基于分布式人工免疫算法的数值优化

戚玉涛[1,2],刘 芳[1,2],焦李成[2]
(1. 西安电子科技大学计算机学院,陕西西安 710071;
2. 西安电子科技大学智能信息处理研究所和智能感知与图像理解教育部重点实验室,陕西西安 710071)

摘 要: 本文提出了一种分布式的人工免疫系统模型——塔式主从模型(TMSM),并基于此模型设计了一种用于解决数值优化问题的分布式免疫记忆克隆选择算法(DIMCSA). 借助 Markov 模型,文中证明了 DIMCSA 的收敛性. 为了摆脱网络连接状态对算法性能的影响,客观地衡量分布式人工免疫优化算法的性能,本文设计了多线程虚拟并行计算仿真系统,并分别考虑算法搜索时间和网络通信时间,给出了一种新的比较分布式随机搜索算法性能的指标. 实验结果表明,DIMCSA 能够用较少的计算代价和通信代价获得更高质量的解,适合解决大规模的复杂优化问题.

关键词: 分布式人工免疫模型; 数值优化; 克隆选择; Markov 链
中图分类号: TP183 **文献标识码**: A **文章编号**: 0372-2112 (2009) 07-1554-08

A Distributed Artificial Immune Algorithm for Numerical Optimization

QI Yu-tao[1,2], LIU Fang[1,2], JIAO Li-cheng[2]
(1. *School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi 710071, China;* 2. *Institute of Intelligent Information Processing and Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an, Shaanxi 710071, China*)

Abstract: This paper proposes a distributed model termed as Tower-like Master-Slave Model (TMSM) for the artificial immune systems. Based on TMSM, a distributed immune memory clonal selection algorithm (DIMCSA) is put forward for solving numerical optimization problem. Using the theorem of Markov chain, we have proved the convergence of DIMCSA. In order to get away from the influence of network conditions and get a veracious estimation on the DIMCSA's efficiency, Multi-thread simulative parallel computing system (MSPCS) is designed here and a novel performing index in which the searching time and network communication time are considered respectively is also proposed for distributed stochastic searching approaches. Experimental results indicate that DIMCSA can achieve better solutions with less computing and fewer communications, and it is capable of solving massive and complicated optimization problems.

Key words: distributed artificial immune model; numerical optimization; clonal selection; markov chain

1 引言

生物免疫系统是一个高度并行化的分布式自适应系统[1],其高度的自适应、自学习、噪声忍耐等特点正吸引着越来越多人工智能领域学者的目光. 他们借鉴生物免疫系统模型和机理,构造出高性能的、自组织的、鲁棒性好的人工智能系统,这就是人工免疫系统(Artificial Immune System, AIS). AIS 已经称为人工智能研究领域的新热点[2].

1958 年 Burnet 等提出的免疫系统克隆选择学说[3]是生物免疫系统最著名的模型之一. 在免疫克隆选择学说的启发下,人工智能领域出现了基于抗体种群进化的克隆选择算法. L. N. De Castro 构造了第一个克隆选择算法,并成功地用于解决模式识别、数值优化和组合优化问题[4]. Kim 等人提出了一种动态克隆选择算法,并用于解决连续变化环境中的异常探测问题[5]. 焦李成等人在前人工作的基础上提出了免疫多克隆策略[6],取得了良好的效果. 和进化算法一样,克隆选择算法也是通过编码来实现与问题无关的搜索. 问题解的编码称之为抗体,问题解的优劣用抗体的亲合度来描述. 免疫克隆选择算法的流程是从一个初始的抗体种群出发,通过克隆操作和免疫基因操作,并以亲合度的高低为标准对种群进行迭代进化,直到找到满意的解为止.

收敛速度和种群多样性之间的矛盾是所有进化类随机搜索算法都必须面对的. 为加快遗传算法的收敛速度,又不希望以牺牲种群多样性为代价,研究者开始研究遗传算法的并行实现[7]. 受并行遗传算法的启发,结合人工免疫系统自身的特点,本文提出了一种人工免

收稿日期:2008-01-17;修回日期:2009-02-23
基金项目:国家 863 高技术研究发展计划(No. 2006AA01Z107);国家自然科学基金(No. 60703107, No. 60703108)

疫系统的分布式模型——塔式主从模型(Tower-like Master-Slave Model,TMSM).在 TMSM 的基础上,构造了用于解决数值优化问题的分布式免疫记忆克隆选择算法(Distributed Immune Memory Clonal Selection Algorithm,DIMCSA).仿真结果表明,DIMCSA 适合求解大规模的复杂优化问题.

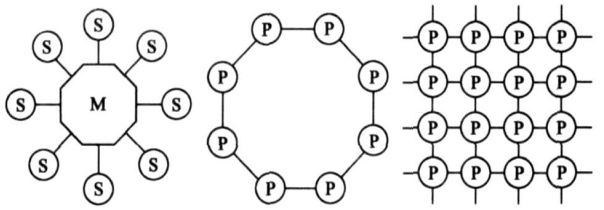
2　分布式人工免疫系统模型

目前关于人工免疫系统并行化模型的研究国内外还不多见.然而并行遗传算法的研究已经取得了一定的进展.现有的遗传算法并行模型主要有:主从式模型,粗粒度模型和细粒度模型[7].

2.1　遗传算法并行模型

主从式并行模型,结构如图 1(*a*)所示.该模型是一种单种群进化模型,主进程 M 执行种群的进化操作算子,从进程 S 分别计算每个染色体的适应度.在主从式模型中主进程和从进程交替工作,且任何一个从进程的滞后和通信故障都会导致整个算法的延误,因此该模型的效率不高,稳定性也较差.

粗粒度并行模型,结构如图 1(*b*)所示.该模型是基于多种群进化的模型,各个子种群进行独立的进化,子种群间以一定的时间间隔进行个体的迁移.粗粒度模型引起了研究者的广泛关注,目前已经出现了大量关于子种群拓扑结构[9],迁移策略[10],子种群分工策略[11]等方面的研究工作.



(*a*)主从式模型　(*b*)粗粒度模型　(*c*)细粒度模型

图1　遗传算法并行模型

细粒度并行模型是基于单个种群进化的模型,种群中个体分布在图 1(*c*)所示的网格上,且规定网格上的个体只能和与其临近的个体进行竞争和交叉.随着研究的深入,又出现了立方体、超立方体网格[12],智能网格[13]等结构的细粒度并行模型.该模型比粗粒度模型更有利于保护局部子种群自身特质,但是个体间的通信过频,通信开销巨大.

2.2　分布式人工免疫系统模型

粗粒度并行模型是一种适合分布式实现的并行模型.为了引入免疫系统特性,本文在粗粒度模型基础上构造了适合人工免疫算法的塔式主从式模型(TMSM),如图 2 所示.

定义 1　TMSM 由一个主种群 M 和若干个从种群 S 按图 2 中的方式组织而成.

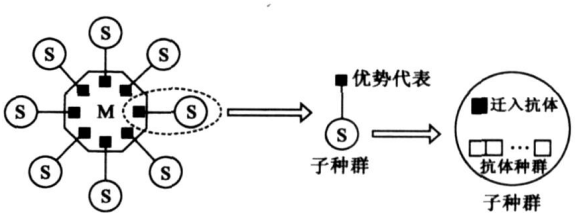


图2　塔式主从式模型

定义 2　从种群 S,称为子种群,由一个抗体种群和一个迁入抗体构成.每个子种群和记忆种群中的一个抗体构成一一映射,与子种群构成映射的记忆抗体称为子种群的优势代表.子种群和对子种群的操作构成子种群进程.

定义 3　主种群 M,称为记忆种群,由各个子种群对应的优势代表抗体构成.记忆种群进程除了包括记忆种群和对记忆种群的操作之外,还完成对子种群进程之间的个体迁移调度.

2.3　塔式主从式模型分析

TMSM 体现了免疫系统特性.模型将一个子种群和记忆抗体建立一一映射,子种群分工策略使各子种群对不同区域的解空间进行搜索,子种群和记忆抗体的对应关系又将子种群的多样性延伸到了记忆种群.这种分布式搜索和分布式记忆机制有助于保持抗体种群的多样性.TMSM 的各计算节点之间既相互独立又相互联系,某个子种群进程的故障不会导致整个算法的中止.TMSM 是一种粗粒度的并行,通信大代价小.模型把抗体之间大量的信息交互限制在计算节点内部,节省了通信开销.

3　解决数值优化问题的分布式免疫记忆克隆选择算法

最优化问题由目标和约束条件两个部分构成:

$$\text{Minimize } f(x) = f(x_1, x_2, \cdots, x_n) \tag{1}$$

$$\text{Subject } x = (x_1, x_2, \cdots, x_n) \in S \subset X \tag{2}$$

$f(X)$ 为数值优化问题的目标函数,满足所有约束条件的解空间 S 称为可行域,$S \subseteq R^n$ 表示边界为 $\underline{x}_i \leq x_i \leq \overline{x}_i$,$i = 1, 2, \cdots, n$ 的 n 维搜索空间,即:

$$S = [\underline{x}, \overline{x}] \quad , \underline{x} = (\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_n) \quad , \overline{x} = (\overline{x}_1, \overline{x}_2, \cdots, \overline{x}_n) \tag{3}$$

可行域中的解称为可行解.在可行域中使目标函数最小的解为最优解.对于最大化问题,可将目标函数乘以负 1,转化为最小化问题求解.对于有复杂约束的数值优化问题,可以通过拉格朗日乘子法或者惩罚函数的方法将有约束的优化问题转化为无约束优化问题来处理.

3.1　抗体编码及亲合度定义

本文采用了实数编码方式,用 0 到 1 之间的实数串

表示一个抗体. 对于一个 m 维的数值优化问题,抗体的编码长度为 m. 记抗体空间中的任意一个抗体为: $A = g_1, g_2, \cdots, g_m, g_i \in [0,1]$, $i = 1,2,\cdots,m$,其解码后对应的函数变量记为: $X = (x_1, x_2, \cdots, x_m)$. 抗体 A 称为变量 X 的编码,记为: $A = e(X)$;反之,变量 X 称为抗体 A 的解码,记为: $X = e^{-1}(A)$. 抗体的解码过程如式(4):

$$x_i = \underline{x_i} + (\overline{x_i} - \underline{x_i}) \times g_i, i = 1,2,\cdots,m \quad (4)$$

抗体的亲合度必须是一个正实数,并且亲合度的值越大的抗体,其对应的解的质量越好. 目标函数 $f(X)$ 的值无法保证是常正或者常负的,因此,不能用 $f(X)$ 直接构造抗体的亲合度函数. 在此,我们引入了一个常负的中间函数 $g(X)$,满足以下条件:对任意的变量 X 都有 $g(X) \leq 0$,且 $g(X)$ 和 $f(X)$ 是一致的,即,对任意的两变量 $X_1, X_2 \in S$, 如果 $g(X_1) > g(X_2)$, 那么 $f(X_1) > f(X_2)$. 引入了 $g(X)$,原来的优化问题就转化为: $\min\{g(e^{-1}(A)) : A \in I\}$,其中 I 是抗体空间. 抗体 A 与抗原之间的亲合度定义为:

$$aff(A) = -g(e^{-1}(A)) \quad (5)$$

n 个抗体构成一个抗体种群,记为 $A = \{A_1, A_2, \cdots, A_n\}$. 规模为 n 的抗体种群构成的抗体种群空间可以表示为:

$$I^n = \{A : A = (A_1, A_2, \cdots, A_n), A_k \in I, 1 \leq k \leq n\} \quad (6)$$

3.2 分布式免疫记忆克隆选择算法流程

DIMCSA 是基于 TMSM 且采用了免疫克隆选择算法流程[6]的多种群异步分布式算法. 算法分为子种群进程和记忆种群进程两个部分.

DIMCSA 的计算任务由记忆种群发起,并由记忆种群判断算法是否停止. 整个算法流程的伪码表示如下:

```
Begin
    设定算法终止条件,抗体编码长度 m,给定子种群数 N,子种群规模 S,子种群克隆规模 C_S,记忆种群克隆规模 C_m,子种群变异概率 P_S,子种群重组概率 P_C 和迁移间隔 G;
    停机信号量 STOP = FALSE;
    启动记忆种群进程;
    记忆种群进程依次启动所有的子种群进程,并给子种群进程编号,同时以某种子种群分工策略给各个子种群分配搜索中心;
    每个子种群进程根据分配到的搜索中心初始化各自的子种群,并计算初始种子群抗体亲合度;
    每个子种群进程用子种群最优抗体初始化与之对应的优势代表,产生初始记忆种群;
    While (NOT STOP)
    记忆种群和每个子种群同时迭代进化;
End
```

停机信号量 STOP 是一个全局信号量,由记忆种群进程控制,一旦 STOP 的取值为 TRUE,记忆种群进程会通知所有的子种群进程停止搜索. 迁移间隔 G 代表了各个子种群进程进行个体迁移操作的间隔为 G 次迭代. 记忆种群进程会在每次迭代结束的时候判断算法终止条件是否满足,如果满足则将停机信号量置为 TRUE,算法停止.

3.3 子种群的迭代进化

各个子种群进程分别对各自的抗体种群进行免疫多克隆[6]的进化迭代操作. 第 i 个($i = 1,2,\cdots,N$) 子种群进程操作步骤的伪码表示如下:

```
Begin
    While (NOT STOP) {
    for (int count = 0; count < G. ;count + +){
    对该子种群抗体进行克隆操作;
    对克隆后的抗体进行免疫基因操作;
    对免疫基因操作后的抗体进行克隆选择操作;
    }
    执行迁出抗体提交操作;
    }
End
```

3.3.1 克隆操作

对第 i 个子种群经过第 g 次迭代后得到的种群 $SP_i(g)$, $i = 1,2,\cdots,N$中的任意一个抗体 $a_j(g)$ 进行规模为 q_j 克隆的克隆操作,定义为:

$$A_j = I_j \times a_j(g), j = 1,2,\cdots,S \quad (7)$$

其中 I_j 为元素为 1 的 q_j 维行向量. q_j 是子种群克隆规模, q_j 的大小与 $a_j(g)$ 的亲合度成正比.

3.3.2 免疫基因操作

算法的免疫基因操作包括重组操作[6]和克隆变异操. 抗体 $a_j(g)$ 克隆后的抗体组 A_j 中的第 k 个抗体记为 $A_j[k]$, $k = 1,2,\cdots q_j$. 记该抗体经过重组操作后的抗体记为 $A_j\,'[k]$. 那么:

$$A_j\,'[k] = \begin{cases} A_j[k], & \text{if } k = 1 \\ CR(A_j[k], a_M), & \text{if } k = 2 \\ CR(A_j[k], a_r(g)), & \text{otherwise} \end{cases} \quad (8)$$

其中 a_M 为当前的迁入抗体, $a_r(g)$ 是子种群 $SP_i(g)$ 中不同于 $a_j(g)$ 的任意抗体, $r \neq j$ 且 $r = 1,2,\cdots,N$. $CR(a_1(g), a_2(g))$ 表示抗体 $a_1(g)$ 和抗体 $a_2(g)$ 以某种方式进行重组得到的两个新抗体中的任意一个.

变异操作是对重组操作后得到的抗体组 A_j 中的每一个抗体进行变异概率为 P_S 的按位高斯变异,即随机改变抗体的 $m \times p_S$ 个抗体位, m 是抗体编码长度. 变异操作后得到新的抗体组 A_j''.

3.3.3 克隆选择操作

克隆选择操作从 A_j'' 和 $a_j(g)$ 中选出一个抗体,记为 $a_j(g+1)$,加入下一代子种群中. 若 $\tilde{a}_j$ 代表 A_j'' 中亲

合度最大的抗体,$a_j(g)$ 代表 $SP_i(g)$ 中 A_j 的克隆父代抗体,那么 $a_j(g+1)$ 可以通过式(9)得到:

$$a_j(g+1)=\begin{cases}\tilde{a}_j, & \text{if } U(0,1)\le p_j\\ a_j(g), & \text{otherwise}\end{cases} \tag{9}$$

其中 $U(0,1)$ 表示 0 到 1 之间的随机实数,p_j 是一个概率,p_j 的取值规则如下:

如果 $aff(a_j(g))<aff(\tilde{a}_j)$,那么:

$$p_j=1 \tag{10}$$

如果 $aff(a_j(g))\ge aff(\tilde{a}_j)$ 且 $a_j(g)$ 不是当前种群中的最优抗体,那么:

$$p_j=\exp\left(-\frac{aff(a_j(g))-aff(\tilde{a}_j)}{\alpha}\right) \tag{11}$$

如果 $aff(a_j(g))\ge aff(\tilde{a}_j)$ 且 $a_j(g)$ 是当前种群中的最优抗体,那么:

$$p_j=0 \tag{12}$$

在式(10)～(12)中,$aff(a_j(g))$ 代表抗体 $a_j(g)$ 的亲合度,$\alpha>0$ 是一个与抗体种群多样性有关的值,一般多样性越好,α 取值越大,反之越小.

3.3.4　迁出抗体提交操作

子种群抗体经过 G 次迭代后,子种群进程在当前子种群中以一定的策略选择一个抗体提交给记忆种群进程的操作称为迁出抗体提交操作. 本文算法将当前最优抗体作为迁出抗体. 子种群进程将迁出抗体发送给记忆种群进程,记忆种群进程收到后将迁出抗体信息加入到迁移调度任务队列.

3.4　记忆种群的迭代进化

记忆种群进程操作步骤的伪码表示如下:

```
Begin
    将迁移调度任务队列 Q 初始化为空队列;
    While (NOT STOP) {
    While (Q 不为空)
    调度位于队首的抗体迁移任务;
    对记忆种群中前 T%个最优抗体进行记忆
    成熟操作;
    if (算法终止条件满足) {
    STOP = TRUE;
    向所有子种群进程发送 STOP = TRUE;
    }
    }
End
```

3.4.1　抗体迁移任务调度

抗体迁移任务调度操作完成两个方面的任务:用子种群的迁出抗体更新其优势代表;给子种群分配一个迁入抗体并发送给子种群进程.

记忆种群进程收到子种群的迁出抗体信息后,会将该信息加入到迁移调度任务队列的队尾. 迁移调度任务队列中保存的数据结构包括发起调度任务的子种群编号 $id(id=1,2,\cdots,N)$,子种群提交的迁出抗体 $\hat{a}_{id}$ 和迁出抗体的亲合度 $aff(\hat{a}_{id})$. 记忆种群经过第 g 次迭代后得到的种群记为 $MP(g)$,$MP(g)$ 中的第 i 个抗体是编号为 i 的子种群对应的优势代表,记为 $MP[i]$,$i=1,2,\cdots,N$.

在更新优势代表的操作如式(13):

$$MP[id]=\begin{cases}\hat{a}_{id}, & \text{if } aff(\hat{a}_{id})\ge aff(MP[id])\\ MP[id], & \text{otherwise}\end{cases} \tag{13}$$

在给编号为 id 的子种群指派迁入抗体的操作中,记忆种群进程随机产生一个自然数 j,满足 $j\neq id$,$j=1,2,\cdots,N$,然后将 $MP[j]$ 作为编号为 id 的子种群的迁入抗体发送给这个子种群进程.

3.4.2　记忆成熟操作

记忆种群进程在记录抗体子种群优势抗体的同时,也在频繁自我进化. 记忆成熟算子对记忆种群中最优的 $T\%$个抗体进行单克隆选择进化,即克隆、变异和选择的迭代过程. T 的值反映了记忆种群进行记忆成熟操作抗体的比例,本文取为 10.

克隆操作和子种群的克隆操作相同,对记忆种群中的每个抗体 $MP[i]$,$i=1,2,\cdots,N$ 进行 q_i 克隆,得到 q_i 维的抗体组 $MP[i]$. 记忆抗体的变异采取方差 σ 随迭代次数 k 递减的高斯变异方式. 变异操作得到新的抗体组记为 $MP'[i]$. 克隆选择操作从 $MP'[i]$ 中选出一个最优的抗体作为下一代记忆抗体替代克隆父代抗体. 与子种群的克隆选择操作不同,这里的克隆选择是一个择优操作.

4　算法收敛性证明

子种群进程的迁出抗体提交操作总是将子种群最优抗体提交给记忆种群进程,且记忆种群进程只有在子种群提交的优势代表优于上一代的优势代表时才进行更新. 因此,子种群的迁出抗体提交操作不会使记忆种群退化. 另外记忆种群进程的记忆成熟操作是完全择优的进化,也保证了记忆种群是不退化的. 因此,记忆种群反映了整个系统的进化过程,如果能证明记忆单元中的抗体收敛到最优解对应的抗体,那么整个算法就是收敛的. 下面通过分析记忆种群的收敛性来说明 DIMCSA 以概率 1 收敛. 由于记忆种群中的每个记忆抗体只与该抗体的前一代抗体相关,因此,可以用马尔可夫链(Markov Chain)描述每个记忆抗体的进化过程.

集合 I 为抗体空间,则记忆种群空间可以记为:

$$I_{MP}=\{MP\mid MP[i]\in I,i=1,2,\cdots,N\} \tag{14}$$

记优化问题最优解集合为:

$$B^{*}=\{a\mid a\in I, aff(a)=f^{*}=\max(aff(a')), a'\in I\} \tag{15}$$

对于记忆抗体种群 MP:

$$\vartheta(MP)=\mid MP\cap B^{*}\mid \tag{16}$$

表示记忆抗体种群 MP 中包含对应着最优解的抗体个数,是一个大于等于零的整数.

定义 4 如果对于记忆种群的任意初始状态 MP^{0},经过均 k 次迭代后得到的 $MP^{(k)}$ 均有:

$$\lim_{k\to\infty}P\{\vartheta(MP^{(k)})\geq 1\mid MP^{(0)}=MP^{0}\}=1 \tag{17}$$

则称,算法以概率 1 收敛到最优种群集.

定理 1 记忆种群中的每个抗体经过一次迭代进化成最优抗体的概率大于零.

证明 记最优抗体种群中的任意一个抗体为 A,$\overline{\mathrm{A}}$是每个基因位都和 A 不同的抗体. 那么对于记忆种群中的任意一个抗体 $MP[i]$, $i=1,2,\cdots,N$,其变异到 A 的概率 δ 都大于等于$\overline{\mathrm{A}}$变异到 A 的概率$\overline{\delta}$. 算法中记忆成熟操作采取方差 σ 随迭代次数 k 递减的高斯变异方式,则对于抗体$\overline{\mathrm{A}}$的第 i 个抗体基因位$\overline{\mathrm{A}_i}$变异到与 A 的第 i 个抗体基因位 A_i 相同的概率为

$$p_i=\left(\sqrt{2\pi}\sigma\right)^{-1}\exp\left(-\left(\mathrm{A}_i-\overline{\mathrm{A}_i}\right)^{2}/2\sigma^{2}\right)>0 \tag{18}$$

因此$\overline{\mathrm{A}}$变异到 A 的概率:

$$\overline{\delta}=\prod_{j=1}^{m}p_j>0 \tag{19}$$

其中 m 是抗体编码长度. 故而记忆单元中的每个抗体经过一次迭代进化成最优抗体的概率 $\delta\geq\overline{\delta}>0$. 定理 1 证毕.

定理 1 说明了,DIMCSA 采取的免疫单克隆的记忆成熟操作使得记忆种群中的每个抗体都有可能经过一次迭代进化成最优抗体.

定理 2 DIMCSA 是概率 1 收敛的.

证明 记忆种群的任意初始状态 MP^{0},经过 k 次迭代没有找到全局最优解的概率记为:

$$\begin{aligned}P_0(k)&=P\{\vartheta(MP^{(k)})=0\}\\&=P\{MP^{(k)}\cap B^{*}=\varnothing\}\end{aligned} \tag{20}$$

那么:

$$\begin{aligned}P_0(k+1)&=P\{\vartheta(MP^{(k+1)})=0\}\\&=P\{MP^{(k+1)}\cap B^{*}=\varnothing\}\end{aligned} \tag{21}$$

根据贝叶斯条件概率公式有:

$$\begin{aligned}P_0(k+1)&=P\{\vartheta(MP^{(k+1)})=0\}\\&=P\{\vartheta(MP^{(k+1)})=0\mid\vartheta(MP^{(k)})\neq 0\}\\&\quad\times P\{\vartheta(MP^{(k)})\neq 0\}\\&\quad+P\{\vartheta(MP^{(k+1)})=0\mid\vartheta(MP^{(k)})=0\}\\&\quad\times P\{\vartheta(MP^{(k)})=0\}\end{aligned} \tag{22}$$

由于记忆种群中的抗体是不退化的,所以

$$P\{\vartheta(MP^{(k+1)})=0\mid\vartheta(MP^{(k)})\neq 0\}=0 \tag{23}$$

将式(23)带入式(22)可得:

$$\begin{aligned}P_0(k+1)&=P\{\vartheta(MP^{(k+1)})=0\mid\vartheta(MP^{(k)})=0\}\\&\quad\times P\{\vartheta(MP^{(k)})=0\}\\&=P\{\vartheta(MP^{(k+1)})=0\mid\vartheta(MP^{(k)})=0\}\\&\quad\times P_0(k)\end{aligned} \tag{24}$$

记忆种群中的抗体获得全局最佳抗体有两种方式,一种是通过记忆种群的免疫操作获得,设此种方式的概率为:

$$P_I=\{P_I^{(1)},P_I^{(2)},\cdots\} \tag{25}$$

另一种是通过抗体子种群提交的迁出抗体获得,设此种方式的概率为

$$P_{II}=\{P_{II}^{(1)},P_{II}^{(2)},\cdots\} \tag{26}$$

因此:

$$\begin{aligned}&P\{\vartheta(MP^{(k+1)})\geq 1\mid\vartheta(MP^{(k)})=0\}\\&=1-\left(1-P_I^{(k+1)}\right)\left(1-P_{II}^{(k+1)}\right)\end{aligned} \tag{27}$$

由定理 1 我们可以得到, $P_I^{(k)}>0$. 记 $\zeta=\min_k P_I^{(k)},;$ $k=1,2,\cdots$则由式(27)可知:

$$P\{\vartheta(MP^{(k+1)})\geq 1\mid\vartheta(MP^{(k)})=0\}>P_I^{(k+1)}\geq\zeta>0 \tag{28}$$

因此有:

$$\begin{aligned}&P\{\vartheta(MP^{(k+1)})=0\mid\vartheta(MP^{(k)})=0\}\\&=1-P\{\vartheta(MP^{(k+1)})\neq 0\mid\vartheta(MP^{(k)})=0\}\\&=1-P\{\vartheta(MP^{(k+1)})\geq 1\mid\vartheta(MP^{(k)})=0\}\\&\leq 1-\zeta<1\end{aligned} \tag{29}$$

由式(24)可知:

$$\begin{aligned}0\leq P_0(k+1)&\leq(1-\zeta)\times P_0(k)\\&\leq(1-\zeta)^{2}\times P_0(k-1)\cdots\leq(1-\zeta)^{k+1}\times P_0(0)\end{aligned} \tag{30}$$

由于$\lim_{k\to\infty}(1-\zeta)^{k+1}=0$,$0\leq P_0(0)\leq 1$,所以:

$$0\leq\lim_{k\to\infty}p_0(k)\leq\lim_{k\to\infty}(1-\zeta)^{k+1}P_0(0)=0 \tag{31}$$

所以:

$$\lim_{k\to\infty}p_0(k)=0 \tag{32}$$

对于记忆抗体的任意初始状态 MP^{0},经过 k 次迭代后找到全局最优解的概率记为:

$$P_1(k)=P\{\vartheta(MP^{(k)})=1\}=P\{MP^{(k)}\cap B^{*}\neq\varnothing\} \tag{33}$$

由式(22)可知:

$$P_1(k)=1-P_0(k) \tag{34}$$

故而:

$$\begin{aligned}\lim_{k\to\infty}P_1(k)&=\lim_{k\to\infty}P\{\vartheta(MP^{(k)})\geq 1\}\\&=1-\lim_{k\to\infty}P\{\vartheta(MP^{(k)})=0\}\\&=1-\lim_{k\to\infty}P_0(k)=1\end{aligned} \tag{35}$$

因此,对于记忆抗体的任意初始状态 MP^{0} 有:

$$\lim_{k\to\infty}P\{\vartheta(MP^{(k)})\geq 1\mid MP^{(0)}=MP^{0}\}=1 \tag{36}$$

由定义 4 可知,DIMCSA 以概率 1 收敛到最优种群集. 定理 2 证毕.

5　仿真实验

本文采用了以下十个标准测试函数:

$$f_1(\mathrm{x}) = \sum_{i=1}^{N} x_i^2;\ \mathbf{S} = [-100,100]^n;\ f(\mathrm{x}^*) = 0 \tag{37}$$

$$f_2(\mathrm{x}) = \sum_{i=1}^{N} |x_i| + \prod_{i=1}^{N} x_i|;\ \mathbf{S} = [-10,10]^n;\ f(\mathrm{x}^*) = 0 \tag{38}$$

$$f_3(\mathrm{x}) = \sum_{i=1}^{N} \left(\sum_{j=1}^{i} x_j\right)^2;\ \mathbf{S} = [-100,100]^n;\ f(\mathrm{x}^*) = 0 \tag{39}$$

$$f_4(\mathrm{x}) = \sum_{i=1}^{n} \left(|x_i + 0.5|\right)^2;\ \mathbf{S} = [-100,100]^n;\ f(\mathrm{x}^*) = 0 \tag{40}$$

$$f_5(\mathrm{x}) = \sum_{i=1}^{N} i x_i^4 + random[0,1);\ \mathbf{S} = [-1.28,1.28]^n;\ f(\mathrm{x}^*) = 0 \tag{41}$$

$$f_6(\mathrm{x}) = \sum_{i=1}^{N} \left(-x_i \sin\left(\sqrt{|x_i|}\right)\right);\ \mathbf{S} = [-500,500]^n;\ f(\mathrm{x}^*) = -418.983 \times n \tag{42}$$

$$f_7(\mathrm{x}) = \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i) + 10);\ \mathbf{S} = [-5.12,5.12]^n;\ f(\mathrm{x}^*) = 0 \tag{43}$$

$$f_8(\mathrm{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right) - \exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right) + 20 + e;\ \mathbf{S} = [-30,30]^n;\ f(\mathrm{x}^*) = 0 \tag{44}$$

$$f_9(\mathrm{x}) = \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1;\ \mathbf{S} = [-600,600]^n;\ f(\mathrm{x}^*) = 0 \tag{45}$$

$$f_{10}(\mathrm{x}) = \frac{1}{n}\sum_{i=1}^{n} \left(x_i^4 - 16x_i^2 + 5x_i\right);\ \mathbf{S} = [-5,5]^n;\ f(\mathrm{x}^*) = -78.33236 \tag{46}$$

f_1～f_3 是单峰函数,f_4 是阶梯函数. f_5 是有噪四次函数. f_6～f_{10} 是多峰函数,其局部最优值的个数随着问题维数的增长而增长.

5.1　多线程虚拟并行计算仿真环境

通常为了验证分布式算法的性能,人们将分布式算法在处理器阵列[9]上或局域网[11,14]上运行,用算法运行时间作为衡量算法性能的重要指标. 这种方法看似合理,但仿真结果和实验环境相关,实际上是有失客观性的. 分布式算法的运行时间由两个重要部分组成:算法计算时间和网络通信时间. 前者只取决于分布式算法本身,与网络延迟无关;后者不仅和分布式算法有关,还受到通信网络状态的影响. 网络开销大的算法受网络状态影响大,反之则较小. 现有的算法评价机制,都没有将两者分离开来考虑. 也就是说,当算法的运行环境有所改变的时候,算法的优劣也可能随之改变.

本文采用多线程技术来模拟分布式的计算资源,构造了一种多线程虚拟并行计算仿真环境(Multi-thread Simulative Parallel Computing System, MSPCS),并基于 MSPCS 设计了衡量并行算法性能的指标. MSPCS 克服了基于处理器阵列和基于局域网的并行仿真系统的缺点,将算法计算时间和网络通信时间分开来考虑. 相应地,算法性能衡量标准也由两个部分组成,算法的使用者可以根据算法运行的网络环境选择合适的算法.

算法在执行过程中用亲合度评价计数器数组 AET 记录记忆种群和各个子种群平均抗体亲合度的次数. AET 规模为 $N+1$,其中 N 为子种群个数. $AET[0]$ 表示记忆种群线程评价抗体亲合度的次数,$AET[i]$,$i = 1, 2, \ldots, N$ 表示第 i 个子种群线程评价抗体亲合度的次数. 用计数器 CT 记录算法通信次数. 那么,算法的平均亲合度评价次数 θ_1 和平均通信次数 θ_2 可以分别由式(47)(48)得到:

$$\theta_1 = \frac{1}{N+1}\sum_{i=0}^{N} AET[i] \tag{47}$$

$$\theta_2 = \frac{1}{N+1} CT \tag{48}$$

算法总的运行时间可以表示为:

$$T = \theta_1 T_1 + \theta_2 T_2 \tag{49}$$

其中,T_1 是算法计算一次抗体亲合度需要的时间代价;T_2 是分布式算法在实际运行环境中计算节点之间进行一次信息传递需要的平均时间代价.

由式(49)可以看出,在决定算法运行时间的四个因素中,θ_1 和 θ_2 只与算法的性能有关,T_1 和 T_2 与问题的规模和网络状态有关. 因此,本文用 θ_1 和 θ_2 来衡量算法的搜索性能. 在实际应用中,使用者根据实际问题的规模和算法的具体运行环境估算出 T_1 和 T_2,进而决定使用哪种算法更优.

5.2　DIMCSA 搜索全局最优解的能力

为了验证 DIMCSA 搜索到全局最优解的能力,对十个不同类型的标准测试函数进行了优化实验,并将实验结果和分布式遗传算法(DGA)[15]进行了比较. 目标函数维数为 30,子种群数量为 10,规模为 2,迁移间隔为 1. 在 DIMCSA 中,子种群克隆规模为 15,变异概率为 $1/m$,m 为编码长度,重组概率为 1,记忆种群克隆规模为 10. 算法的停止条件设置为当前搜索到的最优解达到一定的精度或者迭代次数达到上限 10000 次. 表 1 中的数据是 50 次独立实验的平均结果.

从表 1 的实验结果可以看出,DIMCSA 用较小的计

算代价和通信代价找到了质量更高的解. DIMCSA 比 DGA 的收敛速度更快,精度更高.

表 1 DIMGSA 和 DGA 搜索到全局最优解能力比较

测试函数 f N	算法	平均函数评价次数	平均通信次数	平均迭代次数	平均最优值	全局最优值
f_1 30	DGA	1521.82	16720	836	2.81711×10^{-8}	0
	DIMCSA	646.128	1772	88.6	5.22336×10^{-9}	
f_2 30	DGA	1605.09	17636	881.8	3.83281×10^{-8}	0
	DIMCSA	640.364	1756	87.8	1.50296×10^{-8}	
f_3 30	DGA	18183.6	200000	10000	118.316	0
	DIMCSA	9009.09	24770	1238.5	9.14024×10^{-8}	
f_4 30	DGA	833.091	9144	457.2	0	0
	DIMCSA	526.182	1442	72.1	0	
f_5 30	DGA	7850.46	31395.8	5232.64	9.26937×10^{-3}	0
	DIMCSA	1944.36	5342	267.1	8.58572×10^{-4}	
f_6 30	DGA	3721.82	40920	2046	-12569.421	-12569.5
	DIMCSA	889.818	2442	122.1	-12569.493	
f_7 30	DGA	1193.09	13104	655.2	4.05878×10^{-8}	0
	DIMCSA	425.818	1166	58.3	4.17564×10^{-9}	
f_8 30	DGA	2017.91	14778	738.9	5.09939×10^{-8}	0
	DIMCSA	639.636	1754	87.7	2.78861×10^{-8}	
f_8 30	DGA	1550	17030	8515.5	-78.2916	-78.33236
	DIMCSA	312.364	854	42.7	-78.33201	
f_9 30	DGA	13235.5	145570	7278.5	0.053538	0
	DIMCSA	1235.45	2260	113	$2.68966\times^{-8}$	
f_{10} 30	DGA	1550	17030	851.5	-78.2916	-78.33236
	DIMCSA	312.364	854	42.7	-78.33201	

5.3 DIMCSA 解决大规模优化问题的能力

该组实验验证了 DIMCSA 求解大规模复杂问题的能力. f_6～f_9 是四个多峰函数,其局部最优值的个数随着问题维数的增长而增长. 其中,f_6 是 Schwefel 函数,在 50 维的情况下常见算法将很难快速求得正解; f_7 是 Rastrigin 函数,有很大的曲率,容易使搜索陷入局部最优; f_8 是 Ackley 函数,有大量的局部极小值,且局部极小值的数量随着维数的增加而增加; f_9 是 Griewangk 函数,其全局单峰结构受余弦函数的调制,具有非线性不可分特点.

表 2 DIMCSA 和 DGA 对于高维函数优化能力比较

维数	算法	f_6:Schwefel 函数	f_7:Rastrigin 函数	f_8:Ackley 函数	f_9:Griewangk 函数
100	DGA	17766.7 + 195414	14599.3 + 160572	4241.64 + 46638	4569.27 + 50242
	DIMCSA	6046.18 + 16622	3402.55 + 9352	1706.55 + 4688	2323.27 + 6384
200	DGA	97269.1 + 1069941	27101.8 + 298135	9251.2 + 101744	10344.7 + 113772
	DIMCSA	15798.2 + 43440	6180.73 + 16992	4009.09 + 11020	3824.36 + 10512
400	DGA	/	29552.7 + 283561	17954.5 + 197480	17968.7 + 197636
	DIMCSA	16073.1 + 44196	12350.9 + 33960	9715.27 + 26712	8822.18 + 24256
600	DGA	/	117644 + 1294062	43814.5 + 481940	28578.2 + 314341
	DIMCSA	42371.3 + 116516	25984.9 + 71456	17299.3 + 47568	12741.3 + 34924
800	DGA	/	166804 + 1834820	48730.9 + 536020	41641.8 + 458040
	DIMCSA	64721.8 + 177980	27097.1 + 74512	19107.3 + 52540	15056.4 + 41396
1000	DGA	/	/	/	/
	DIMCSA	71918.9 + 197772	36093.3 + 99249.9	27361.6 + 75237.2	24092.2 + 66235.3

表 2 比较了 DIMCSA 和 DGA 对高维函数的优化结果,表中的数据格式为"平均函数评价次数 + 平均通信次数". 算法参数设置与 5.2 中实验相同,终止条件设置为精度达到 0.01 或者迭代次数达到上限 10^5 次. 表中数据为 30 次独立实验的平均值.

从表 2 中的实验数据充分体现了 DIMCSA 在求解大规模多峰数值优化问题中的优越性. 表中的符号"/"表示算法不能够在 10^5 次迭代以内找到函数的最优解. 在对四个高维多峰函数的优化结果中,DIMCSA 在平均函数评价次数和平均通信次数两个指标上都优于

DGA. 尤其对于 Schwefel 函数,DGA 在 400 维的情况下已经很难找到全局最优解,而 DIMCSA 在 1000 维的情况下仍然能够找到全局最优. 因此,DIMCSA 对于解决大规模的复杂优化问题是十分有效的.

6　总结与展望

本文提出了一种分布式的人工免疫系统模型 TMSM,并在该模型的基础上构造了用于解决数值优化问题的分布式人工免疫算法 DIMCSA. TMSM 的主从分布式结构在模拟免疫系统分布式特性的同时,为系统引入了免疫记忆机制. 基于 TMSM 模型之上的 DIMCSA 是一种多种群的克隆选择算法适合在分布式的计算环境下运行.

借助 Markov 模型,本文证明了 DIMCSA 算法的收敛性. 通过对十个标准测试函数的实验结果也说明了 DIMCSA 的优越性. 文中分析了算法总的运行时间不能够作为衡量算法在工程应用中有效性的客观指标,提出了基于多线程技术的虚拟并行仿真环境,并通过计算各个子种群的平均函数评价次数和平均通信次数作为衡量算法性能的指标. 在具体的工程应用中,使用者可以通过具体问题的复杂度和算法运行的网络环境选择适合的算法. 在本文提出的指标体系下,DIMCSA 无论在平均函数评价次数还是平均通信次数上都比 DGA 有着明显的优势. 对高维和超高维多峰函数的优化结果表明,DIMCSA 适合解决大规模的复杂优化问题.

对于大规模的复杂优化问题,集中式的随机搜索算法需要花费大量的计算时间搜索问题的最优解. DIMCSA 为人工免疫算法提供了一种分布式的解决方案,提高了算法的运行速度,这势必为人工免疫算法开辟更为广阔的应用空间.

参考文献:

[1] D Dasgupta. An Overview of Artificial Immune System and Their Applications. In Artificial Immune System and Their Applications[M]. Berlin:Springer-Verlag,1999,3 - 18.

[2] D Dasgupta,S Forrest. Artificial immune systems in industrial applications[A]. Proceedings of Intelligent Processing and Manufacturing of Materials '99[C]. Honolulu:IEEE press,1999,257 - 267.

[3] F M Burnet. Clonal selection and after[A]. Theoretical Immunology[C]. New York:Marcel Dekker Inc,1978,63 - 85.

[4] L N De Castro,F J Von Zuben. The clonal selection algorithm with engineering application[A]. Proceedings of GECCO '00, Workshop on Artificial Immune System and Their Applications[C]. Las Vegas:Morgan Kaufman,2000,36 - 37.

[5] J Kim,P J Bentley. Towards an artificial immune system for network intrusion detection:an investigation of dynamic clonal selection[A]. Proceedings of Congress on Evolutionary Computation[C]. Washington DC:IEEE Press,2002,1015 - 1020.

[6] H F Du,et al. A novel artificial immune system algorithm for high-dimensional function numerical optimization[J]. Progress in Natural Science,2005,15(5):463 - 471.

[7] C Erick. A Survey of Parallel Genetic Algorithms. Department of Computer Science[R]. Urbana. IL:University of Illinois at Urbana Champaign. 1998.

[8] A Gasper,P Collard. From GAs to artificial immune systems:improving adaptation in time dependent optimization[A]. Proceedings of Congress on Evolutionary Computation[C]. Washington DC:IEEE Press,1999,1859 - 1866.

[9] Y Tanimura,T Hiroyasu,M Miki. Discussion on searching capability of distributed genetic algorithm on the grid[A]. Proceedings of the Congress on Evolutionary Computation[C]. Canberra:IEEE Press,2003,1086 - 1094.

[10] E Noda,A L V Coelho,I L M Ricarte,A Yamakami,A A Freitas. Devising adaptive migration policies for cooperative distributed genetic algorithms[A]. Proceedings of Congress on Systems,Man and Cybernetics[C]. Honolulu:IEEE press,2002,438 - 443.

[11] M Nakamura,N Yamashiro,Y Gong. Iterative parallel and distributed genetic algorithms with biased initial population[A]. Proceedings of Congress on Evolutionary Computation[C]. Piscataway:IEEE press,2004,2296 - 2301.

[12] R Tanese. Parallel genetic algorithm for a hypercube[A]. Proceedings of the Second International Conference on Genetic Algorithms[C]. UK:Lawrence Erlbaum Assoc Inc,1987,177 - 183.

[13] W C Zhong,J Liu,M Z Xue,L C Jiao. A multi-agent genetic algorithm for global numerical optimization[J]. IEEE Transactions on System,Man and Cybernetics,Part B:Cybernetics,2004,34(2):1128 - 1141.

[14] O Brudaru,O Buzatu. Distributed genetic algorithm for finding fuzzy rational approximators[A]. Proceedings of IEEE International Conference on Parallel Computing in Electrical Engineering[C]. California:IEEE Computer Society Press,2004,394 - 397.

[15] B Porter,F Xue. Niche evolution strategy for global optimization[A]. Proceedings of the Congress on Evolutionary Computation[C]. New Jersey:IEEE Press,2001,1086 - 1092.

作者简介:

戚玉涛　男,1981 年 10 月出生于河南潢川. 西安电子科技大学电子工程学院博士生. 主要研究方向包括:进化算法、人工免疫系统、并行计算等. E-mail:qi.yutao@163.com

刘　芳　女,1963 年 2 月生于湖南华容. 现为西安电子科技大学计算机学院教授,博士生导师. 研究方向包括:人工智能、图像处理、模式识别和进化计算等.