文章编号:1001 - 9081(2006)07 - 1706 - 03

基于 JADE 的并行遗传算法的设计与实现

张秋余,黄 鹏,迟 宁
(兰州理工大学 计算机与通信学院,甘肃 兰州 730050)
(hp31@163.com)

摘 要:为解决传统遗传算法运行时间过长、寻优率偏低的问题,在研究简单遗传算法的基础上,利用 JADE,提出了一种基于多 Agent 协同工作的并行遗传算法。该算法实现了对客户容器动态加入参与运行的支持。实验结果表明,该算法能较明显地提高传统遗传算法的运行效率和寻优的成功率。

关键词:并行遗传算法;Java Agent Development Environment(JADE);多 Agent
中图分类号: TP301.04 **文献标识码**:A

Design and realization of parallel genetic algorithm based on JADE

ZHANG Qiu-yu, HUANG Peng, CHI Ning
(*School of Computer and Communication, Lanzhou University of Technology, Lanzhou Gansu* 730050, *China*)

Abstract: For solving the problems of the traditional genetic algorithms, the excessively long runtime and the somewhat low rate of seeking the optimal results, a parallel genetic algorithm based on the cooperation of multi-agents was presented, with utilization of JADE development platform. Based on simple genetic algorithms, this algorithm implemented the support for client container dynamic joining into the system to run. The experiment results show that the algorithm raises working efficiency of traditonal genetec algorithms and increases success rate to seek the optimal solution.

Key words: parallel genetic algorithm; Java Agent Development Environment(JADE); multi-agent

0 引言

遗传算法(Genetic Algorithm,GA)是一类基于遗传机制和自然选择原理的随机搜索算法[1]。它引进生物学中基因遗传和"自然选择,适者生存"的进化论思想,将优化问题的求解看成可行解的进化过程。一般地,遗传算法以一群随机产生的可行解开始,每个解用遗传编码表示为个体,由优化目标函数确定个体的适应度对个体进行评价,通过交叉、变异等遗传算子的操作对种群进行组合产生下一代个体,逐步向优化的种群进化。

随着计算机网络,特别是 Internet 技术的日益成熟和普及,并行地运行多个遗传算法并让它们之间进行个体交换以组成粗粒度并行遗传算法,已是一种广为接受的遗传算法并行策略[1]。而传统意义上的并行,往往是考虑到多台同构机器组成的机群系统,并没有考虑到实际应用中机器可能存在的异构性。为了能够对实际网络和计算机进行有效的利用以及达到平台无关性,因此选择 Java 作为开发工具,利用 Agent 技术,提出一种抛开计算机硬件结构的分布式遗传算法。

1 系统拓扑结构及算法

1.1 JADE[2~4]

JADE(Java Agent Development Environment)是严格遵守 FIPA 规范的多 Agent 系统开发平台,JADE 编程人员可以在构建 Agent 的时候使用纯 Java 进行开发。同时,由于 JADE 简化了各 Agent 之间的通信,通过符合 FIPA 规范的消息来传递信息,甚至可以在消息中嵌入可序列化的对象,实现参数传递的规范性。另外,由于 JADE 提供的 DF(Directory Facilitator),使得系统的黄页功能能够直接被利用,方便了客户容器的注册。通过 JADE 实现的 AMS(Agent Management System)和 Sniffer 工具,用户可以很方便地对所实现的平台进行调试,完善系统功能。

1.2 系统拓扑结构

本文所涉及的多 Agent 系统结构见图 1。由于采用的是 JADE,所以各个 Agent 位于不同的容器中[2~4]。而系统中主要包含了两种容器,服务容器和客户容器。系统的整个功能的实现就是靠这两类容器之间的消息传递来进行控制的。

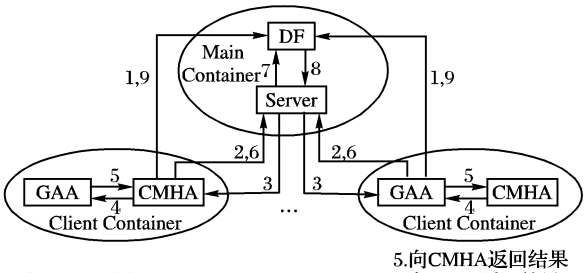

图 1 多 Agent 系统拓扑结构

在该结构中,处于容器之内的就是我们所要实现的具体 Agent。对于服务器容器,包含 DF 和 Server 两种 Agent。DF 实现的是客户容器中消息处理 Agent 的注册服务,实际上,在 JADE 中,DF 的功能是由主容器来实现的,但是为了便于表示,我们不妨将其认为是服务器容器实现的 Agent。Server 主要实现的功能是对注册的客户容器中的代理进行任务分配,并对客户返回的结果进行选择和输出。对于客户容器,主要

收稿日期:2006 - 01 - 16;**修订日期**:2006 - 02 - 28 **基金项目**:甘肃省科学技术攻关计划项目(2GS047-A52-002-03)

作者简介:张秋余(1966 -),男,河北辛集人,副研究员,主要研究方向:软件工程、数据库、信息安全、制造业信息化; 黄鹏(1980 -),男,福建福安人,硕士研究生,主要研究方向:信息安全; 迟宁(1976 -),男,陕西西安人,硕士研究生,主要研究方向:软件工程.

实现了 CMHA(Client Message Handler Agent)和 GAA(Genetic Algorithm Agent)两种 Agent。CMHA 就是客户消息处理代理,它的功能主要是向服务容器发送注册请求进行注册,从服务容器接收初始任务参数,向 GAA 发送运行指令,接收 GAA 运行结果并反馈给服务容器。GAA 就是具体执行遗传算法的 Agent。

1.3　算法结构

本文提出的并行遗传算法(Parallel Genetic Algorithm, PGA)由若干独立计算单元组成,每个单元运行一个简单遗传算法。从 MAS(Multi Agent System)的观点看,每个计算单元都是独立的 Agent。

定义　$PGA = (M, SubGA, t, T)$

在这里 $SubGA$ 是各客户容器所执行的遗传算法,t 是个体迁移参数,T 是算法终止条件,各参数的意义如下:

算法维数　即并行算法 PGA 中客户容器的个数 M。

客户端算法　客户端算法 $SubGA$ 可以是任何一种简单遗传算法,如最优保持遗传算法、稳态遗传算法。客户端算法用 $Ai, i = 1, 2, \cdots, M$ 表示,它们在结构上是完全相同的。在本系统中采用的就是简单遗传算法,但是由于对遗传算法进行单独的封装,因此可以方便地进行替换。

个体迁移参数 t　服务容器和客户之间交换个体的操作用 t 表示。这个参数由服务容器设定,在规定的时间内,客户容器必须对服务容器进行种群择优传递,否则服务容器就会在注册池内将该客户容器删除。在本系统中设定传递参数的个数为 1,即选择最优结果进行传递。

停止准则　用以决定算法何时停止运行。在本系统中,为了便于操作将该条件设定为运行代数达到服务容器设定值时即停止。

这个系统涉及到的具体算法主要包括了 DF 运行算法,Server 算法,CMHA 算法以及 GAA 算法。在 JADE 中,DF 实际上是由主容器[2~4]实现的。下面重点来讨论 Server 算法、CMHA 算法和 GAA 算法。

```
Procedure Server
Begin
  if (初次请求未到达) Then
    等待;
  Else
  Begin
    req_num ++;                          //请求客户数加 1
    generation ++;                       //已运行代数加 1
    开始发送初始运行消息;
  End;
  while( generation < MAX_GEN)           //运行代数小于规定值
  Begin
    If( 收到结果信息) Then
    Begin
      If (消息中结果 > 现有最好结果)
        用消息中结果替换现在最好结果;
      req_num --;
      //一个客户结束,要将总客户数减 1
      If (该客户响应超时) Then
        从注册池中删除该客户;
      If( req_num == 0) Then             //所有结果已返回
      Begin
        查询 DF 中提供的所有可用客户;
        将下次运行参数封装成消息进行广播;
        genertion += 可用客户数;
      End;
    End;
    If( 收到出错消息) Then
    Begin
      向 CMHA 发送终止消息;
      req_num --;
    End;
  End
  输出结果;
  结束客户端;
End
Procedure CMHA
Begin
  Do Begin
    向 DF 发送注册请求;
    向 Server 发送运行要求;
  End While( 没有收到响应);
  While( 没有收到结束消息)
  Begin
    If (收到 Server 运行消息) Then
      向 GAA 发送运行消息及运行参数;
    If( 收到 GAA 结果信息) Then
    Begin
      If( GAA 运行超时) Then
      Begin
        将结果置为 0;
        向 GAA 发送结束消息;
        取消在 DF 中的注册;
        向 Server 发送出错消息;
      End;
      将结果转发到 Server;
    End;
  End;
  向 GAA 发送结束消息;
End
Procedure GAA
Begin
  While( 没有收到结束消息)
  Begin
    If (收到 CMHA 运行消息)
      初始化种群;
    If (Server 不是第一次运行)          //说明是后插入的客户
      在种群中插入 Server 当前最好个体;
    开始遗传运算;
    将结果返回 CMHA;
  End;
End
```

可以看出系统中首先开始运行的是 Server 和 DF 这两个 Agent。伴随着每一个客户容器的加入,DF 将接受对每个容器的注册,同时第一个加入该系统的客户容器将激活 Server,由 Server 对该客户容器发送首次执行指令和运行参数,而其他加入的客户容器将等到上一个操作周期结束后才能真正得到运行的机会。这样既可以保证对于客户容器动态加入运行的需求,同时又能保证运行的结果能够及时、有效地被处理并参与到下一次的运行当中,对于得到优异的结果,将有明显的促进作用。

1.4　算法创新点

和以往的算法相比,该算法有以下几个创新点:

1)这是一个完全并行的系统,各个主机之间各自独立地运行自己的任务,主机与主机之间通过消息方式传递数据。

2）由于采用了 Java 作为开发工具，增强了系统的可移植性与跨平台性。

3）在运行的过程中，Server 由于采用了注册查询机制，可以在整个任务运行过程中接受新的客户容器动态加入。

4）对遗传算法模块进行了封装，使得可以用符合接口要求的其他遗传算法模块进行替换，实现系统的扩展。

2 试验结果分析

在本系统中所采用的遗传算法是简单遗传算法[5]。其中要解决的目标函数是求一元函数 $f(x) = x \cdot \sin(10\pi x) + 2$ 的最大值，优化变量区间为 $[-1,2]$。使用 Matlab 可以得到函数 $[-1,2]$ 之间的最大值应该比 3.85 稍大，同时可以画出函数图形，见图 2。如果设定求解精度到 6 位小数，由于区间长度为 3，必须将闭区间分为 3×10^6 等份。所以编码的二进制串长设定为 22 位。而将二进制串 $(b_{21}b_{20}\cdots b_0)_2$ 转化为区间 $[-1,2]$ 内对应的实数，则采用如下两个步骤：

1）将一个二进制串 $(b_{21}b_{20}\cdots b_0)_2$ 代表的二进制数转化为 10 进制数 x'。

2）利用公式 $x = -1.0 + x' \cdot \frac{2-(-1)}{2^{22}-1}$ 计算 x' 对应的区间 $[-1,2]$ 内的实数。

考虑到本例目标函数在定义域内均大于 0，而且是求函数最大值，所以直接应用目标函数作为适应度函数。在试验的过程中，我们设定种群大小为 30，最大进化代数为 150，交叉率和变异率分别设定为 0.8 和 0.05[5]。

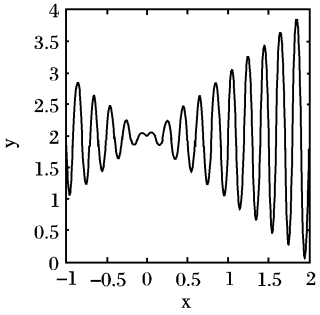

图 2 函数曲线

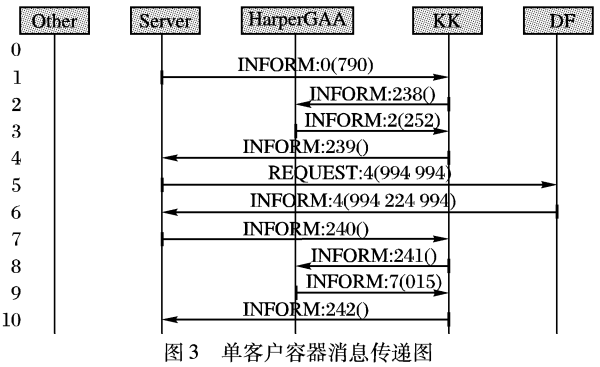

图 3 单客户容器消息传递图

对于系统的测试试验主要是在局域网中进行，主机和主机之间连接带宽为10M，主机采用Windows操作系统。试验过程中，采用了 IBM 笔记本和联想品牌机两台机器分别运行两个客户容器的方式进行，同时在 IBM 笔记本上运行主容器和 Server Agent。

特别是在运行双客户容器时可以看出，只有等到第一个容器运行完一次并返回结果后，在下一次的运行中，第二个容器才参与工作。而且由于两台机器的运行效率上的差异，所以 Server 只有等到运行中的容器都返回结果以后才发送下一次的运行指令。这样，系统的运行性能就将取决于系统中效率最低的容器。

根据所给出的算法，我们分成三个阶段进行了运行，运行过程中的消息传递见图 3 和图 4。图中 HarperGGA 和 HarperGGA1 代表了两个 CMHA，而 KK 和 KK1 代表的是两个 GAA。从图中可以看出，所有的消息都如设计时所考虑的一样正常运行。

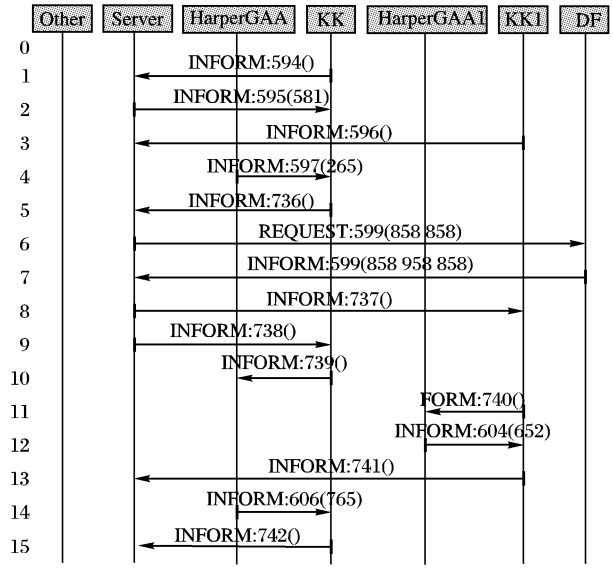

图 4 双客户容器消息传递图

我们将运行的结果进行比较，见表 1。

首先可以看出，单客户容器运行的时候，运行时间比单机运行遗传算法要长，而运行过程中未发现最大值的比率却下降了 30%。这和算法的构造是有关系的。由于单机运行的时候，没有各模块之间的消息的封装和拆分，所以，在运算速度上是有优势的。而且在服务器和客户容器算法中需要进行上一次最好结果的选择和插入，因此，速度上自然要比单机的慢，但是正是由于添加了该步骤，才使得各个代之间的遗传关系更加密切，结果也向好的方向发展，才会出现未发现最优结果的比率的下降。

其次，可以看出在增加一个客户容器的情况下，算法运行的效率得到了明显的提高，不仅仅比单客户容器快，而且比单机运行的速度也快，当然，由于算法的结构所决定，计算的执行时间比较明显地取决于算法实施过程中最慢的客户容器的速度。而在本次试验中，系统的运行在两个差别比较明显的计算机上，造成了一台机器的空等，而另外一台计算机的持续忙碌，这就直接造成对算法性能的提高没有预期中那么高。

表 1 运行结果对比表

算法	运行次数	每次运行代数	找到最大值	未发现最大值比率(%)	平均运行时间/ms
单机遗传算法	30	150	3.850274	30	1120574.63
单客户容器	30	150	3.850274	0	1380634.70
双客户容器	30	150	3.850274	0	912893.23

（下转第 1712 页）

并入一个序列即恢复了原来的非平稳性。把预测点加入原始序列作为历史数据使用,即可进一步预测后面的数据点。反复执行该过程即可完成一定时间长度的中、短期预测。

一个数据点的预报由两个要素构成:预测值和预测值置信区间。预测值总会存在一定的误差,预测值置信区间则给出预测值可能的取值范围。预测误差受随机因素的影响,可以用正态分布近似描述其分布规律。由此可得出预测值的95%置信区间:$(\hat{y}-1.96\times s/\sqrt{n},\ \hat{y}+1.96\times s/\sqrt{n})$,其中$\hat{y}$为预测值,$s$为残差序列的样本方差。

3 实际测试与分析

实测数据使用图2所示的,某通信企业下辖某市某区某网元的日话务量指标,数据采集自2004年11月19日起;使用前11个周期共77个真实点作为历史数据,连续预测4个周期共28点,与真实点进行对比分析。真实值与预测值数量比为11:4。预测分4次完成,每次并行预测7个点。预测中,第77个点以后的真实数据是不可见的,比如,预测第13周期的数据时,使用第12周期各点的预测值而不是真实值来作为历史数据。同时,不进行异常点处理,保留真实数据以验证抗干扰能力。

以(预测值-真实值)/预测值计算预测误差,如表1所示;并把预测结果汇总到Excel表中,与原始数据用折线图进行对比,如图5所示。

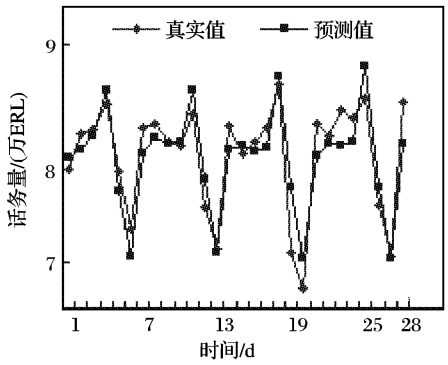

图5 预测值/实际值对比

由表1统计,预测误差绝对值低于1%有7个点,占25%;低于2%有13个点,占46.4%;低于4%有26个点,占93%;误差取绝对值,最小为0.191%,平均仅为2.3%,预测值是很准确的。真实数据均落入了预测值的95%置信区间内,如图6所示。

如图2所表明的,历史数据点总体呈现较大的无规律波动,并且历史数据最后3个周期出现明显的数据异常,预测时间段紧随其后。为验证模型对异常点的处理能力,模型没有进行异常剔除,保留真实数据进行预测。结果表明,预测值没有多大误差,这说明该预测方法具有较强的抗干扰能力。

表1 预测值误差分析

周期	预测值/ERL	预测误差	周期	预测值/ERL	预测误差
1	79593.00897	0.013996	1	85779.62178	0.025818
2	80331.76377	-0.019100	2	77554.31168	0.032153
3	81615.32560	-0.007110	3	70958.97879	-0.003950
4	85736.47395	0.013004	4	80453.03483	-0.027070
5	76457.75544	-0.023880	5	80774.55420	0.010344
6	70472.62512	-0.038210	6	80218.31171	-0.010270
7	80055.96888	-0.027970	7	80507.41675	-0.023330
1	81384.41753	-0.017160	1	87162.89755	0.008408
2	80982.35157	-0.002360	2	76854.32296	0.077545
3	81101.27846	0.004037	3	70357.61098	0.039677
4	79807.94395	-0.036390	4	80989.38923	-0.009030
5	80674.06953	-0.040100	5	81093.59298	-0.026760
6	87977.27932	0.032968	6	76919.28681	0.022580
7	70369.11100	-0.001910	7	80962.68265	-0.045980

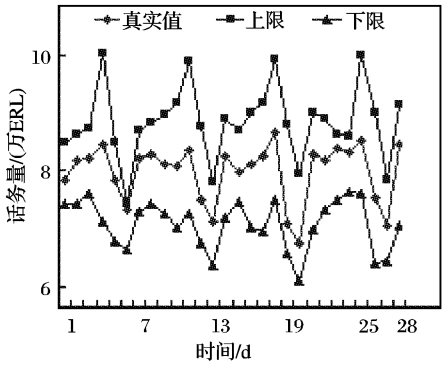

图6 真实值与预测值的95%置信区间

参考文献:

[1] BURGES JC. A tutorial on support vector machines for pattern recognition[J]. Data Mining and Knowledge Discovery, 1998,2(2): 121-167.

[2] PLATT JC. Sequential minimal optimization: a fast algorithm for training support vector machines[A]. Advances in Kernel Methods-Support Vector Learning[C]. MIT Press, 1999.185-208.

[3] VAPNIK V. Statistical Learning Theory[M]. John Wiley and Sons Inc, 1998.

[4] (美)PANDIT SM,吴宪民. 时间序列及系统分析与应用[M]. 李昌琪,荣国俊,译. 北京:机械工业出版社, 1988.

[5] 张波. 人工神经网络在流量预测中的运用[J]. 水道港口, 2005, 26(2): 80-82.

[6] 柳进,于继来,唐降龙. 基于数据挖掘的电网高峰负荷预测系统[J]. 计算机工程, 2005, 31(1): 10-11.

[7] 邹柏贤 刘强. 基于ARMA模型的网络流量预测[J]. 计算机研究与发展, 2002, 39(12): 1645-1651.

(上接第1708页)

3 结语

本文在对传统遗传算法分析的基础上,利用JADE进行了分布式遗传算法的研究,提出了一种基于多Agent协同的并行遗传算法结构。该结构由若干计算单元组成,每个计算单元实际上就是一个运行简单遗传算法的独立的计算Agent。通过试验可以明显感到,分布式运行效率较高,通过主从式运行机制,简化了各部分之间的通讯环节,提高了运行效率,加快了实现速度。

参考文献:

[1] 江瑞, 罗予频, 胡东成, 等. 一种基于多Agent协同的准并行遗传算法[J]. 电子学报, 2002, 30(10), 1490-1491.

[2] BELLIFEMINE F, CAIRE G, TRUCCO T, *et al*. JADE ADMINISTRATOR'S GUIDE [EB/OL]. http://jade.cselt.it/, 2005-03.

[3] BELLIFEMINE F, CAIRE G, TRUCCO T, *et al*. JADE PROGRAMMER'S GUIDE[EB/OL]. http://jade.cselt.it/, 2005-03.

[4] CAIRE G. JADE PROGRAMMING FOR BEGINNERS[EB/OL]. http://jade.cselt.it/, 2003-11.

[5] 王小平,曹立明. 遗传算法——理论、应用与软件实现[M]. 西安:西安交通大学出版社, 2002.18-50.