doi:10.3772/j.issn.1002-0470.2015.03.001

# 求解两物种小系统发育问题的遗传算法题①

吴璟莉② 王军伟 胡资鹏

(广西师范大学广西多源信息挖掘与安全重点实验室 桂林 541004)

(广西师范大学计算机科学与信息工程学院 桂林 541004)

**摘 要** 基于复制-丢失比对(DLA)问题模型,研究了复制-丢失(D-L)演化模型下两物种(2-species)小系统发育问题(SPP),缩写为2-SPP-DL问题。通过引入比对算法、标记算法及3种智能变异算子,提出了求解2-SPP-DL问题的遗传算法——G2SP算法。G2SP算法采用普通算子和智能算子相结合的方式,普通算子能有效地保持种群的多样性,而智能算子则能提高种群的收敛性,使其更快地进化到最优解区域。利用4种真实菌属的tRNA和rRNA基因数据对算法性能进行测试,实验结果表明,G2SP算法能够获得较PBLP算法更小的进化代价,是求解2-SPP-DL问题的一种有效方法。

**关键词** 复制,丢失,两物种小系统发育问题,序列对比,遗传算法

## 0 引 言

基因测序技术的飞速发展产生了大量的DNA序列数据,这使得对基因家族分子进化的研究成为可能。近年来基因家族进化史的重建得到了广泛的关注和重视,因为重建基因家族进化史对解决许多基本生物学问题起着非常关键的作用[1]。例如,转运核糖核酸(transfer RNA,tRNA)是合成蛋白质必不可少的重要物质,重建物种间tRNA的进化史或许可为转录机制的研究提供新的思路[1-3]。小系统发育问题(small phylogeny problem,SPP)模型[1,4,5]是构建基因家族进化史的重要问题模型。根据不同的基因序列结构限制条件及不同的演化模型,SPP衍生出各种不同的基因重排问题模型[4,5],以应用于不同的基因组求解。2012年,Holloway等[1]针对基因家族演化主要受到复制(duplication)和丢失(loss)操作影响的特点[6-9],提出了SPP模型的一个子问题模型,即复制-丢失演化模型下的两物种小系统发育问题(2-species SPP in the Duplication-Loss model,2-SPP-DL)。Holloway等[1]进一步把2-SPP-DL问题归约成复制-丢失比对(Duplication-Loss Alignment,DLA)问题,并提出了求解DLA问题的伪布尔线性规划(Pseudo-Boolean Linear Programming,PBLP)算法。本文对DLA问题模型进行了研究,通过引入比对算法、标记算法及3种新颖的智能变异算子,提出了求解该问题的遗传算法(Genetic algorithm for solving 2-SPP-DL,G2SP)。实验结果表明,G2SP算法求得的进化代价比PBLP算法更少。

## 1 问题及符号定义

给定字母表$\Sigma$,其中每个字母表示特定的基因家族。令字符串$\boldsymbol{X}=x_1\,x_2\cdots x_n$表示一条取值于$\Sigma$且有重复值的基因序列,即$x_i\in\Sigma(i=1,\cdots,n)$,记为$\boldsymbol{X}\in\Sigma^n$,$\boldsymbol{X}[i..j]$表示子串$x_i\cdots x_j(1\leqslant i<j\leqslant n)$。$\overline{\boldsymbol{X}}$表示$\boldsymbol{X}$的逆串,$\overline{\boldsymbol{X}}=x_n\,x_{n-1}\cdots x_1$。例如,给定字母表$\Sigma=\{a,b,c,d,e\}$,基因组$\boldsymbol{X}=$"acbacd"包含两个

① 国家自然科学基金(61363035,61165009,61272535),广西高等学校科学技术项目(2013YB028),"八桂学者"工程专项,广西多资源信息挖掘与安全重点实验室系统性研究基金(14-A-03-02)和广西区域多源信息集成与智能处理协同创新中心资助项目。

② 女,1978年生,博士,教授;研究方向:生物信息学,进化计算;联系人,E-mail:wjlhappy@mailbox.gxnu.edu.cn

(收稿日期:2014-10-14)

基因家族 a 的基因、两个基因家族 c 的基因、一个基因家族 b 的基因和一个基因家族 d 的基因，$\overline{\boldsymbol{X}}$ = “dcabca”。$\Phi$ 表示一个进化操作集，且 $\Phi = \{D, L\}$，其中 D 和 L 分别表示基因家族演化模型中的复制操作和丢失操作，定义如下。

**定义 1** 复制操作 D：给定基因组 $\boldsymbol{X} = x_1 x_2 \cdots x_n$，将长度为 $k+1$ 的子串 $\boldsymbol{X}[i..i+k]$ 复制到 $\boldsymbol{X}$ 的位置 $j(j<i$ 或 $j>i+k)$ 上，并称原串 $\boldsymbol{X}[i..i+k]$ 为复制源，复制串为复制体。

**定义 2** 丢失操作 L：给定基因组 $\boldsymbol{X} = x_1 x_2 \cdots x_n$，将长度为 $k+1$ 的子串 $\boldsymbol{X}[i..i+k]$ 从 $\boldsymbol{X}$ 中删除。

假设给定两条基因序列 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$，若进化操作序列 $\boldsymbol{O} = O_1, O_2, \cdots, O_m (O_i \in \Phi, i = 1, \cdots, m)$ 能够将 $\boldsymbol{X}$ 转变为 $\boldsymbol{Y}$，则称 $\boldsymbol{O}$ 为 $\boldsymbol{X}$ 到 $\boldsymbol{Y}$ 的进化史，记为 $\boldsymbol{O}_{\boldsymbol{X}\to\boldsymbol{Y}}$。这里限定 $\boldsymbol{O}_{\boldsymbol{X}\to\boldsymbol{Y}}$ 为可见进化史(visible history)，即 $\boldsymbol{X}$ 是 $\boldsymbol{Y}$ 的可见祖先(visible ancestor)[1]。若令 $C(O_i)$ 为 $\boldsymbol{O}_{\boldsymbol{X}\to\boldsymbol{Y}}$ 中第 $i$ 个操作的代价，则 $\boldsymbol{O}_{\boldsymbol{X}\to\boldsymbol{Y}}$ 的进化代价定义如下：

$$C(\boldsymbol{O}_{\boldsymbol{X}\to\boldsymbol{Y}}) = \sum_{i=1}^{m} C(O_i) \tag{1}$$

令 $\Phi_{\boldsymbol{X}\to\boldsymbol{Y}}$ 表示从 $\boldsymbol{X}$ 到 $\boldsymbol{Y}$ 所有可能的进化史的集合，若 $\Phi_{\boldsymbol{X}\to\boldsymbol{Y}}$ 非空，即至少存在一个从 $\boldsymbol{X}$ 到 $\boldsymbol{Y}$ 的进化史，则称 $\boldsymbol{X}$ 是 $\boldsymbol{Y}$ 潜在的祖先(potential ancestor)。$\boldsymbol{X}$ 到 $\boldsymbol{Y}$ 的进化代价 $C(\boldsymbol{X}\to\boldsymbol{Y})$ 定义为集合 $\Phi_{\boldsymbol{X}\to\boldsymbol{Y}}$ 中所有进化史代价的最小值，如下式所示：

$$C(\boldsymbol{X}\to\boldsymbol{Y}) = \min_{\boldsymbol{O}_{\boldsymbol{X}\to\boldsymbol{Y}} \in \Phi_{\boldsymbol{X}\to\boldsymbol{Y}}} C(\boldsymbol{O}_{\boldsymbol{X}\to\boldsymbol{Y}}) \tag{2}$$

2012 年，Holloway 等[1]提出 2-SPP-DL 问题：给定基因组 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$，以及进化操作集 $\Phi$，找到 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 的潜在公共祖先 $\boldsymbol{G}^*$ 以使得进化代价之和 $C(\boldsymbol{G}^*\to\boldsymbol{G}_1) + C(\boldsymbol{G}^*\to\boldsymbol{G}_2)$ 最小。

由于复制和丢失操作并未改变基因的顺序，Holloway 等[1]将 2-SPP-DL 问题转化为 DLA 问题。假设用一个 $2\times\alpha$ 矩阵 $\boldsymbol{G}_{2\times\alpha}$ 表示基因序列 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 的一个比对，其中每个元素取值于集合 $\Sigma \cup \{-\}$。令 $(\boldsymbol{G}_{1i}, \boldsymbol{G}_{2i})(1\leqslant i\leqslant\alpha)$ 为 $\boldsymbol{G}$ 中第 $i$ 列的取值，有如下两种情况：

(1) $\boldsymbol{G}_{1i}$ 匹配 $\boldsymbol{G}_{2i}$：$\boldsymbol{G}_{1i}\neq -, \boldsymbol{G}_{2i}\neq -, \boldsymbol{G}_{1i} = \boldsymbol{G}_{2i}$；

(2) $\boldsymbol{G}_{1i}$ 不匹配 $\boldsymbol{G}_{2i}$：$\boldsymbol{G}_{1i}\neq -, \boldsymbol{G}_{2i} = -$，或 $\boldsymbol{G}_{2i}\neq -, \boldsymbol{G}_{1i} = -$。

其中，$\boldsymbol{G}_{1i}$ 不匹配 $\boldsymbol{G}_{2i}$ 可看作基因组 $\boldsymbol{G}_2(\boldsymbol{G}_1)$ 在此列中有丢失操作，或基因组 $\boldsymbol{G}_1(\boldsymbol{G}_2)$ 在此列有复制操作。因此，序列比对 $\boldsymbol{G}$ 可用一个复制-丢失操作序列来描述，称为 $\boldsymbol{G}$ 的一个标记(labeling)[1]，记为 $l(\boldsymbol{G}) = O_1, O_2, \cdots, O_{|l(\boldsymbol{G})|} (O_i \in \Phi, i = 1, \cdots, |l(\boldsymbol{G})|)$。被标记的比对 $\boldsymbol{G}$ 的代价 $C(\boldsymbol{G})$ 即为 $l(\boldsymbol{G})$ 中各操作的代价和，如下式所示：

$$C(\boldsymbol{G}) = \sum_{i=1}^{|l(\boldsymbol{G})|} C(O_i) \tag{3}$$

基于上述定义，DLA 问题定义如下[1]：给定两条基因序列 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$，找到代价最小的比对 $\boldsymbol{G}$。

## 2 G2SP 算法

### 2.1 比对算法

给定基因序列 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$，比对算法试图在 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 中插入不定数量的字符“-”，以得到 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 的一个比对 $\boldsymbol{G}$。其主要思想是当 $\boldsymbol{G}_{1i}\neq -, \boldsymbol{G}_{2i}\neq -, \boldsymbol{G}_{1i}\neq\boldsymbol{G}_{2i} (1\leqslant i\leqslant \min(|\boldsymbol{G}_1|, |\boldsymbol{G}_2|))$ 时，通过插入字符“-”以得到一个 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 匹配字符数较多的比对，如图 1 所示。图 1(a) 表示基因组 $\boldsymbol{G}_1$ = abbbbd 和 $\boldsymbol{G}_2$ = abd，图 1(b) 表示通过执行比对算法得到的一个比对。比对算法如图 2 所示。

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\boldsymbol{G}_1$: | a | b | b | b | b | d | $\boldsymbol{G}$: | a | b | b | b | b | d |
| $\boldsymbol{G}_2$: | a | b | d | | | | | a | b | - | - | - | d |
| | (a) | | | | | | | (b) | | | | | |

**图 1 基因组 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$(a) 以及 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 的一个对比 $\boldsymbol{G}$(b)**

### 2.2 标记算法

如前所述，给定基因组 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 的比对 $\boldsymbol{G}$，可用一个复制-丢失操作序列来定义它的一个标记，这种标记过程已被证明为 APX-hard[10] 的。本文给出了一种标记比对的启发式算法，该算法分别从左至右和从右至左两个方向标记 $\boldsymbol{G}$，并取对应较小代价的操作序列作为 $\boldsymbol{G}$ 的标记。与文献[1]一致，本文采用如下的代价计算方法：$C(D(k)) = 1, C(L(k)) = k$，整数 $k$ 表示被复制(丢失)的基因个数。由于两个方向的标记原理一致，下面仅给出从左至右的标记算法，如图 3 所示。

**算法 1**:比对算法

**输入**:基因序列 $G_1$ 和 $G_2$

**输出**:$G_1$ 和 $G_2$ 的一个序列对比 $G$

**for** ( $i=1$; $i\leqslant|G_1|$, $i\leqslant|G_2|$; $i++$ )

**if** ($G_{1i}\neq G_{2i}$) **then**

在 $G_2$ 中找最长基因块 $G_2[j..k]$ ($i<j\leqslant k\leqslant|G_2|$) 以使得

$G_2[j..k]=G_1[i..l]$ ($i\leqslant l\leqslant|G_1|$)

在 $G_1$ 中找最长基因块 $G_1[j'..k']$ ($i<j'\leqslant k'\leqslant|G_1|$) 以使得

$G_1[j'..k']=G_2[i..l']$ ($i\leqslant l'\leqslant|G_2|$)

**if** ($|G_2[j..k]|>0$ and $|G_1[j'..k']|>0$) **then** //均找到基因块

**if** ($((k-j+1)-(j-i))\geqslant 0$ and $((k'-j'+1)-(j'-i))\geqslant 0$) **then**

//基因块均长于空格块

**if** ($(k-j)>(k'-j')$) **then**

从 $G_1$ 的第 $i$ 位置开始插入 $(j-i)$ 个'-'; $i=k+1$

**else if** ($(k-j)<(k'-j')$) **then**

从 $G_2$ 的第 $i$ 位置开始插入 $(j'-i)$ 个'-'; $i=k'+1$

**else if** ($(k-j)=(k'-j')$) **then**

**if** ($j'>j$) **then**

从 $G_1$ 的第 $i$ 位置开始插入 $(j-i)$ 个'-'; $i=k+1$

**else**

从 $G_2$ 的第 $i$ 位置开始插入 $(j'-i)$ 个'-'; $i=k'+1$

**else if** ($((k-j)-(j-i))\geqslant 0$) **then**

从 $G_1$ 的第 $i$ 位置开始插入 $(j-i)$ 个'-'; $i=k+1$

**else if** ($((k'-j')-(j'-i))\geqslant 0$) **then**

从 $G_2$ 的第 $i$ 位置开始插入 $(j'-i)$ 个'-'; $i=k'+1$

**else**

$r=\text{rand}()\%2+1$; $G_{ri}=$''; $i++$ //随机产生下标 $r$

**else if** ($|G_2[j..k]|>0$) **then** //在 $G_2$ 中找到基因块

**if** ($((k-j)-(j-i))\geqslant 0$) **then**

从 $G_1$ 的第 $i$ 位置开始插入 $(j-i)$ 个'-'; $i=k+1$

**else**

$r=\text{rand}()\%2+1$; $G_{ri}=$''; $i++$

**else if** ($|G_1[j'..k']|>0$) **then** //在 $G_1$ 中找到基因块

**if** ($((k'-j')-(j'-i))\geqslant 0$) **then**

从 $G_2$ 的第 $i$ 位置开始插入 $(j'-i)$ 个'-'; $i=k'+1$

**else**

$r=\text{rand}()\%2+1$; $G_{ri}=$'-'; $i++$

**else** //均未找到基因块

$r=\text{rand}()\%2+1$; $G_{ri}=$''; $i++$

重新计算 $|G_1|$ 和 $|G_2|$

**if** ($|G_1|\neq|G_2|$) **then**

在较短的基因序列末插入'-'以使两基因序列长度相同

**图 2** 比对算法

**算法 2**:标记算法(从左至右)

**输入**:$G_1$ 和 $G_2$ 的一个序列对比 $G$

**输出**:$G$ 的一种标记 $l(G)$ 及其代价 $C(G)$

$i=1$; $C(G)=0$; $l(G)=$'';

**while** ($i\leqslant\alpha$) //$\alpha$ 为 $G$ 的列数

**if** ($G_{1i}=G_{2i}$) **then**

$G^*[i]=G_{1i}$; $i++$; //数组 $G^*$ 记录 $G_1$ 和 $G_2$ 的祖先序列;

**else if** ($G_{1i}=\$$ or $G_{2i}=\$$ or $G_{1i}=L$ or $G_{2i}=L$ or $0\leqslant G_{1i}\leqslant\alpha$ or $0\leqslant G_{2i}\leqslant\alpha$)

$i++$;

**else if** ($G_{1i}\neq G_{2i}$ and $G_{1i+1}=G_{2i+1}$) **then**

**if** ($G_{1i}\neq -$ and $G_{2i}=-$) **then**

$P_1=G_1$; $P_2=G_2$;

**else**

$P_1=G_2$; $P_2=G_1$;

**if** (基因 $P_{1i}$ 在 $P_1$ 中出现过) **then**

$P_{2i}=\$$; $i++$; /*标记 $ 表示或者 $P_{1i}$ 是 $P_{1j}$ ($1\leqslant j\leqslant|P_1|$, $i\neq j$) 的复制体,或者 $P_2$ 在第 $i$ 处发生丢失*/

**else** //$P_2$ 在第 $i$ 处发生丢失

$G^*[i]=P_{1i}$; $P_{2i}=$L; $l(G)=l(G)+$L$(P_{2i})$; $i++$; $C(G)++$;

**else** //连续不匹配基因块长度大于 1

**if** ($G_1[i..u]\in\Sigma^{u-i+1}$ and $G_2[i..u]\in\{-\}^{u-i+1}$ ($i\leqslant u\leqslant\alpha$)) **then**

$P_1=G_1$; $P_2=G_2$;

**else**

$P_1=G_2$; $P_2=G_1$;

**if** ($P_1[i..u]=P_1[j..k]$) and 无循环复制[1]) **then**

// ($1<j\leqslant k\leqslant\alpha$, $j>u$ 或 $i>k$)

$P_2[i..u]=\text{index}(P_1[j..k])$; //存 $P_1[j..k]$ 下标

$l(G)=l(G)+$D$(P_1[j..k])$; $C(G)++$; $i=u+1$;

// $P_1[i..u]$ 是复制体,$P_1[j..k]$ 是复制源

**else**

搜索满足如下条件的基因块 $P_1[j..k]\in(\Sigma\cup\{-\})^{k-j+1}$):

1. 移除 $P_1[j..k]$ 中空格后的串 $P_1[j'..k']$ 是 $P_1[i..u]$ 的子串,即 $P_{1j'}=P_{1i}, P_{1j'+1}=P_{1i+1},\cdots,P_{1k'}=P_{1i+(k'-j')}$

2. $P_2$ 上与 $P_1[j..k]$ 中空格对应的基因在 $P_2$ 中都有复制源

**if** (存在满足上述条件的 $P_1[j..k]$ and $|P_1j'..k']|>1$ and 无循环复制[1]) **then**

$P_2[i..i+(k'-j')]=\text{index}(P_1[j'..k'])$; //存 $P_1[j'..k']$ 下标

$l(G)=l(G)+$D$(P_1[j'..k'])$; $C(G)++$; $i=i+(k'-j')+1$;

// $P_1[i..i+(k'-j')]$ 是复制体,$P_1[j'..k']$ 是复制源

**else if** ($P_{1i}$ 在基因序列 $P_1$ 中出现过) **then**

$P_{2i}=\$$; $i++$;

**else**

$G^*[i]=P_{1i}$; $P_{2i}=$L; $l(G)=l(G)+$L$(P_{2i})$; $i++$; $C(G)++$;

**for** ($i=1$; $i\leqslant\alpha$; $i++$) //处理标记 MYM,默认处理为丢失操作

**if** ($G_{1i}=\$$) **then**

$G^*[i]=G_{2i}$; $G_{1i}=$L; $l(G)=l(G)+$L$(G_{1i})$; $C(G)++$;

**else if** ($G_{2i}=\$$) **then**

$G^*[i]=G_{1i}$; $G_{2i}=$L; $l(G)=l(G)+$L$(G_{2i})$; $C(G)++$.

**图 3** 标记算法(从左至右)

### 2.3 遗传算法

本节提出一种求解复制-丢失比对(DLA)问题的遗传算法 G2SP。算法输入为基因序列 $G_1$ 和 $G_2$,输出为 $G_1$ 和 $G_2$ 的一个比对 $G$ 及祖先基因序列 $G^*$。

#### 2.3.1 染色体编码及初始种群

直接采用 $G_1$ 和 $G_2$ 的对比 $G$ 来编码问题解,故染色体编码长度是可变的。种群规模为 $N$,循环执行比对算法(算法 1)以产生初始种群中的每个个体。由于在算法 1 中注入了随机信息,从而保持了初始种群个体的多样性。

#### 2.3.2 选择算子

采用轮盘赌选择策略,以个体的相对适应值作为该个体在下一代中存活的概率,使每个个体都有被选择的机会。

#### 2.3.3 交叉算子

针对该问题编码,采用单点交叉算子。给定父本 $F1$ 和 $F2$,首先在 $F1$ 中随机选择一个位置作为交叉点,将 $F1$ 分割成两个部分;然后在 $F2$ 中找到与 $F1$ 对应位置的交叉点;接着,基于交叉点对 $F1$ 和 $F2$ 实行交换操作,交换不足的地方用字符“-”补上;最后,检测交叉后的每一列,删除取值均为“-”的列,得到子代个体 $C1$ 和 $C2$。如图 4 所示,$F1$ 中加粗列表示交叉点,$F2$ 中加粗列是 $F1$ 中交叉点对应的列,通过单点交叉后得到子代个体 $C1$ 和 $C2$。

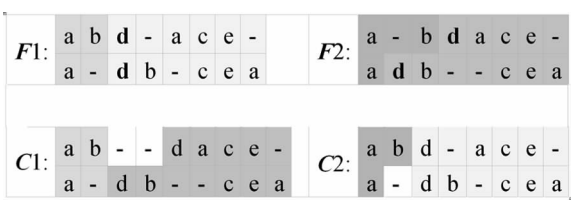


**图 4 交叉算子举例**

#### 2.3.4 变异算子

变异算子模拟生物进化过程中的基因突变,增加了种群的多样性。但是,普通的变异算子由于缺乏任何指导信息,不利于算法的有效收敛。本文根据生物序列进化过程中的一些插入删除特性,提出带有指导性的智能变异算子,在保证种群多样性的基础上,智能变异算子能够使算法快速地收敛到较优解。下面介绍本文提出的 3 种智能变异算子:重新匹配基因块、基因智能移动和移动基因块。下文中,给定父本 $F_{2\times\alpha}$,$F_1$ 和 $F_2$ 分别表示父本 $F$ 中的行。

(1)重新匹配基因块

该算子主要思想是:若父本中存在连续的不匹配基因块,则对这个基因块进行重新匹配。给定父本 $F$ 的连续不匹配基因块($F_1[s..e]$,$F_2[s..e]$)($F_1[s..e]$, $F_2[s..e]\in(\Sigma\cup\{-\})^{e-s+1}$,$s(e)$ 为基因块的起始(终止)下标,$1\leqslant s<e\leqslant\alpha$),删除其中的字符“-”后得到的基因块简记为($F'_1$,$F'_2$);利用算法 1 分别对($F'_1$,$F'_2$)及其逆序块($\overline{F'_1}$,$\overline{F'_2}$)执行比对算法,得到的比对基因块仍记为($F'_1$,$F'_2$)和($\overline{F'_1}$,$\overline{F'_2}$);分别用($F'_1$,$F'_2$)及($\overline{F'_1}$,$\overline{F'_2}$)的逆序块($\overline{\overline{F'_1}}$,$\overline{\overline{F'_2}}$)替换父本 $F$ 中原不匹配基因块($F_1[s..e]$,$F_2[s..e]$),得到两个新个体 $C1$ 和 $C2$,并选择适应度较大的作为父本 $F$ 的子代个体 $C$。图 5 给出了重新匹配基因块变异算子的例子。图 5(a)为父本 $F$,阴影部分表示连续不匹配基因块;图 5(b)表示(a)中连续不匹配基因块去掉字符“-”后的基因块;图 5(c)和图 5(d)分别表示对图 5(b)中基因块及其逆序块执行算法 1 后,替换父本中阴影部分后的结果,且图 5(d)具有较大适应值,作为父本的子代 $C$。

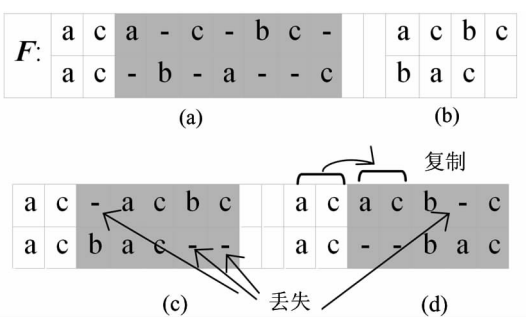


**图 5 重新匹配基因块变异算子举例**

(2) 基因智能移动

该算子主要思想是通过移动父本中的字符“-”以增大新个体的适应度。给定父本 $F$,若存在两个不匹配列$\begin{pmatrix}F_{2i}\\F_{1i}\end{pmatrix}$和$\begin{pmatrix}F_{2j}\\F_{1j}\end{pmatrix}$满足 $F_{1i}$ 匹配 $F_{2j}$(或 $F_{2i}$

匹配 $\boldsymbol{F}_{1j}$),在不改变原基因顺序的前提下,可以通过移动字符"-"使基因 $\boldsymbol{F}_{1i}$ 和 $\boldsymbol{F}_{2j}$(或 $\boldsymbol{F}_{2i}$ 和 $\boldsymbol{F}_{1j}$)被移至同一列,且删除取值全为"-"的列得到新个体。比较新个体和父本个体的适应值,取适应值较大的作为下一代个体。图 6 给出了该算子的一个范例。图 6(a)为父本 $\boldsymbol{F}$,阴影部分表示两个不匹配列;通过移动字符"-",使匹配基因移至同一列,如图 6(b)所示;最后删除取值全为"-"的列得到新个体,通过算子操作,新个体的进化代价减小了 2。

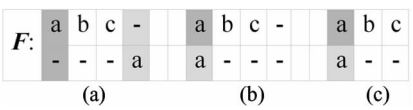


**图 6　基因智能移动变异算子举例**

(3)移动基因块

该算子主要思想是通过移动父本中的基因块以增大新个体的适应度。具体步骤如下:给定父本 $\boldsymbol{F}$,在其中找到一组满足条件 $\boldsymbol{F}_t[i..j] \in \Sigma^{j-i+1}$ 及 $\boldsymbol{F}_{3-t}[i..j] \in \{-\}^{j-i+1}$($t=1, 2$, $1 \leqslant i < j \leqslant \alpha$)的连续不匹配基因块($\boldsymbol{F}_t[i..j]$,$\boldsymbol{F}_{3-t}[i..j]$)。假设 $\boldsymbol{F}_1[i..j] \in \Sigma^{j-i+1}$,$\boldsymbol{F}_2[i..j] \in \{-\}^{j-i+1}$($i<j$),尝试将基因序列 $\boldsymbol{F}_2[1..i-1]$($\boldsymbol{F}_2[j+1..n]$)中的部分基因序列向右(向左)移动,在不改变原基因顺序的前提下,使移动的基因序列能与 $\boldsymbol{F}_1[i..j]$ 形成匹配或部分匹配列,从而得到新个体 $\boldsymbol{C}1$($\boldsymbol{C}2$),并选择适应度较大的作为父本 $\boldsymbol{F}$ 的子代个体 $\boldsymbol{C}$。图 7 给出了该算子的一个范例。图 7(a)为父本 $\boldsymbol{F}$,阴影部分表示连续不匹配基因块;图 7(b)表示 $\boldsymbol{F}_2[7..9]$ 向右移动得到的新个体 $\boldsymbol{C}1$;图 7(c)表示 $\boldsymbol{F}_2[15..16]$ 向左移动得到的新个体 $\boldsymbol{C}2$,且 $\boldsymbol{C}1$ 具有较大适应值,作为父本的子代 $\boldsymbol{C}$。

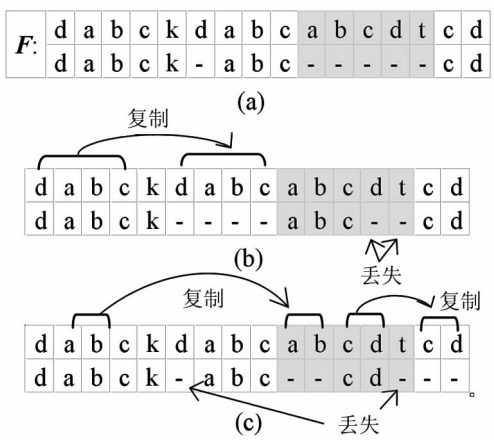


**图 7　移动基因块变异算子举例**

2.3.5　适应度函数

适应度函数是评价种群个体进化程度的标准,是算法演化过程的驱动力,是进行自然选择的惟一依据。给定染色体 $\boldsymbol{G}$,适应度函数 $Fitness(\boldsymbol{G})$ 定义如式

$$Fitness(\boldsymbol{G}) = \frac{1}{C(\boldsymbol{G})} \tag{4}$$

所示,其中 $\boldsymbol{C}(\boldsymbol{G})$ 表示被标记的比对 $\boldsymbol{G}$ 的代价。

根据上述算法设计,G2SP 算法的描述如图 8 所示。

**算法 3**:G2SP 算法

**输入**:基因组序列 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$

**输出**:带标记的比对 $\boldsymbol{G}$,祖先基因序列 $\boldsymbol{G}^*$

**Step0**:算法的基本参数设置:种群规模 $N$,交叉率 $p_c$,变异率 $p_m$,最大演化代数 $maxgen$;

**Step1**:随机产生 $N$ 个个体以生成初始种群 $pop_0$,$gen=0$;

**Step2**:计算 $pop_{gen}$ 中每个个体的适应值,并保存最优个体 $\boldsymbol{G}$;

**Step3**:如果 $gen > maxgen$,转向 **Step7**;否则执行 **Step4**;

**Step4**:利用选择算子从 $pop_{gen}$ 中选择 $(1-p_c)\times N$ 个个体直接放入 $pop_{gen+1}$ 中;选择 $p_c \times N/2$ 对个体,对每对个体采用单点交叉以产生两个后代个体,将所有的后代个体加入 $pop_{gen+1}$ 中;

**Step5**:从 $pop_{gen+1}$ 中按均匀概率选出 $p_m \times N$ 个个体,并对其随机采用重新匹配基因块、基因智能移动或移动基因块变异算子以产生后代个体;

**Step6**:$gen = gen+1$,执行 **Step2**;

**Step7**:根据带标记的对比 $G$,得到祖先基因序列 $G^*$,输出 $\boldsymbol{G}$ 和 $\boldsymbol{G}^*$,算法结束。

**图 8　G2SP 算法**

## 3　实验结果

本文实验数据来源于真实的基因数据,对伪布尔线性规划(PBLP)算法和(G2SP)算法进行了比较分析。PBLP 算法在一台安装了 Ubuntu 12.04 操作系统的联想工作站(Intel(R) Core(TM) i5-3570 CPU @3.40GHz,内存为 4GB)上运行,程序编译器为 Python 2.7.3。G2SP 算法在一台安装了 Windows

XP Professional 操作系统的同样配置的联想工作站上运行,程序编译器为 Microsoft Visual C# 2012。

### 3.1 实验数据

本文使用 4 种菌属的稳定 RNA(tRNA 和 rRNA)基因数据进行实验,基因数据从病害体系资源集成中心(PATRIC)①和美国国立生物技术信息中心(NCBI)②网站下载而来,其中包括 5 个芽孢杆菌、5 个大肠杆菌、10 个链球菌和 11 个葡萄球菌的 tRNA 和 rRNA 数据。为方便描述,本文对各菌属的基因组进行编号,具体情况请参见附录 1,NC 开头的基因序列来源于 NCBI 网站,其余来源于 PATRIC 网站。

### 3.2 性能评价

本文通过进化代价和运行时间两项评价指标对算法 PBLP 和 G2SP 进行比较分析:

**表 1 进化代价比较(芽孢杆菌)**

| 基因组编号组合 | PBLP | G2SP |
|---|---|---|
| {1, 2} | 17 | 9 |
| {1, 3} | 18 | 10 |
| {1, 4} | 64 | 47 |
| {1,5} | 59 | 38 |
| {2,3} | 3 | 2 |
| {2,4} | 50 | 41 |
| {2,5} | 63 | 59 |
| {3,4} | 49 | 40 |
| {3,5} | 62 | 60 |
| {4,5} | 21 | 14 |

**表 2 进化代价比较(大肠杆菌)**

| 基因组编号组合 | PBLP | G2SP |
|---|---|---|
| {1, 2} | 142 | 134 |
| {1, 3} | 166 | 147 |
| {1, 4} | 71 | 59 |
| {1,5} | 90 | 81 |
| {2,3} | 77 | 66 |
| {2,4} | 127 | 118 |
| {2,5} | 118 | 105 |
| {3,4} | 145 | 135 |
| {3,5} | 125 | 106 |
| {4,5} | 109 | 98 |

G2SP 算法的参数设置为 $N = 100$ ,$maxgen = 50$,$p_c = 0.8$,$p_m = 0.2$。为简化描述,基因组组合采用附录 1 中的基因组编号组合代替。例如,芽孢杆菌的基因组编号组合{1, 2}表示基因组 NC _ 004722 和基因组 NC _ 006274 的组合。

表 1 针对 5 种芽孢杆菌设置了 10 个基因组组合,基因组的稳定 RNA 家族数约为 42 左右,每个基因组的稳定 RNAs 数目在 96 – 149 之间。表中数据显示,在各种基因编号组合下,算法 G2SP 均能获得较算法 PBLP 更小的进化代价。表 1 中,由芽孢杆菌 NC _ 006274(编号为 2)与芽孢杆菌 NC _ 007530(编号为 3)的基因组推算其祖先基因组所对应的进化代价最小,而芽孢杆菌 NC _ 007530(编号为 3)与芽孢杆菌 NC _ 000964(编号为 5)的基因组推算其祖先基因组所对应的进化代价最大。因此,在 5 个芽孢杆菌基因组中,NC _ 006274 与 NC _ 007530 基因组之间相对其它组合而言具有较近的亲缘关系,而 NC _ 007530 与 NC _ 000964 基因组之间相对其它组合而言具有较远的亲缘关系。

表 2 针对 5 种大肠杆菌设置了 10 组序列组合,基因组的稳定 RNA 家族数约为 42 左右,每个基因组的稳定 RNAs 数目在 87 – 103 之间。表中数据显示,在各种基因组编号组合下,算法 G2SP 均能获得较算法 PBLP 更小的进化代价。由大肠杆菌 Escherichia _ blattae _ DSM _ 4481(编号为 1)与大肠杆菌 Escherichia _ coli _ TW10509(编号为 4)的基因组推算其祖先基因组所对应的进化代价最小,而大肠杆菌 Escherichia _ blattae _ DSM _ 448(编号为 1)与大肠杆菌 Escherichia _ coli _ O111-H-_ str _ 11128(编号为 3)的基因组推算其祖先基因组所对应的进化代价最大。因此,在 5 个大肠杆菌基因组中, Escherichia _ blattae _ DSM _ 4481 与 Escherichia _ coli _ TW10509 基因组之间相对其它组合而言具有较近的亲缘关系,而 Escherichia _ blattae _ DSM _ 448 与 Escherichia _ coli _ O111-H-_ str _ 11128 基因组之间相对其它组合而言具有较远的亲缘关系。

图 9 和图 10 分别针对 10 种链球菌的 45 种基

① http://patricbrc.vbi.vt.edu/
② http://www.ncbi.nlm.nih.gov/

因序列组合和 11 种葡萄球菌的 55 种基因组组合,对算法 PBLP 和 G2SP 的进化代价进行比较。链球菌基因组的稳定 RNA 家族数为 35 个左右,每个基因组的稳定 RNAs 数目在 80 – 101 之间。葡萄球菌基因组的稳定 RNA 家族数为 36 个左右 ,每个基因组的稳定 RNAs 数目在 72 – 82 之间。各图中,$x$ 和 $y$ 坐标轴表示基因组编号,$z$ 坐标轴表示进化代价。从这两张图中的数据可以看出,在两种菌属的各种基因组编号组合下,算法 G2SP 均能获得较算法 PBLP 更小的进化代价。

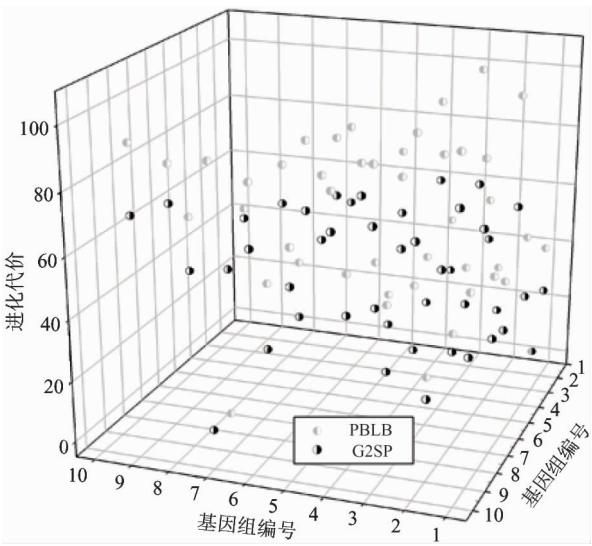


**图 9　进化代价比较(链球菌)**

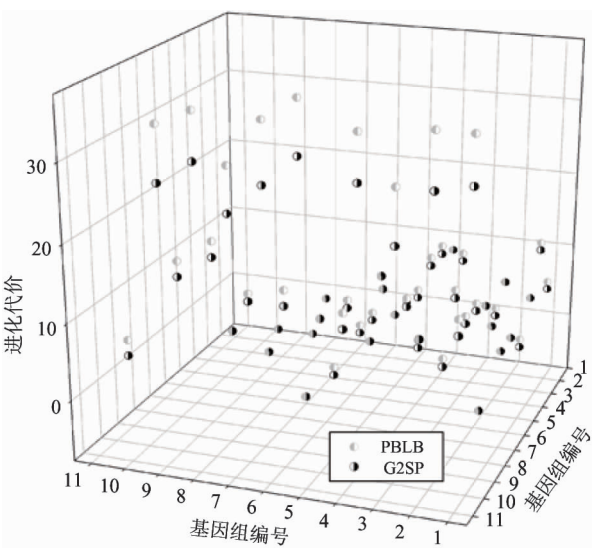


**图 10　进化代价比较(葡萄球菌)**

表 3 给出了各菌属基因组所有组合的平均运行时间,虽然 G2SP 算法的运行时间较 PBLP 算法要长,但是最长时间仅 910.3s,在实际应用中仍然是切实可行的。

为进一步比较分析算法的进化代价,本文选取三组样本来进行显著性测试:第一组为上述 4 个菌种的样本数据,样本大小 $n_s$ 为 120;第二组和第三组采用模拟生成的独立随机样本数据,基因组长度分别是 100 和 200,样本大小 $n_s$ 均为 100,模拟数据采用文献[1]中的生成方法获得。针对 G2SP 算法,$Cost_G$ 表示进化代价的样本均值,$S_G^{Cost}$ 表示进化代价的样本方差,同理为 PBLB 算法设置变量 $Cost_P$ 和 $S_P^{Cost}$。下面给出显著性测试的分析结果。

**表 3　运行时间比较 (s)**

| 菌类基因 | PBLP | G2SP |
|---|---|---|
| 芽孢杆菌 | 58.2 | 164.2 |
| 大肠杆菌 | 11.7 | 910.3 |
| 链球菌 | 58.78 | 329.8 |
| 葡萄球菌 | 1 | 117.4 |

针对进化代价,我们设定无效假设 $H_0$ 为“G2SP 的进化代价均值大于 PBLB 的进化代价均值”,备择假设 $H_1$ 为“PBLB 的进化代价均值大于 G2SP 的进化代价均值”。检验统计量 $Z_{Cost}$ 的计算方法如式

$$Z_{Cost} = \frac{Cost_P - Cost_G}{\sqrt{\frac{S_G^{Cost}}{n_s} + \frac{S_P^{Cost}}{n_s}}} \tag{5}$$

所示,显著性水平 $a$ 设置为 0.01。若 $Z_{Cost} > Z_a = Z_{.01}$,$H_0$ 假设被拒绝,由于 $Z_{.01} = 2.58$,所以拒绝域为 $Z_{Cost} > 2.58$。如表 4 所示,在三种样本条件下,$Z_{Cost}$ 均在拒绝域之内,因此假设 $H_0$ 被拒绝,即 PBLB 的进化代价均值大于 G2SP 的进化代价均值。因此表 4 的显著性测试表明 G2SP 算法在求解 DAL 模型上具有较 PBLB 算法更好的求解性能。

**表 4　进化代价的显著性测试**

| $n_s$ | $Cost_G$ | $Cost_P$ | $S_G^{Cost}$ | $S_p^{Cost}$ | $Z_{Cost}$ |
|---|---|---|---|---|---|
| 120 | 36.50 | 50.88 | 1050.77 | 1790.02 | 2.95 |
| 100 | 20.28 | 42.51 | 3.03 | 59.46 | 28.14 |
| 100 | 40.53 | 85.69 | 4.47 | 84.32 | 48.04 |

从上述实验数据可以看出,G2SP 算法能获得较 PBLP 算法更小的进化代价。下面对 G2SP 算法的特点进行总结:(1) G2SP 算法运用比对算法来获得初始解,同时兼顾了初始解质量及初始种群多样性两方面因素;(2)采用普通算子与智能算子相结合的方式,普通交叉算子为进化过程注入了随机因素,保证了种群的多样性,3 种智能变异算子在进化过程中不断地对问题解进行有指导性的优化,使种群能快速地收敛到较优解。

## 4 结 论

本文对复制-丢失演化模型下两物种小系统发育问题进行了研究,并基于复制-丢失比对问题模型,提出一种有效的求解算法 G2SP。算法 G2SP 通过采用普通算子和智能算子相结合的方式,获得了较好的优化效果。利用 4 种菌属的 tRNA 和 rRNA 基因数据对算法 G2SP 和 PBLP 进行测试。实验结果显示, G2SP 算法能够获得较 PBLP 算法更小的进化代价,且其运行时间在实际应用中是可行的。下一步将对算法 G2SP 的并行性进行研究,以进一步提高算法的效率。

**参考文献**


[1] Holloway P, Swenson K, Ardell D H, et al. Evolution of genome organization by duplication and loss: An alignment approach. *Lecture Notes in Computer Science*, 2012, 7262: 94-112

[2] Dong H, Nilsson L, Kurland C G. Co-variation of tRNA abundance and codon usage in Escherichia coli at different growth rates. *Journal of Molecular Biology*, 2006, 260(5): 649-663

[3] Kanya S, Yamada Y, Kudo Y, et al. Studies of codon usage and tRNA genes of 18 unicellular organisms and quantification of Bacillus subtilis tRNAs: Gene expression level and species-specific diversity of codon usage based on mul-tivariate analysis. *Gene*, 1999, 238(1): 143-155

[4] Kovác J, Brejová B, Vinař T. Populations and Evolution. A New Approach to the Small Phylogeny. http://arxiv.org/abs/1012.0935: Cornell University, 2010

[5] Gupta A, Ma? uch J, Ladislav S, et al. Small Phylogeny Problem: Character Evolution Trees. *Lecture Notes in Computer Science*, 2004, 3109: 230-243

[6] Blomme T, Vandepoele K, Bodt S D, et al. The gain and loss of genes during 600 million years of vertebrate evolution. *Genome Biology*, 2006, 7(5): R43

[7] Cotton J A, Page R D M. Rates and patterns of gene duplication and loss in the human genome. In: Proceedings of the Royal Society B: Biological Sciences, London, England, 2005. 277-283

[8] Wapinski I, Pfeffer A, Friedman N, et al. Natural history and evolutionary principles of gene duplication in fungi. *Nature*, 2007, 449(7158): 54-61

[9] Demuth J P, De B T, Stajich J, et al. The evolution of mammalian gene families. PLoS One, 2006, 1: e85

[10] Dondi R, El-mabrouk N. On the complexity of minimum labeling alignment of two genomes. CoRR abs/1206.1877. 2012


# A genetic algorithm for solving the two-species small phylogeny problem


Wu Jingli, Wang Junwei, Hu Zipeng

(Guangxi Key Laboratory of Multi-source Information Mining & Security, Guangxi Normal University, Guilin 541004)

(College of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004)



**Abstract**

The two-species small phylogeny problem (SPP) in the duplication-loss (D-L) model, called the 2-SPP-DL problem, was studied based on the model of duplication-loss alignment (DLA) problem, and a Genetic algorithm for solving the 2-SPP-DL problem, called the G2SP algorithm model of for short, was presented by introducing an alignment algorithm, a labeling algorithm and three smart mutatation operators. The G2SP algorithm adopts the method of combining the ordinary operator and the smart operators. The general operator maintains the species diversity effectively, while the smart operators improve the population convergence and make species evolve into the optimal solution domain more quickly. The tRNA and rRNA gene data of four kinds of real bacterium were used for algorithm performance test. The experimental results indicate that the G2SP algorithm can get fewer evolution cost than the pseudo-Boolean linear programming (PBLB) algorithm, and it is an effective method for solving the 2-SPP-DL problem.



**Key words**: duplication, loss, two-species small phylogeny problem, alignment, genetic algorithm

# 附录 1

Bacillus（芽孢杆菌）

1 NC _ 004722 Bacillus cereus ATCC 14579 Bacteria
2 NC _ 006274 Bacillus cereus E33L Bacteria
3 NC _ 007530 Bacillus anthracis str. ´Ames Ancestor´ Bacteria
4 NC _ 006322 Bacillus licheniformis ATCC 14580 Bacteria;Firmicutes
5 NC _ 000964 Bacillus subtilis subsp. subtilis str. 168 Bacteria

Escherichia（大肠杆菌）

1 Escherichia _ blattae _ DSM _ 4481
2 Escherichia _ coli _ KTE210
3 Escherichia _ coli _ O111-H-_ str _ 11128
4 Escherichia _ coli _ TW10509
5 Escherichia _ fergusonii _ ATCC _ 35469

Streptococcus（链球菌）

1 Streptococcus _ agalactiae _ A909
2 Streptococcus _ gallolyticus _ subsp _ gallolyticus _ ATCC _ BAA-2069
3 Streptococcus _ agalactiae _ ILRI112
4 Streptococcus _ mutans _ UA159
5 Streptococcus _ pneumoniae _ GA17971
6 Streptococcus _ pyogenes _ MGAS9429
7 Streptococcus _ salivarius _ M18
8 Streptococcus _ sp _ CC94A
9 Streptococcus _ thermophilus _ JIM _ 8232
10 Streptococcus _ urinalis _ 2285-97

Staphylococcus（葡萄球菌）

1 NC _ 002952 Staphylococcus aureus subsp. aureus MRSA252
2 NC _ 002953 Staphylococcus aureus subsp. aureus MSSA476
3 NC _ 005951 Staphylococcus aureus subsp. aureus MSSA476
4 NC _ 006629 Staphylococcus aureus subsp. aureus COLStaphylococcus _ aureus _ COL
5 NC _ 002774 Staphylococcus aureus subsp. aureus Mu50
6 NC _ 002745 Staphylococcus aureus subsp. aureus N315
7 NC _ 003140 Staphylococcus aureus subsp. aureus N315
8 NC _ 007790 Staphylococcus aureus subsp. aureus USA300
9 NC _ 007622 Staphylococcus aureus RF122
10 NC _ 004461 Staphylococcus epidermidis ATCC 12228
11 NC _ 005003 Staphylococcus epidermidis ATCC 12228