEEE Transactions on Data and Knowledge Engineering, Special Issue on Peer-to-Peer Based Data Management, 2004, 16 (7): 843-857

[4] Abdul-Rahman A, Halles S. A distributed trust model // Proceedings of the New Security Paradigms Workshop'97. Cumbria, U K, 1997: 48-60

[5] Abdul-Rahman A, Hailes S. Supporting trust in virtual communities // Proceedings of the 33rd Hawaii International Conference on System Sciences. Maui, Hawaii, 2000: 4-7

[6] Beth T, Borcherding M, Kleinl B. Valuation of trust in open network // Proc. of the European Symp on Research in Security (ESORICS). Berlin: Springer Verlag, 1994: 13-18

[7] Gambetta D. Trust. Oxford, Blackwell, 1990

[8] Zeng C, Xing C X, Zhou L Z. Similarity measure and instance selection for collaborative filtering // Bakonyi P, Hencsey G, et al. , eds. Proc. of the 12th Int'l World Wide Web Conf. (WWW 2003). New York: ACM Press, 2003: 652-658

[9] 陈颖熙，李贤有. 基于内容相似度的对等网络信用模型研究[J]. 计算机科学，2007，34(8)

[10] Bobba R B, Eschenauer L, Gligor V, et al. Bootstrapping security associations for routing in mobile ad hoc networks // Proc. IEEE GLOBECOM. San Francisco, CA, Dec. 2003: 1511-1515

[11] Zhang Lin, Xu Feng, Wang Yuan, et al. A Semantic and Time Related Recommendation-Feedbacd Trust Model // Second International Conference on Availability. Reliability and Securty (ARES'07) IEEE. 0070-7695-2775-2/07

[12] 石志国，贺也平，张宏. 一种对等计算安全性的时间自衰减信任管理算法[J]. 计算机研究与发展，2007：1-10