基于 Memetic 算法的 DNA 序列数据压缩方法

谭 丽* 孙季丰 郭礼华
(华南理工大学电子与信息学院 广州 510641)

摘 要：该文提出一种基于 CPMA(Collaborative Particle swarm optimization-based Memetic Algorithm) 算法的 DNA 序列数据压缩方法，CPMA 分别采用综合学习粒子群优化(Comprehensive Learning Particle Swarm Optimization, CLPSO)算法和动态调整的混沌搜索算子(Dynamic Adjustive Chaotic Search Operator, DACSO)进行全局搜索和局部搜索。该文采用 CPMA 寻找全局最优的基于扩展操作的近似重复矢量(Extended Approximate Repeat Vector, EARV)码书，并用此码书压缩 DNA 序列数据。实验结果表明，CPMA 比其它优化算法有很大的改善，对文中采用的大部分测试函数，其解都非常接近全局最优点；对于 DNA 基准测序序列，与文中所列的经典 DNA 序列压缩算法相比，基于 CPMA 算法的压缩性能得到了显著提升。

关键词：DNA 序列压缩；Memetic 算法；扩展的近似重复矢量(EARV)；粒子群优化(PSO)；动态混沌局部搜索
中国分类号：TP391 **文献标识码**：A **文章编号**：1009-5896(2014)01-0121-07
DOI: 10.3724/SP.J.1146.2013.00303

DNA Sequence Data Compression Method Based on Memetic Algorithm

Tan Li Sun Ji-feng Guo Li-hua
(*School of Electronic and Information Engineering*, *South China University of Technology*, *Guangzhou* 510641, *China*)

Abstract: A DNA sequence compression method based on Collaborative Particle swarm optimization-based Memetic Algorithm (CPMA) is proposed. CPMA adopts the Comprehensive Learning Particle Swarm Optimization (CLPSO) as the global search and a Dynamic Adjustive Chaotic Search Operator (DACSO) as the local search respectively. In CPMA, it looks for the global optimal code book based on Extended Approximate Repeat Vector (EARV), by which the DNA sequence is compressed. Experimental results demonstrate better performance of HMPSO than the other optimization algorithms, and it is very close to the global optimization point in most of the test functions adopted by the paper. The compression performance of the method based on CPMA is markedly improved compared to many of the classical DNA sequence compression algorithms.

Key words: DNA sequence compression; Memetic algorithm; Extended Approximate Repeat Vector (EARV); Particle Swarm Optimization (PSO); Dynamic chaotic local search

1 引言

DNA(DeoxyriboNucleic Acid)是由多个核苷酸头尾相连而成的生物大分子化合物，是储存、复制和传递遗传信息的主要物质基础[1]。随着人类 DNA 序列测定工程的快速发展，DNA 序列数据压缩技术的研究已经成为人们关注的热点之一。如何在有限的存储资源内有效储存急剧膨胀的 DNA 序列数据，用较小的存储空间存放较大的 DNA 序列数据是计算机专家和生物学家面临的新课题。

由于 DNA 序列数据的特殊性，使用传统的压缩算法并不是很理想。由此出现了许多专门针对 DNA 序列的压缩算法，如 GenCompress[2], DNApack [3], GeNML[4]等。这些算法大部分都属于基于替代的压缩算法，主要是利用了 DNA 序列数据中的冗余特性 (如直接重复、镜像、反转和互补回文)，使用了基于字典或其它更为简略的表达方式去替代存储匹配的 DNA 序列子串，以达到压缩的目的。然而 DNA 序列数据压缩算法的性能评价标准[5]表明，已有的 DNA 压缩算法存在耗时长，资源消耗大，压缩性能不稳定等问题。针对以上问题，纪震等人[6]于 2010 年首次将 DNA 序列数据的生物信息学特征和生物学含义引入到压缩处理中，打破了以往仅从数据构成特点入手的 DNA 序列压缩算法模式。文献[7]提出一种在线 DNA 压缩机算法

2013-03-12 收到，2013-07-27 改回
国家自然科学基金(60902087)和广州市科委新星计划项目(2010A090100016)资助课题
*通信作者：谭丽 t.li07@mail.scut.edu.cn

(DNA Sequence Compressor, DNASC)，该算法从水平和垂直方向对 DNA 序列数据进行在线压缩，最大特点是基于替代和统计的混合算法。2012 年，文献[8]又提出基于自配置单粒子优化的 DNA 序列压缩算法 (Self-Configuration Single Particle Optimizer, SCSPO)，采用自配置结构的粒子群优化算法优化码书，然后利用此码书压缩 DNA 序列数据，从而达到较好的压缩效果；该算法成功地将智能优化算法应用于 DNA 序列数据的压缩过程。

本文借鉴 SCSPO 算法思想，提出了另外一种基于 Memetic 算法的 DNA 序列数据的压缩方法 (Collaborative Particle swarm optimization-based Memetic Algorithm, CPMA)。CPMA 算法将综合学习粒子群优化算法 (Comprehensive Learning Particle Swarm Optimizer, CLPSO)[9]作为全局搜索，同时设计出一种动态调整的混沌搜索算子 (Dynamic Adjustive Chaotic Search Operator, DACSO)作为局部搜索，两者引入到 Memetic 算法框架协同进化，找出全局最优的基于扩展操作的近似重复矢量(Extended Approximate Repeat Vector, EARV)码书，用于 DNA 序列数据的压缩处理。

2 基于扩展操作的近似重复矢量(EARV)码书模型

在文献[10]提出的近似重复矢量(ARV)码书模型的基础上，本文提出一种基于扩展操作的近似重复矢量(EARV)码书模型。将 3 种基本编辑操作(替代 R，字符插入 Insc，字符删除 Dc)扩展为 8 种，来完成更多的近似匹配重复模式。这 8 种编辑操作，可用 3 bit(000~111)来表示。ARV 模型中的 β (一个编辑操作使用的比特数)大小为 $\lceil \log_2 M \rceil + 5$，以 β_i 表示 EARV 中第 i 种扩展编辑操作所需的比特数目，其中 i=1,2,…,5，用 M 表示码书中 EARV 的长度，则 5 种扩展编辑操作的定义以及 β_i 与 β 的比较描述如下：

(1)互补替代(Complementary replace)：记为(Rc, pos)，表示在序列子串中第 pos 个位置处的字符替换为对应的互补字符。相对于替代操作，减少了第 3 操作子。$\beta_1 = \lceil \log_2 M \rceil + 3$，有 $\beta_1 < \beta$；

(2)字符串插入(Insert string)：记为(Inss, pos, len, str)，表示在序列子串 pos 和 pos+1 的位置之间插入长度为 len 的字符串 str。$\beta_2 = 3+2\times\lceil \log_2 M \rceil + 2\times$ len，采用 ARV 编码所需的编辑操作比特数目 $\beta_{\mathrm{ARV}} = \mathrm{len}\times\beta$，则 $\beta_{\mathrm{ARV}} - \beta_2 = 3\times(\mathrm{len}-1) + \lceil \log_2 M \rceil \times(\mathrm{len}-2)$，因为 len>1，则 $(\beta_{\mathrm{ARV}} - \beta_2) > 0$，即 $\beta_2 < \beta_{\mathrm{ARV}}$；

(3)字符串删除(Delete string)：记为(Ds, pos, len)，表示删除序列子串中从 pos 开始的长度为 len 的字符串。$\beta_3 = 3 + 2\times\lceil \log_2 M \rceil$，采用 ARV 编码所需的编辑操作比特数目 $\beta_{\mathrm{ARV}} = \mathrm{len}\times\beta$，则 $\beta_{\mathrm{ARV}} - \beta_3 = (\mathrm{len}-2)\times\lceil \log_2 M \rceil + 5\times\mathrm{len} - 3$，因 len>1，所以 $(\beta_{\mathrm{ARV}} - \beta_3) > 0$，即 $\beta_3 < \beta_{\mathrm{ARV}}$；

(4)字符交换(Exchange)：记为$(E, \mathrm{pos}_1, \mathrm{pos}_2)$，表示交换序列子串中 pos_1 位置和 pos_2 位置的字符。$\beta_4 = 3+2\times\lceil \log_2 M \rceil$，采用 ARV 编码所需的编辑操作比特数目 $\beta_{\mathrm{ARV}} = 2\times\beta$，则计算可得 $\beta_{\mathrm{ARV}} - \beta_4 = 7 > 0$，即 $\beta_4 < \beta_{\mathrm{ARV}}$；

(5)反序(Inversion)：记为(Inv, pos, len)，其中 pos 为待反序串的起始位置，len 为待反序串的长度。$\beta_5 = 3+2\times\lceil \log_2 M \rceil$，采用 ARV 编码所需的编辑操作比特数目 $\beta_{\mathrm{ARV}} = \mathrm{len}\times\beta$，因 len>1，有 $\beta_5 < \beta_{\mathrm{ARV}}$。

从以上定义及分析可知，EARV 比 ARV 编码所需的比特数更少。

假设有如下一段 DNA 序列：

$$\cdots \mathrm{AATCTG} \cdots \mathrm{GCCAA} \cdots \mathrm{TTGC}$$
$$\cdots \mathrm{CATTG} \cdots \mathrm{GACTA} \cdots \quad (*)$$

用 v, v^{-1}, v', v'^{-1} 分别表示 DNA 序列中的直接、镜像、反转、互补回文 4 种重复模式。如果模式串为“AACTG”，则序列(*)应用 EARV 码书模型后被编辑为如图 1 所示。

图 1 中的 EARV 编码后的 4 个参数分别代表相对位置，重复模式种类，编辑距离和编辑操作。假设第 1 个碱基片段“AATCTG”在序列(*)中的绝对位置为 a_1，在模式 v 下对模式串“AACTG”的第 3 个位置前插入碱基“T”就可以得到碱基片段“AATCTG”，该近似重复矢量可表示为$\{a_1, v, 1, (\mathrm{Insc}, 3, \mathrm{T})\}$；同样在序列(*)$a_2$ 绝对位置处的第 2 个碱基片段“GCCAA”，近似重复矢量标记为$\{a_2\text{-}a_1, v^{-1}, 1, (R, 2, \mathrm{C})\}$，表示模式串“AACTG”在镜像匹配$(v^{-1})$下将第 2 个碱基替换为碱基“C”，$a_2\text{-}a_1$ 表示第 2 个碱基片段“GCCAA”相对于第 1 个碱基片段“AATCTG”的位置。序列(*)中其余 3 个近似片段的 EARV 编码过程类似。

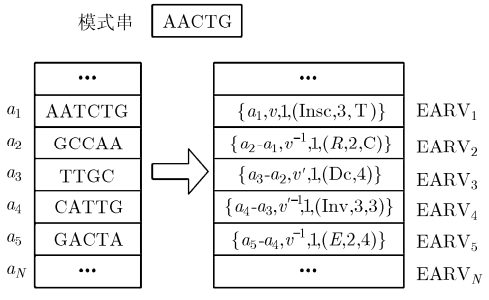

图 1 EARV 码书中的近似匹配实例

3　基于 Memetic 算法的 DNA 序列数据压缩方法

与单纯的进化算法相比，Memetic 算法往往能以更少的计算代价，得到具有更高精确度的解，在解决复杂、高维、大规模问题时具有明显的优势[11]，近年来受到了国内外学者的重视。考虑到 Memetic 算法是基于种群全局搜索和个体局部搜索的混合方法，本文提出了一种基于 CPMA 的 DNA 序列数据压缩方法。

3.1 粒子编码和适应度函数的计算

将每个 EARV 按顺序连接，构造 CPMA 寻优粒子的位置矢量，对其进行粒子编码，如图 2 所示，其中 M, N 分别表示码书中 EARV 的长度和数量。

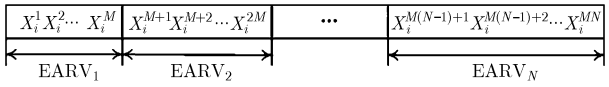

图 2 使用 EARV 码书模型构造寻优第 i 个粒子

适应度函数的计算过程如图 3 所示，先将寻优粒子位置矢量离散切分为含有 EARV 的码书，接着用 AGREP[12](Approximate Global search Regular Expression and Print out the line)近似搜索 DNA 原始序列中含有的 EARV，最后将 DNA 序列数据的压缩率(Bit Per Base, BPB)作为 CPMA 的适应度函数值：

$$f(X)=\mathrm{BPB}=N_{\mathrm{bit}}/N_{\mathrm{base}} \tag{1}$$

其中 N_{base} 表示待压缩的 DNA 序列中碱基个数，N_{bit} 为编码这些碱基所需的比特数目。适应度函数值越小，说明对 DNA 序列数据的压缩率越低，即压缩性能就越好。

式(1)的 N_{bit} 计算如式(2)：

$$N_{\mathrm{bit}}=2MN+\alpha\times N_{\mathrm{mf}}+\beta_{\mathrm{EARV}}\times N_{\mathrm{ed}}+2\times(N_{\mathrm{base}}-N_{\mathrm{mt}}) \tag{2}$$

对式(2)进一步说明如下：

(1)存储编码所用的码书，需要(2MN) bit；

(2)N_{mf} 代表 EARV 的个数，所需比特数 $\alpha=\lceil\log_2\mathrm{maxra}\rceil+\lceil\log_2 N\rceil+\lceil\log_2\mathrm{maxed}\rceil+2$，由相对位置，模式串在码书中的序列号，编辑距离和匹配类型组成；

(3)N_{ed} 表示编辑操作的个数。β_{EARV} 为描述一个编辑操作所需的平均比特数目，各编辑操作所使用的比特数目在第 2 节已做分析；

(4)N_{mt} 表示与码书中的某模式串形成“匹配”的基本碱基个数。N_{base}-N_{mt} 表示未形成“匹配”的碱基总个数，每个碱基的编码采用 2 bit 的基本编码。

初始化一粒子群，包含 ps 个粒子，一个粒子代表一个码书，维度为 D。则寻优粒子位置矢量离散切分为含有 EARV 的码书过程如下：

(1)对于种群中的第 i 个粒子，其位置矢量 $\boldsymbol{X}_i=[X_i^1, X_i^2, \cdots, X_i^j, \cdots, X_i^D]$；

(2)将位置矢量 $\boldsymbol{X}_i$ 按照式(3)离散化为 DNA 序列中的 4 种碱基(A, T, C, G)符号的离散矢量，即 $\boldsymbol{S}_i=[s_i^1, s_i^2, \cdots, s_i^j, \cdots, s_i^D]$，令粒子 i 的位置矢量最大、最小值分别为 $X_{\max}$, $X_{\min}$，并令 $\delta=(X_{\max}-X_{\min})/4$；

(3)本文将 $\boldsymbol{S}_i$ 以 M=8 切分为含有 N=32 个 EARV 的码书，即 Codebook$_i$=[EARV$_1$, EARV$_2$, $\cdots$, EARV$_N$]。其中，EARV$_k$=[$s_{(k-1)\times 8+1}, s_{(k-1)\times 8+2}, \cdots, s_{(k-1)\times 8+8}$], k =1, 2, $\cdots$,32。

$$\Omega_i^j=\begin{cases}\mathrm{T}, & X_i^j\in[X_{\min},\ X_{\min}+\delta)\\ \mathrm{G}, & X_i^j\in[X_{\min}+\delta,\ X_{\min}+2\delta)\\ \mathrm{C}, & X_i^j\in[X_{\min}+2\delta,\ X_{\min}+3\delta)\\ \mathrm{A}, & X_i^j\in[X_{\min}+3\delta,\ X_{\max}]\end{cases} \tag{3}$$

3.2 全局搜索

在 CPMA 算法中，采用 CLPSO 作为全局搜索策略，以保证粒子种群的多样性。CLPSO 算法的速度和位置更新公式[9]如式(4)至式(7)：

$$V_i^d(k+1)=w\times V_i^d(k)+c\times\mathrm{rand}_i^d\times(\mathrm{pbest}_{f_i(d)}^d-X_i^d(k)) \tag{4}$$

$$V_i^d(k+1)=\min(V_{\max}^d,\ \max(V_{\min}^d, V_i^d(k+1))) \tag{5}$$

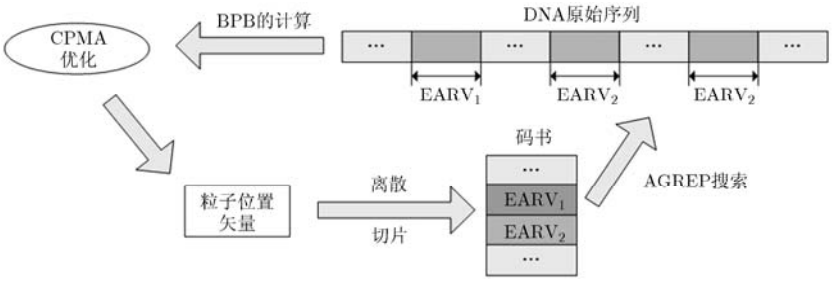

图 3 适应度函数的计算过程

$$X_i^d(k+1)=X_i^d(k)+V_i^d(k+1) \tag{6}$$

$$X_i^d(k+1)=\min(X_{\max}^d,\max(X_{\min}^d,\ X_i^d(k+1))) \tag{7}$$

其中 $\boldsymbol{V}$ 为粒子速度矢量，$\boldsymbol{X}$ 为位置矢量。变量 i 为当前更新粒子的序号，d 为粒子的维数($0<d<D$)，k 为迭代次数；w 为惯性权重；rand 为[0,1]范围内的均匀随机数；c 为学习因子；$f_i(d)$表示粒子 i 在第 d 维的学习对象。式(5)和式(7)分别控制粒子的速度和位置范围为$[V_{\min},\ V_{\max}]^D$, $[X_{\min},\ X_{\max}]^D$。

CLPSO 全局搜索策略善于探索出解空间中的优秀区域，保证了 CPMA 算法的“开拓能力”，而采用 DACSO 的局部搜索策略则能够很快地发现局部区域中的最优点，保证了 CPMA 算法的“开发能力”。给定变化阈值 f_s，当连续 T 次有 $\Delta f<f_s$，则判定算法已处于早熟收敛状态。其中 Δf 为

$$\Delta f=\frac{|\ f(\mathrm{pbest}_i^k)-f(\mathrm{pbest}_i^{k+1})\ |}{|\ f(\mathrm{pbest}_i^k)\ |} \tag{8}$$

其中 $\mathrm{pbest}_i{}^k$和 $\mathrm{pbest}_i{}^{k+1}$ 分别表示第 i 个粒子在 k 和 k+1 次更新后的局部最优位置。

依次考察粒子 i 的第 d 维速度值 V_i^d，与停滞速度阈值 V_s 作比较，若 $V_i^d<V_s$，则用混沌搜索算子(Chaotic Search Operator, CSO)对粒子 i 进行局部搜索。为了从当前种群中挑选出合适的粒子进行局部搜索，本文采用 DACSO 的方法，在上一次的局部搜索过程中，若搜索出较多的更优粒子，则在下一次的搜索中，增加其搜索比重。本文中的局部搜索主要针对活泼粒子，采取的策略是：设定活跃系数 ca 和引入差值系数 cd。这两个系数分别动态调整 T 和混沌搜索次数，以使优秀粒子被赋予更多的搜索机会，使非优秀粒子减少搜索机会，以使局部搜索占用的计算开销更有意义。其中第 i 个粒子的活跃系数 ca_i 定义为式(9)。

$$\mathrm{ca}_i=\max(5/T,\ 1-0.05\times\mathrm{act}_i) \tag{9}$$

其中 act_i 表示上一次的局部搜索中粒子 i 寻优成功的次数。

CPMA 算法的具体步骤如表 1 所示。

3.3 局部搜索

考虑到将局部搜索策略应用到每一个粒子上进行搜索，会大大增加算法的运行时间成本，本文采用了 DACSO 的局部搜索方法。在 DACSO 中通过 cd 去动态调整混沌搜索次数，使其在较少的计算次数内获得更大的性能提升。差值系数 cd_i 定义为式(10)

$$\mathrm{cd}_i=1-\mathrm{dis}_i/\max(\mathrm{dis}_i) \tag{10}$$

其中 $\mathrm{dis}_i=|\ f(\mathrm{pbest}_i)-f(\mathrm{gbest})|$。同时本文还引进动态收缩半径的机制，并在收缩区域内随机产生粒子来替代性能较差的粒子，如式(11)：

表 1 CPMA 算法的具体步骤

步骤 1　设置粒子初始位置范围$[X_{\min},\ X_{\max}]^D$、粒子速度范围$[V_{\min},\ V_{\max}]^D$及初始化规模为 ps，维度为 D 的粒子群，计算其初始 pbest_i及 gbest(粒子所对应的全局最优位置)，设置搜索空间范围，令 ca_i=1, act_i=0, i=1, 2,⋯,ps；

步骤 2　设置各参数包括 flag, m, P_c, c(参考 CLPSO[9]), V_s, f_s, T，最大局部搜索次数 cnt；

步骤 3　若未达到终止条件(即函数计算的次数小于最大次数)，则执行步骤 4～步骤 10，否则执行步骤 11；

步骤 4　利用式(4)～式(7)分别更新粒子的速度和位置；

步骤 5　根据式(1)计算每个粒子的适应度函数值，并更新 pbest_i, gbest 和 flag；

步骤 6　根据式(8)计算 Δf_i 值，若 Δf_i 连续小于 f_s的次数达到 $T\times\mathrm{ca}_i$，则执行步骤 7，否则跳至步骤 10；

步骤 7　令 $X_{\mathrm{temp}}=X_i$，依次考察粒子 i 的第 d 维速度值 V_i^d，若 $V_i^d<V_s$，则执行步骤 8，否则跳至步骤 9；

步骤 8　对从当前种群中挑选出来的合适的粒子采用 DACSO 的局部搜索；

步骤 9　根据式(9)计算 ca_i；

步骤 10　$\mathrm{act}_i=0$，跳至步骤 3；

步骤 11　结束。

$$r=r\times(1-0.5\times|\eta|),\quad \eta\sim\mathrm{Gauss}(0,1) \tag{11}$$

其中 r 随着进化迭代次数的增加而动态地变小，其值越小说明 DACSO 的搜索范围越小；η 服从期望值为 0，方差为 1 的高斯分布。设置混沌初始值 cx_1=rand(0,1)，混沌搜索半径的初值 $r_1=(V_{\max}-V_{\min})/2$，令 j=1, DACSO 的局部搜索方法步骤如表 2 所示。

表 2 DACSO 的局部搜索方法步骤

步骤 1　利用式(12)生成 Logistic 序列[13]第 j 项；

$$\mathrm{cx}_j=\lambda\times\mathrm{cx}_j\times(1-\mathrm{cx}_j) \tag{12}$$

其中当 λ=4 时，Logistic 序列具有混沌特性；

步骤 2　若 $j\le\mathrm{cnt}\times\mathrm{cd}_i$，则执行步骤 3，否则跳至步骤 7；

步骤 3　利用式(13)使混沌映射在粒子 i 的个体极值附近小的邻域内；

$$X_j=\mathrm{pbest}_i+r_j\times(2\times\mathrm{cx}_j-1) \tag{13}$$

其中 X_j表示第 j 次搜索获取的 pbest_i的邻域位置；

步骤 4　若 $f(X_j)<f(\mathrm{pbest}_i)$，则 $\mathrm{pbest}_i=X_j$, $\mathrm{act}_i=\mathrm{act}_i+1$；

步骤 5　利用式(11)动态收缩 CSO 的搜索范围；

步骤 6　$j=j+1$，跳至步骤 1；

步骤 7　结束。

整体上来说，CPMA 算法根据式(8)从群体中选出备用粒子再进行局部搜索。随着迭代的进行，局部搜索 DACSO 的频率呈上升趋势，全局搜索 CLPSO 根据 DACSO 的反馈进行粒子间的信息交流，再将结果作用于式(8)，选定新一代即将进行局部搜索的粒子。

4　实验结果

4.1 参数设置

本实验将 CPMA 与其它 4 种 PSO 改进算法(CLPSO[9], CEPSO9[14], ISPO[15], SCSPO[8])分别对 19 个高维测试函数集[16]和 11 条 DNA 基准测序序列[17]进行实验。其中，19 个高维测试函数具有不同的特点，如单峰或高峰函数、随着维度的增加是否更加容易被优化等；11 条 DNA 基准测序序列来源于美国基因数据库(GenBank)，包含了不同物种不同功能的 DNA 数据片段，能够有效评估压缩算法对含有不同特征的 DNA 序列的压缩能力。我们设置各算法对 19 个高维测试函数分别在维度 D=100 和 D=500 各执行 25 次，且每次执行中函数的最大计算次数 MaxFEs 为 5000×D, 5 种比较算法的参数设置如表 3 所示。

表 3 各算法的参数设置

算法	参数
CLPSO[9]	|ps|=10, w:0.9~0.4, c=1.49445, m=8
CEPSO9[14]	|ps|=10, w=0.8, $c1$=$c2$=2
ISPO[15]	J=30, a=150, p=10
SCSPO[8]	|ps|=10, w=0.5, $c1$=$c2$=2, M=10, N=30
CPMA	|ps|=10,f_s=1E−6,V_s=1E−4,T=200,cnt=100

表 3 中，CPMA 算法中的全局搜索 CLPSO 参数 w_0,w_1,c,m 的设置同文献[9]；实验中为更好地选择进行局部搜索的粒子，一般取 f_s<1E−3；经考察测试函数在迭代中各粒子 Δf 的变化趋势，发现 Δf 连续不变或极小变化的次数一般小于 1E+3，所以 T<1E+3; CPMA 算法中粒子各维速度的更新主要是在全局搜索 CLPSO 中进行，到进化中后期其值将变得非常小，V_s>1E−7 则可改善 CPMA 在进化中前期 LS 搜索率过低的问题；cnt 的取值在 1E+2 左右能较好地平衡全局和局部搜索。除此之外，粒子的位置搜索范围[$X_{\min}$, $X_{\max}$]设置见文献[16]。

4.2 19 个高维测试函数

5 种 PSO 改进算法在 D=100 和 D=500 下对 19 个高维测试函数的平均误差值对比结果分别见表 4 和表 5。做“显著性水平”的判断是为了说明算法之间的差异是否属于机会变异，即是否具有统计上的显著差异，由此来帮助分析算法的性能。本文显著性水平测试是在各个函数上，将不同算法运行 25 次产生的适应值数据序列，进行双尾配对样本 t 检验(two-tailed paired t-test)，其中显著性水平因子 $\alpha = 0.05$。表 4 和表 5 中符号“†”表示统计显著优胜值，其实验结果值小于10^{-14}近似零处理。

由表 4 和表 5 数据可知，与其它 4 种优化算法相比，CPMA 算法在测试函数 F_1,F_6,F_7,F_9~F_{12},F_{15}, F_{16},F_{18}和 F_{19}上的平均误差值均取得最佳结果；相对 CLPSO 算法，CPMA 算法在大部分测试函数上都能达到较好的结果(除 F_3,F_5 以外)，尤其在部分函数(F_1,F_6,F_7,F_{10},F_{15}和 F_{19})的优化上达到或非常接近理论最优值，显示出局部搜索 DACSO 较好的性能，同时，在函数 F_2,F_3,F_8,F_{13}和 F_{17}上，两个算法的优化性能相当；CEPSO9 算法是 5 种改进算法中优化效果最差的算法，其所有的平均误差值都远大于其它算法的优化结果，故 CEPSO9 算法应用于优化高

表 4 各算法在 D=100 的平均误差值优化结果

函数	CLPSO	CEPSO9	ISPO	SCSPO	CPMA
F_1	7.03E−07	2.60E+05	6.87E+02	7.54E−02	**0.00E+00**†
F_2	4.82E+01	1.51E+02	7.30E+01	**3.81E−01**†	1.18E+02
F_3	**3.23E+02**	1.87E+11	8.16E+07	5.00E+02	4.33E+03
F_4	9.87E+01	1.63E+03	6.46E+02	**1.20E+00**	1.39E+00
F_5	**1.42E−05**†	2.52E+03	7.52E−01	2.24E−01	1.77E−01
F_6	6.62E+00	2.11E+01	1.80E+01	2.19E+00	**0.00E+00**†
F_7	3.28E-04	5.51E+02	5.99E−10	6.87E−02	**0.00E+00**†
F_8	1.77E+04	2.18E+05	5.17E+04	**7.08E−01**†	9.07E+04
F_9	5.10E+01	1.14E+03	5.63E+02	5.82E+00	**6.49E−01**
F_{10}	4.33E−05	1.37E+04	8.28E+01	1.96E−01	**0.00E+00**†
F_{11}	7.74E+01	1.18E+03	5.75E+02	4.98E+00	**7.67E−01**
F_{12}	8.12E+00	2.31E+05	3.28E+02	8.17E−01	**6.03E−03**†
F_{13}	2.46E+02	1.11E+11	3.21E+07	**6.35E+01**	1.31E+02
F_{14}	3.00E+01	1.13E+03	5.29E+02	1.84E+00	**2.06E−01**
F_{15}	1.86E−04	4.74E+03	1.99E+01	7.34E−02	**0.00E+00**†
F_{16}	2.92E+01	1.04E+05	4.58E+02	2.13E+00	**1.48E−02**†
F_{17}	1.26E+02	7.45E+09	4.35E+06	**6.91E+00**	1.81E+02
F_{18}	1.13E+01	5.03E+02	3.20E+02	1.64E+00	**4.28E−01**
F_{19}	4.72E−05	1.36E+04	6.13E+01	8.84E−02	**0.00E+00**†

表 5 各算法在 D=500 的平均误差值优化结果

函数	CLPSO	CEPSO9	ISPO	SCSPO	CPMA
F_1	9.26E−02	2.15E+06	1.50E+03	7.15E−02	**0.00E+00**†
F_2	6.88E+01	2.03E+02	2.88E+02	**3.96E−01**†	1.56E+02
F_3	**5.54E+02**	2.49E+12	4.26E+07	1.27E+03	6.17E+03
F_4	1.35E+02	1.04E+04	1.55E+03	**1.86E+00**	2.00E+00
F_5	**9.40E−05**†	1.77E+04	3.93E+00	1.79E−01	4.89E−01
F_6	6.94E−02	2.20E+01	8.98E+01	2.02E−01	**0.00E+00**†
F_7	2.36E−03	2.76E+03	1.37E−09	7.27E−02	**0.00E+00**†
F_8	5.25E+04	6.10E+06	8.46E+04	**6.73E−01**†	3.18e+06
F_9	1.70E+02	6.30E+03	1.41E+03	5.56E+00	**8.56E−01**
F_{10}	2.50E−04	9.94E+04	2.00E+02	1.77E−01	**0.00E+00**†
F_{11}	2.31E+02	6.30E+03	1.39E+03	5.42E+00	**3.47E+00**
F_{12}	3.93E+01	1.92E+06	1.51E+03	7.41E−01	**1.44E−01**
F_{13}	2.23E+02	7.42E+11	2.91E+07	**1.50E+02**	7.23E+03
F_{14}	2.39E+01	7.99E+03	1.43E+03	**1.22E+00**	2.07E+00
F_{15}	4.62E−04	2.21E+04	5.08E+01	8.35E−02	**0.00E+00**†
F_{16}	9.55E+01	1.09E+06	1.35E+03	2.15E+00	**2.02E−01**
F_{17}	2.27E+02	2.59E+11	9.99E+06	**8.85E+00**	3.51E+02
F_{18}	2.77E+01	3.27E+03	8.74E+02	1.43E+00	**6.60E−01**
F_{19}	1.75E−04	7.64E+04	1.54E+02	1.11E−01	**0.00E+00**†

维多模函数时有待进一步改进; SCSPO 算法的平均误差值优化结果全部优于 ISPO 算法的优化结果，这主要是因为SCSPO算法是在ISPO算法基础上添加了自适应处理机制，从而使得高维多模函数的优化性能得以提升。但与 CPMA 算法比较，在 19 个高维测试函数中除了 F_2,F_3,F_8,F_{13},F_{14} 和 F_{17} 以外，其优化性能都不及CPMA算法。配对 t 检验结果表明，相对其它算法 CPMA 在函数 F_1,F_6,F_7,F_{10},F_{15} 和 F_{19} 上是统计显著的，具有统计学意义。

4.3 DNA 基准测序序列

假设 5 种 PSO 改进算法对 11 条 DNA 基准测序序列优化码书大小均为 M=8, N=32，优化时均执行 30 次且 MaxFEs 设置为 2×10^5。实验使用各算法在基准测序序列上的压缩率作为评价指标，并与传统经典 DNA 序列压缩算法(如 Gencompress[2](Gen),DNAcompress[18](DNAc),DNApack[3](DNAp), GeNML[4])进行对比，结果如表 6 所示。

表 6 表明，相对于其它压缩算法，本文提出的 CPMA 算法对 DNA 基准测序序列的压缩率(BPB)最小。同时 5 种优化算法与经典的 DNA 序列数据压缩算法相比，对 DNA 序列的压缩率得到了提高，并且不同序列的压缩效果也比较稳定。

5 结论

本文提出一种基于 Memetic 算法的 DNA 序列数据压缩方法，即 CPMA 算法。该方法采用 CLPSO,

表 6 各算法对 11 条 DNA 基准测序序列的压缩率比较

序列	Gen	DNAc	DNAp	GeNML	CLPSO	CEPSO9	ISPO	SCSPO	CPMA
CHNTXX	1.6146	1.6127	1.6103	1.6102	1.6101	1.6121	1.6130	1.6009	1.5616
CHMPXX	1.6730	1.6716	1.6602	1.6613	1.6610	1.6911	1.7089	1.6216	1.5995
MPOMTCG	1.9058	1.8920	1.8932	1.8822	1.8805	1.8932	1.9021	1.7776	1.6192
HUMGHCSA	1.0969	1.0272	1.0390	1.0124	1.3328	1.6520	1.6416	1.1921	1.0109
HUMHBB	1.8204	1.7897	1.7771	1.7513	1.6611	1.8039	1.7902	1.6270	1.5197
HUMHSABCD	1.8192	1.7951	1.7394	1.7132	1.7213	1.8008	1.8054	1.6828	1.6176
HUMDYSTROP	1.9231	1.9116	1.9088	1.9126	1.9021	1.9112	1.9189	1.8442	1.7182
HUMHPRTB	1.8466	1.8165	1.7886	1.7587	1.6012	1.7805	1.8124	1.5932	1.5126
VACCG	1.7614	1.7580	1.7583	1.7649	1.6586	1.7329	1.7436	1.5833	1.5272
HEHCMVCG	1.8470	1.8492	1.8346	1.8397	1.7639	1.8172	1.8411	1.6631	1.6059
SCCHRIII	1.9486	1.9518	1.8331	1.9437	1.7437	1.8903	1.9159	1.5989	1.5388
均值	1.7506	1.7341	1.7130	1.7137	1.6851	1.7805	1.7903	1.6168	1.5301

DACSO 的全局搜索和局部搜索相结合的 Memetic 算法寻找 EARV 码书，然后用此码书压缩 DNA 序列数据。CPMA 算法不仅对标准复合测试函数具有较好的优化性能，而且对 DNA 基准测序序列具有较好的压缩率；同时，相比于经典的 DNA 序列数据的压缩算法，将优化算法应用于 DNA 序列压缩，其压缩性能有较大提升。

参 考 文 献

[1] Witold K. Towards cognitive analysis of DNA[C]. 2010 9th IEEE International Conference on Cognitive Informations, Beijing, 2010: 6–7.

[2] Chen X, Kwong S, and Li Ming. A compression algorithm for DNA sequences and its applications in genome comparison[C]. Proceedings of the 10th Workshop on Genome Informatics, Tokyo, 1999: 51–61.

[3] Behzadi B and Fessant F L. DNA compression challenge revisited: a dynamic programming approach[C]. Proceedings of the 16th Annual Symposium on Combinatorial Pattern Matching, Jeju Island, 2005: 190–200.

[4] Korodi G and Tabus I. An efficient normalized maximum likelihood algorithm for DNA sequence compression[J]. *ACM Transactions on Information Systems*, 2005, 23(1): 3–34.

[5] Chern B G, Ochoa I, Manolakos A, *et al*.. Reference based genome compression[C]. 2012 IEEE Information Theory Workshop, Lausanne, 2012: 427–431.

[6] 纪震, 周家锐, 朱泽轩, 等. 基于生物信息学特征的 DNA 序列数据压缩算法[J]. 电子学报, 2011, 39(5): 991–995.
Ji Zhen, Zhou Jia-rui, Zhu Ze-xuan, *et al*.. Bioinformatics features based DNA sequence data compression algorithm[J]. *Acta Electronica Sinica*, 2011, 39(5): 991–995.

[7] Kamta N M, Anupam A, Edries A, *et al*.. An efficient horizontal and vertical method for online DNA sequence compression[J]. *International Journal of Computer Applications*, 2010, 3(1): 39–46.

[8] Ji Zhen, Zhou Jia-rui, Zhu Ze-xuan, *et al*.. Self-configuration single particle optimizer for DNA sequence compression[J]. *Soft Computing*, 2013, 17(4): 675–682.

[9] Liang J J, Qin A K, Suganthan P N, *et al*.. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. *IEEE Transactions on Evolutionary Computation*, 2006, 10(3): 281–295.

[10] Zhu Ze-xuan, Zhou Jia-rui, Ji Zhen, *et al*.. DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm[J]. *IEEE Transactions on Evolutionary Computation*, 2011, 15(5): 643–658.

[11] Wang Hong-feng, Moon I K, Yang S, *et al*.. A memetic particle swarm optimization algorithm for multimodal optimization problems[J]. *Information Science*, 2012, 197: 38–52.

[12] Li Hong-jian, Ni Bing, Wong Man-hon, *et al*.. A fast CUDA implementation of agrep algorithm for approximate nucleotide sequence matching[C]. 2011 IEEE 9th Symposium on Application Specific Processors, San Diego, 2011: 74–77.

[13] Long Min and Tan Li. A chaos-based data encryption algorithm for image/video[C]. 2010 2nd International Conference on MultiMedia and Information Technology, Kaifeng, 2010: 172–175.

[14] Bilal Alatas, Erhan Akin, and A Bedri Ozer. Chaos embedded particle swarm optimization algorithms[J]. *Chaos, Solitons and Fractals*, 2009, 40(4): 1715–1734.

[15] 纪震, 周家锐, 廖惠连, 等. 智能单粒子优化算法[J]. 计算机学报, 2010, 33(3): 556–561.
Ji Zhen, Zhou Jia-rui, Liao Hui-lian, *et al*.. A novel intelligent single particle optimizer[J]. *Chinese Journal of Computers*, 2010, 33(3): 556–561.

[16] Lozano M, Molina D, and Herrera F. Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems[J]. *Soft Computing*, 2011, 15(11): 2085–2087.

[17] Benson D A, Karch-Mizrachi I, Lipman D J, *et al*.. GenBank[J]. *Nucleic Acids Research*, 2011, 39: D32–D37.

[18] Chen X, Li M, Ma B, *et al*.. DNACompress: fast and effective DNA sequence compression[J]. *Bioinformatics*, 2002, 18(12): 1696–1698.

谭　丽：　女，1984 年生，博士生，研究方向为生物信息学、图像与视频处理、计算智能.

孙季丰：　男，1962 年生，教授，博士生导师，研究方向为图像与视频处理、自组织通信网.

郭礼华：　男，1978 年生，副教授，研究方向为图像理解、图像分析、模式识别.