高技术通讯 2009 年 第 19 卷 第 2 期:194 ~ 199

doi: 10.3772/j.issn.1002-0470.2009.02.015

一种基于最少片段删除模型重建单体型的粒子群优化算法①

吴璟莉②* ** 陈建二* 王建新*

(* 中南大学信息科学与工程学院 长沙 410083)

(** 广西师范大学计算机科学与信息工程学院 桂林 541004)

摘 要 利用最少片段删除(MFR)模型研究了个体单体型重建的算法。利用单核苷酸多态性(SNP)位点杂合率低的特性,引入了一种短粒子编码方式,提出了一种重建单体型的粒子群优化算法 P-MFR。利用国际人类基因组单体型图计划发布的 CEPH 样本(祖籍是北欧或西欧的美国犹他州人)中 60 个个体在 1 号染色体上的单体型进行实验分析,实验结果显示,与以往求解 MFR 模型的算法相比较,P-MFR 算法能够获得更高重建率的单体型。此外,由于采用了较短的粒子位置编码方式,P-MFR 算法在重建长单体型时仍具有较高的执行效率,有很好的实用价值。

关键词 单核苷酸多态性,单体型,最少片段删除,粒子群优化,编码

0 引 言

单核苷酸多态性(single nucleotide polymorphisms,SNPs)是各种遗传变异中最显著的一种形式,研究 SNPs 对阐明疾病易感性机制、设计个体化治疗方案和药物研制具有重要意义和实际应用价值[1]。然而,检测人类染色体上所有 1000 万个常见 SNPs 的费用极其昂贵,所幸的是,由于连锁不平衡现象以及缺乏重组事件,一些相邻的多态位点趋于在一起共同遗传,这些变异连锁的区域即为单体型。最近的研究表明,在与疾病相关的研究中,单体型数据通常比单个 SNP 携带更多的信息[2]。但在当前的实验技术下,直接通过生物学实验手段测定单体型既费钱又费时,因此通常利用计算的方式来获得单体型。单体型检测问题主要分为两类:群体单体型检测,也称为单体型推断[3,4];个体单体型检测,也称为单体型重建[5]。本文主要对后者进行研究。

2001 年,Lancia 等首次将单体型重建问题形式化成一个优化问题[5]。DNA 测序错误及人类染色体的二倍性使这个问题变得非常复杂。针对来自于样本污染的错误类型,即样本中混入了其它个体的 DNA 片段[1],文献[5]引入了最少片段删除(minimum fragment removal,MFR)模型。当每条片段至多存在 k 个洞(片段中的空值位点)且 k 值较小时,该模型是多项式可解的[6]。Rizzi 等首先提出时间复杂度为 $O(2^{2k}m^2n+2^{3k}m^3)$ 的动态规划算法[6],其中 m 为 DNA 片段总数,n 为 SNP 位点总数。Xie 等通过引入小参数 k_1 和 k_2,提出了时间复杂度为 $O(mk_1k_2 2^k+2^{3k}mk_2^2+m\log m+nk_2+mk_1)$ 的参数化动态规划算法[7]。而在一般情况下,该模型是 NP 难题[5],Panconesi 等提出启发式算法 Fast Hare,它能对模型的一般情况进行处理,并获得较动态规划算法更高重建率的单体型[8]。

本文提出一种基于 MFR 模型重建单体型的粒子群优化算法 P-MFR,它在片段有空隙或无空隙情况下均适用。利用 SNP 位点杂合率低的特性,本文为 P-MFR 算法设计了一种短粒子编码方式。实验结果显示,与以往求解 MFR 模型的算法相比较,P-MFR 算法能够获得更高重建率的单体型。此外,由于采用了较短的粒子位置编码方式,P-MFR 算法在重建长单体型时仍具有较高的执行效率,有很好的实用价值。

1 问题及符号定义

人类等二倍体生物的 DNA 序列是按染色体成对出现的。对于任意一个 SNP 位点,若其在一对同源染色体上的碱基相同,称为纯合(homozygous)位点,否则称为杂合(heterozygous)位点。几乎所有 SNP

① 国家自然科学基金重点项目:生物信息学中的相关组合理论和算法研究(60433020),新世纪优秀人才支持计划(NCET-05-0683),长江学者和创新团队发展计划(IRT0661)资助项目。

② 女,1978 年生,博士生;研究方向:生物信息学,E-mail:wjlhappy@mailbox.gxnu.edu.cn

(收稿日期:2008-04-24)

位点上的碱基都只有两种取值,因此单体型数据可以用定义在二元字符集{0,1}上的字符序列来表示,而不必用真正的碱基字符。给定 m 条来自某对同源染色体的 SNP 片段,其对应的单体型长度为 n,由此定义一个 $m \times n$ 的 SNP 矩阵 $\boldsymbol{M}$,其中每个元素 $\boldsymbol{M}[i, j]$的值为 0、1 或 -,- 表示片段在该位点的取值未知,矩阵 $\boldsymbol{M}$ 中每一行表示一条 SNP 片段,每一列表示一个 SNP 位点。

下面给出本文采用的符号定义:

在矩阵 $\boldsymbol{M}$ 中,令 $n_x(j)$为第 j 列中值为 x 的元素个数,令 $f_x(j)$为第 j 列中值为 x 的元素在该列所有非空元素中所占的比例,即 $f_0(j) = n_0(j)/(n_0(j) + n_1(j))$,$f_1(j) = n_1(j)/(n_0(j) + n_1(j))$。给定变量 $x, y \in \{0, 1, -\}$,$s(x, y)$和 $d(x, y)$分别定义如下:

$$s(x, y) = \begin{cases} 1, \text{若 } x \neq -, y \neq -, \text{且 } x = y \\ 0, \text{否则} \end{cases} \quad (1)$$

$$d(x, y) = \begin{cases} 1, \text{若 } x \neq -, y \neq -, \text{且 } x \neq y \\ 0, \text{否则} \end{cases} \quad (2)$$

对于两条字符串 $\boldsymbol{U} = u_1, \cdots, u_n$ 和 $\boldsymbol{V} = v_1, \cdots, v_n$,且 $u_i, v_i \in \{0,1, -\}$,$S(\boldsymbol{U}, \boldsymbol{V})$和 $D(\boldsymbol{U}, \boldsymbol{V})$定义为

$$S(\boldsymbol{U}, \boldsymbol{V}) = \sum_{i=1}^{n} s(u_i, v_i) \quad (3)$$

$$D(\boldsymbol{U}, \boldsymbol{V}) = \sum_{i=1}^{n} d(u_i, v_i) \quad (4)$$

这里 S($\boldsymbol{U}$, $\boldsymbol{V}$)表示字符串 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 对应位取值相同(同为 1 或同为 0)的位数,$D(\boldsymbol{U}, \boldsymbol{V})$表示 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 对应位取值相异(一个为 1 则另一个为 0)的位数。将字符串 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 看成两条 SNP 片段,当 $D(\boldsymbol{U}, \boldsymbol{V})$大于 0 时,表示片段 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 是冲突的,否则它们是相容的。若两条 SNP 片段之间相互冲突,意味着它们分别来自于两条染色单体或者片段数据中存在测序错误。如果所有片段数据均没有测序错误,则矩阵 $\boldsymbol{M}$ 中的行可以分成两个不相交的子集,每个子集中的所有行相容且决定一条单体型,这时矩阵 $\boldsymbol{M}$ 被称为是可行的。

Lancia 等在文献[5]中提出下述单体型重建问题的优化模型:

最少片段删除(MFR):给定一个 SNP 矩阵 $\boldsymbol{M}$,删除最少的行(片段)使得 SNP 矩阵可行。

假设矩阵 $\boldsymbol{M}$ 中的“错误”行已经被删除,得到可行矩阵 $\boldsymbol{M}'$。将 $\boldsymbol{M}'$ 的行分成两个互不相交的子集,且 $\boldsymbol{G} = \{\boldsymbol{M}'[i_1, -], \cdots, \boldsymbol{M}'[i_k, -]\}$为其中的一个行子集,即 $\boldsymbol{G}[j, -] = \boldsymbol{M}'[i_j, -]$,这里$\boldsymbol{M}'[i_j, -]$表示矩阵 $\boldsymbol{M}'$中第 $i_j (j = 1, \cdots, k)$行。由 $\boldsymbol{G}$ 中片段构建的单体型表示为$\boldsymbol{h}(\boldsymbol{G}) = h_1(\boldsymbol{G}), \cdots, h_n(\boldsymbol{G})$,其中

$$h_c(\boldsymbol{G}) = \begin{cases} 0, \text{若 } N_0(\boldsymbol{G}, c) > N_1(\boldsymbol{G}, c), c = 1,2, \cdots, n \\ 1, \text{否则} \end{cases} \quad (5)$$

这里 $N_0(\boldsymbol{G}, c)$和 $N_1(\boldsymbol{G}, c)$分别表示集合$\{\boldsymbol{G}[1, c], \cdots, \boldsymbol{G}[k, c]\}$中 0 和 1 的个数。

在文中,我们使用重建率[9](reconstruction rate, RR)来衡量重建单体型的正确度,假设 $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2)$为一对真实单体型,$\hat{\boldsymbol{h}} = (\hat{\boldsymbol{h}}_1, \hat{\boldsymbol{h}}_2)$为一对重建单体型,$RR$ 表示单体型 $\hat{\boldsymbol{h}}$ 中正确构建的核苷酸比例,定义如下:

$$RR(\boldsymbol{h}, \hat{\boldsymbol{h}}) = 1 - \frac{\min\{r_{11} + r_{22}, r_{12} + r_{21}\}}{2n} \quad (6)$$

其中 $r_{ij} = D(\boldsymbol{h}_i, \hat{\boldsymbol{h}}_j)(i, j = 1,2)$。

2 P-MFR 算法

算法 P-MFR 的输入为一个 $m \times n$ 的 SNP 矩阵 $\boldsymbol{M}$ 和参数 t,输出为一对长度为 n 的单体型 $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2)$。算法首先对矩阵 $\boldsymbol{M}$ 进行预处理,删除矩阵中的冗余信息;然后利用粒子群算法[10]进行求解,生成一个最优解 $\boldsymbol{h}' = (\boldsymbol{h}'_1, \boldsymbol{h}'_2)$,它表示一对只含杂合位点的单体型。最后,算法对单体型 $\boldsymbol{h}'$ 进行扩展,以得到最终结果 $\boldsymbol{h}$。下面将分别介绍算法 P-MFR 中的几个关键步骤。

(1)预处理。为了更有效地求解问题,首先对矩阵 $\boldsymbol{M}$ 进行预处理。对每个列 j,如果满足条件 $f_0(j) \leqslant t$ 或$f_1(j) \leqslant t$,这里 t 设置为 0.2[8],则将该列从矩阵 $\boldsymbol{M}$ 中删除并称其为 1-列或 0-列。将所有满足上述条件的列删除之后,某些行将变成空行(元素值全为-),它们对于重建工作没有任何帮助,因此也将其删除。预处理后保留下来的 SNP 位点均为杂合位点,得到的新矩阵记为 $\boldsymbol{M}1_{m1 \times n1}$。下文为方便描述,仍用 $\boldsymbol{M}_{m \times n}$表示新矩阵。

(2)粒子群优化。粒子群优化(particle swarm optimization, PSO)于 1995 年由 Eberhart 和 Kennedy 首次提出[10],它通过模拟鸟群捕食的行为来完成一个自演化系统。PSO 首先随机初始化一群粒子(随机解),然后通过迭代搜索最优解。在每次迭代中,粒子在解空间中运动,并通过其个体极值与群体极值更新自身的速度和位置[11],如公式

$$\boldsymbol{V}_i(T+1) = w \times \boldsymbol{V}_i(T) + C_1 \times rand_1 \times (\boldsymbol{P}_i - \boldsymbol{X}_i(T)) + C_2 \times rand_2 \times (\boldsymbol{P}_g - \boldsymbol{X}_i(T)) \quad (7)$$

$$X_i(T+1) = X_i(T) + V_i(T+1) \quad (8)$$

所示。其中，$\boldsymbol{X}_i$ 表示第 i 个粒子的位置，其速度为 $\boldsymbol{V}_i$，T 表示迭代次数；w 为惯性因子；$\boldsymbol{P}_i$ 表示第 i 个粒子从搜索初始到当前迭代所对应的个体极值；$\boldsymbol{P}_g$ 表示从搜索初始到当前迭代所对应的群体极值；C_1 和 C_2 为正常数；$rand_1$ 和 $rand_2$ 是介于(0, 1) 之间的随机数。下面给出求解单体型重建问题的粒子群算法设计。

① 粒子表示：P-MFR 算法采用二进制串 $\boldsymbol{X}(x_1, x_2, \cdots, x_n)$ $(x_i \in \{0,1\})$ 和 $V(v_1, v_2, \cdots, v_n)$ $(v_i \in \{0,1\})$ 来分别表示一个粒子的位置和速度。粒子位置代表一条只含杂合位点的单体型。如前所述，新矩阵中的片段只保留了杂合位点，则由它们构建的单体型必定也只具有杂合位点。由于一对单体型在其杂合位点上的值是相异的(一个值为 0(1)，另一个值则为 1(0))，因此对于这样一对只具有杂合位点的单体型，可以通过其中一条推导出另一条。所以，由一个粒子的位置可以惟一决定一对只含杂合位点的单体型。

② 初始粒子群：粒子群由 N 个粒子组成，即群体规模为 N。随机初始化粒子群，即初始化群中所有粒子的速度和位置。初始速度为随机生成的一个二进制串。初始位置的生成方式如下：随机将新矩阵 $\boldsymbol{M}$ 中的片段分成两个集合，并根据公式(5) 生成一对单体型(只含杂合位点)，且任选其中一条作为粒子的初始位置。

③ 粒子的速度表示及粒子间的运算操作定义：

(a) 粒子的速度 $\boldsymbol{V}$ 定义为两个位置 $\boldsymbol{X}_1(x_{11}, x_{12}, \cdots, x_{1n})$ 和 $\boldsymbol{X}_2(x_{21}, x_{22}, \cdots, x_{2n})$ 之间的距离。

$$\boldsymbol{V} = \boldsymbol{X}_1 - \boldsymbol{X}_2 = (v_1, \cdots, v_n) \quad (9)$$

$$v_i = \begin{cases} 0, x_{1i} = x_{2i} \\ 1, x_{1i} \neq x_{2i} \end{cases} i = 1,2,\cdots,n \quad (10)$$

(b) 速度 $\boldsymbol{V}_1(v_{11}, v_{12}, \cdots, v_{1n})$ 和 $\boldsymbol{V}_2(v_{21}, v_{22}, \cdots, v_{2n})$ 间的加法操作定义为其相应位的逻辑加，结果为速度 $\boldsymbol{V}$。

$$\boldsymbol{V} = \boldsymbol{V}_1 + \boldsymbol{V}_2 = (v_1, \cdots, v_n)$$
$$v_i = v_{1i}\ OR\ v_{2i},\ i = 1,2,\cdots,n \quad (11)$$

(c) 粒子速度 $\boldsymbol{V}_1$ 与概率 C 的乘积，结果为速度 $\boldsymbol{V}$。

$$\boldsymbol{V} = C \times \boldsymbol{V}_1 = (v_1, \ldots, v_n) \quad (12)$$

$$v_i = \begin{cases} 1 - v_{1i}, \text{若 } 0.5 \leqslant C \leqslant 1 \\ v_{1i}, \text{否则} \end{cases}, \quad i = 1,2,\cdots,n \quad (13)$$

(d) 速度 $\boldsymbol{V}$ 和位置 $\boldsymbol{X}_1(x_{11}, x_{12}, \cdots, x_{1n})$ 间的加法操作定义为其相应位的逻辑异或，结果为位置 $\boldsymbol{X}$。

$$\boldsymbol{X} = \boldsymbol{X}_1 + \boldsymbol{V} = (x_1, \cdots, x_n)$$
$$x_i = x_{1i}\ \ XOR\ \ v_i,\ \ i = 1,2,\cdots,n \quad (14)$$

④ 适应度函数：适应度函数用于评价粒子的搜索性能，指导粒子群的搜索过程。给定某个粒子位置 $\boldsymbol{X}$ 及新矩阵 $\boldsymbol{M}$ 中的所有片段 $\boldsymbol{f}_i$ $(i = 1, \cdots, m)$，$\boldsymbol{X}$ 的适应度函数 $Fitness$ $(\boldsymbol{X})$ 定义为：

$$Fitness(\boldsymbol{X}) = \frac{1}{R(\boldsymbol{X})} \quad (15)$$

$$R(\boldsymbol{X}) = |\{i \mid D(\boldsymbol{f}_i, \boldsymbol{X}) \neq 0, S(\boldsymbol{f}_i, \boldsymbol{X}) \neq 0, i = 1,2,\cdots,m\}| \quad (16)$$

其中，粒子位置 $\boldsymbol{X}$ 表示一对仅含杂合子的单体型对 $(\boldsymbol{h'}_1, \boldsymbol{h'}_2)$ 中的一条，例如 $\boldsymbol{h'}_1$。于是 $S(\boldsymbol{f}_i, \boldsymbol{X})$ 表示片段 $\boldsymbol{f}_i$ 与单体型 $\boldsymbol{h'}_1$ 间等位基因相同的位点个数，即片段 $\boldsymbol{f}_i$ 与单体型 $\boldsymbol{h'}_2$ 间等位基因相异的位点个数；$D(\boldsymbol{f}_i, \boldsymbol{X})$ 表示片段 $\boldsymbol{f}_i$ 与单体型 $\boldsymbol{h'}_1$ 间等位基因相异的位点个数；$R(\boldsymbol{X})$ 表示对应于单体型对 $(\boldsymbol{h'}_1, \boldsymbol{h'}_2)$ 的最少片段删除个数。

根据上述算法设计，求解单体型重建问题的粒子群优化算法如图 1 所示。

粒子群算法
输入：预处理后的新 SNP 矩阵 $\boldsymbol{M}$，w，C_1，C_2，N，*iteration _ times*；
// *iteration _ times* 为算法的最大迭代次数
输出：一对单体型 $\boldsymbol{h'} = (\boldsymbol{h'}_1, \boldsymbol{h'}_2)$
步骤 1. 初始化粒子群，$T = 0$
步骤 2. 如果 $T <$ *iteration _ times*，转向步骤 3；否则，执行步骤 5.
步骤 3. 对群中的每个粒子 $\boldsymbol{X}_i(T)$ $(i = 1, \cdots, N)$，执行如下操作：
(a)根据公式(15)计算 $\boldsymbol{X}_i(T)$ 的适应值；
(b)如果 $\boldsymbol{X}_i(T)$ 的适应值大于 $\boldsymbol{P}_i$ 的适应值，则 $\boldsymbol{P}_i = \boldsymbol{X}_i(T)$；
(c)如果 $\boldsymbol{X}_i(T)$ 的适应值大于 $\boldsymbol{P}_g$ 的适应值，则 $\boldsymbol{P}_g = \boldsymbol{X}_i(T)$；
(d)根据公式(7)和(8)进行粒子速度及位置的迭代.
步骤 4. T 加 1，转向步骤 2.
步骤 5. 将 $\boldsymbol{P}_g$ 转换成单体型对 $\boldsymbol{h'} = (\boldsymbol{h'}_1, \boldsymbol{h'}_2)$ 并输出 $\boldsymbol{h'}$，算法运行结束

图 1 粒子群算法

(3) 扩展结果。由于预处理时将单体型中的同合 SNP 位点删掉了，所以最后要将其重新加回来。对于只含杂合位点的单体型 $\boldsymbol{h'} = (\boldsymbol{h'}_1, \boldsymbol{h'}_2)$，如果某个已删除的同合位点为 0-列(1-列)，则将 0(1)插回到单体型 $(\boldsymbol{h'}_1, \boldsymbol{h'}_2)$ 的相应位置，以此得到扩展后的单体型 $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2)$。依据以上叙述，P-MFR

的算法步骤如图 2 所示。

P-MFR 算法
输入:$m \times n$ *SNP* 矩阵 $\boldsymbol{M}$, 参数 t
输出:一对单体型 $\boldsymbol{h}=(\boldsymbol{h}_1,\boldsymbol{h}_2)$
步骤 1. 预处理矩阵 $\boldsymbol{M}$, 得到新 SNP 矩阵 $\boldsymbol{M}_1$
步骤 2. 执行粒子群优化算法, 得到单体型对 $\boldsymbol{h}'=(\boldsymbol{h}'_1,\boldsymbol{h}'_2)$
步骤 3. 对单体型对 $\boldsymbol{h}'$ 进行扩展操作,得到单体型对 $\boldsymbol{h}$, 并将其输出

图 2 P-MFR 算法

3 实验结果

本文利用真实的单体型数据来进行实验测试。实验在一台安装了 Windows XP Professional 操作系统的 IBM 工作站(Intel Pentium IV 2.0GHz, 内存为 512MB)上进行,程序编译器为 Microsoft Visual C++ 6.0。

3.1 实验数据

本文实验采用的单体型数据来自于国际人类基因组单体型图计划(The International HapMap Project)[12]2007 年 12 月发布的数据文件 genotypes _ chr1 _ CEU _ r22 _ nr.b36 _ fwd _ phase.gz (http://www.hapmap.org/downloads/phasing/2007-08 _ rel22/phased/下载而来),该文件中包含了 CEPH 样本(祖籍是北欧或西欧的美国犹他州人)中 60 个个体在 1 号染色体上的单体型,每个单体型有 193554 个 SNP 位点,本文实验随机选择一个个体指定长度的一对单体型。下面采用两种方法来生成片段,且分别称其为 Fast Hare 实例和 Celsim 实例。

Fast Hare 实例的生成方式如下[8]:将每条单体型复制成 c 个副本,并且将每个副本随机打断成大约 $n/5$ 条片段,使每条片段的长度控制在 3 到 7 之间。最后为每条片段模拟植入测序错误,即对每条片段中的每一位,根据概率 P_s 将其值变换(由 0 变为 1,或由 1 变为 0,或置为空值 -)。这些参数的实际值通常为:$c=5\sim10$; $P_s\in[0.02,0.05]$[8]。

Celsim 实例用著名的鸟枪法测序模拟片段生成器 Celsim[13]生成。每组实例生成 $n\times c/5$ 条单片段(非 mate-pair)和 $10\times c$ 条 mate-pair 片段,且每条 mate-pair 片段由两条来自于同一条单体型的单片段组成。每条单片段的长度控制在 3 到 7 之间,mate-pair 片段长度设置为 $n/10$,植入错误率为 P_s。

3.2 性能评价

我们用运行时间和重建率两个指标来比较分析算法 Fast Hare 和 P-MFR. 每组设置参数生成 100 组实例,每个计算结果均为 100 次重复测试的平均值。P-MFR 算法的运行参数为:$N=20$, *iteration _ times* = 100, $w=0.8$, $C_1=C_2=0.7$。

表 1 到表 3 利用 Fast Hare 实例来比较这两个算法。在表 1 中,针对错误率 P_s 生成 7 组参数,P_s 变化范围为 0 到 0.1, $c=5$, $n=100$。从表 1 中可以看出,在各种 P_s 值设置下,P-MFR 算法的重建率比 Fast Hare 要高,且它们的重建率随 P_s 的增加慢慢下降。在表 2 中,针对片断覆盖率 c 生成了 9 组参数,且 $n=100$, $P_s=0.05$,随着 c 的增加,P-MFR 算法的

表 1 算法的重建率比较($c=5, n=100$)

P_s	*RR*	
	Fast Hare	P-MFR
0	0.9800	0.9996
0.01	0.9600	0.9994
0.02	0.9400	0.9993
0.03	0.9388	0.9987
0.04	0.9373	0.9898
0.05	0.9345	0.9863
0.1	0.9148	0.9564

表 2 算法的重建率比较($n=100$, $P_s=0.05$)

c	*RR*	
	Fast Hare	P-MFR
2	0.8250	0.9054
3	0.9300	0.9837
4	0.9330	0.9855
5	0.9345	0.9863
6	0.9400	0.9938
7	0.9455	0.9998
8	0.9500	0.9993
9	0.9536	0.9998
10	0.9595	0.9999

表 3 算法的重建率的比较($c=5, P_s=0.05$)

n	*RR*	
	Fast Hare	P-MFR
100	0.9345	0.9863
200	0.9450	0.9917
300	0.9367	0.9923
500	0.9168	0.9865
800	0.9084	0.9870
1000	0.9132	0.9867

重建率增长速度比 Fast Hare 要快。表 3 针对单体型长度 n 设置参数，且 $c=5$，$P_s=0.05$，从表中可以看出单体型长度对算法的重建率影响不大。

图 3 和图 4 利用 Celsim 实例来测试这两个算法的重建率。在图 3 中，x 坐标轴表示覆盖率 c，y 坐标轴表示错误率 P_s，z 坐标轴表示重建率 RR。在图 4 中，x 坐标轴表示覆盖率 c，y 坐标轴表示单体型长度 n，z 坐标轴表示重建率 RR。从这两个图中可以看出，P-MFR 获得的重建率较 Fast Hare 要高，且随着 c 的不断增大和 P_s 的不断减小，P-MFR 的优越性越来越明显。

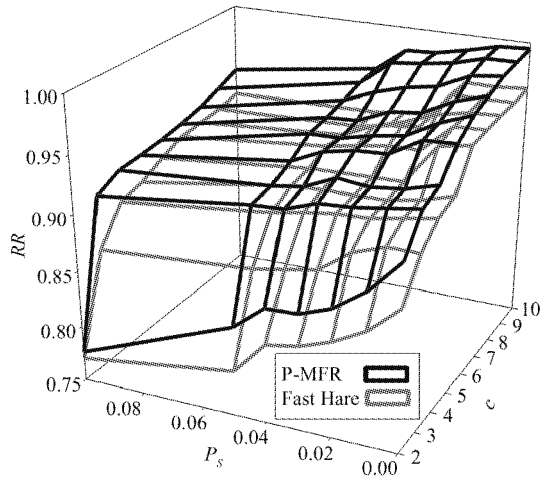

图 3 算法在不同 c 和 P_s 下的重建率比较（$n=100$）

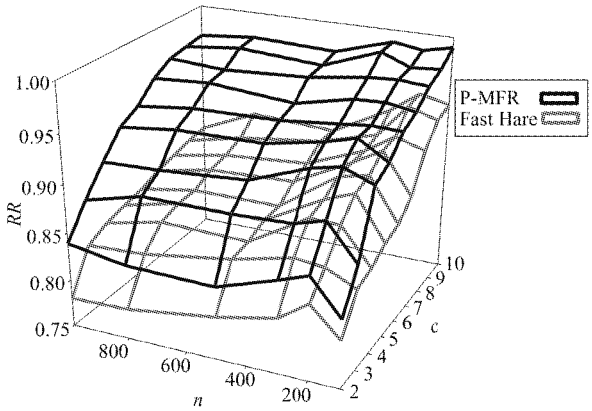

图 4 算法在不同 c 和 n 下的重建率比较（$P_s=0.05$）

实验结果显示 P-MFR 算法能够获得较 Fast Hare 算法更高的单体型重建率。粒子群算法是一种群体智能优化算法，其性能与解空间大小紧密相关，而解空间大小取决于粒子编码长度。P-MFR 算法的粒子编码长度等于一条单体型的杂合 SNP 位点个数，大约为 $n\times d$。由于 SNP 位点的实际杂合率较低，即 d 约为 0.2[8]，这使得 P-MFR 算法的粒子编码较短，其对应的解空间也较小。因此，这对 P-MFR 算法能够获得好结果起了非常积极的作用。

表 4 和表 5 利用 Fast Hare 实例来测试两个算法的运行速度。表 4 中，针对片段覆盖率 c 生成了 9 组参数，c 变化范围为 2 到 10，且 $n=100$，$P_s=0.05$。

这两个算法的运行时间随 c 的增加而逐步增加。表 5 针对单体型长度 n 设置了 6 组参数，且 $c=5$，$P_s=0.05$，从表中可以看出这两种算法的运行时间随着单体型长度的增加也逐步增加。

表 4 运行时间的比较（$n=100$，$P_s=0.05$）

c	运行时间(s)	
	Fast Hare	P-MFR
2	0.0005	0.1427
3	0.0008	0.1461
4	0.0017	0.1833
5	0.0025	0.2416
6	0.0066	0.2919
7	0.0070	0.3881
8	0.0118	0.6515
9	0.0127	0.7710
10	0.0142	0.9112

表 5 运行时间的比较（$c=5$，$P_s=0.05$）

n	运行时间(s)	
	Fast Hare	P-MFR
100	0.0025	0.2416
200	0.0294	0.3467
300	0.0392	0.6850
C500	0.1875	3.1600
800	0.3308	5.1242
1000	0.5416	7.9162

从这两个表中可以看出 Fast Hare 算法的运行速度很快，最长运行时间不超过 0.6s。尽管 P-MFR 的运行时间比 Fast Hare 要长，它在重建长单体型时也只需要几秒钟。P-MFR 的运行时间由粒子的编码长度、粒子群规模和迭代次数三个参数所决定，这三个参数越大，则运行时间越长。其中粒子群规模和迭代次数这两个参数的大小取决于粒子的编码长度。如前所述，本文所设计的粒子编码长度为 $n\times d$，由于 d 约为 0.2[8]，即便当单体型长度 n 为 1000 时，粒子编码仍然较短，因此 P-MFR 算法能够在使用小规模粒子群和较少的迭代次数下获得好结果，具有较快的运行速度。

4 结 论

单体型检测问题已成为计算生物学最热门的领域之一。本文提出一种基于粒子群优化的 P-MFR 算法,并利用国际人类基因组单体型图计划发布的 CEPH 样本中 60 个个体在 1 号染色体上的单体型数据对其性能进行实验分析。P-MFR 算法在片段有空隙或无空隙情况下均适用。由于它采用了一种短粒子编码方式,因此它能产生一个较小的解空间,从而能快速地获得好的结果。实验结果表明,与以往求解 MFR 模型的算法相比较,P-MFR 算法能够获得更高重建率的单体型。此外,P-MFR 算法在重建长单体型时仍具有较高的执行效率。因此,P-MFR 算法是求解个体单体型重建问题的一个可行方法,具有很好的实用价值。

参考文献

[1] Bafna V, Istrail S, Lancia G, et al. Polynomial and APX-hard cases of the individual haplotyping problem. *Theoretical Computer Science*, 2005, 335: 109-125

[2] Stephens J C, Schneider J A, Tanguay D A, et al. Haplotype variation and linkage disequilibrium in 313 human genes. *Science*, 2001, 293: 489-493

[3] Clark A G. Inference of haplotypes from PCR-amplified samples of diploid populations. *Mol Biol Evol*, 1990, 7: 111-122

[4] Gusfield D. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *J Comput Biol*, 2001, 8: 305-323

[5] Lancia G, Bafna V, Istrail S, et al. SNPs problems, complexity and algorithms. In: Proceedings of the 9th Annual European Symposium on Algorithms. London: Springer-Verlag, 2001. 182-193

[6] Rizzi R, Bafna V, Istrail S, et al. Practical algorithms and fixed parameter tractability for the single individual SNP haplotyping problem. In: Proceedings of the 2nd International Workshop on Algorithms in Bioinformatics. London: Springer-Verlag, 2002. 29-43

[7] Xie M Z, Chen J E, Wang J X. Research on parameterized algorithms of the individual haplotyping problem. *J Bioinform Comput Biol*, 2007, 5: 795-816

[8] Panconesi A, Sozio M. Fast hare: a fast heuristic for single individual SNP haplotype reconstruction. In: Proceedings of the 4th Workshop on Algorithms in Bioinformatics. Heiderberg: Springer-Verlag, 2004. 266-277

[9] Wang R S, Wu L Y, Li Z P, et al. Haplotype reconstruction from SNP fragments by minimum error correction. *Bioinformatics*, 2005, 21: 2456-2462

[10] Kennedy J, Eberhart R C. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. New Jersey: IEEE, 1995. 1942-1948

[11] Kennedy J, Eberhart R C. Swarm Intelligence. San Francisco: Morgan Kaufmann, 2001. 165-178

[12] Consortium T I H. The international HapMap project. *Nature*, 2003, 426(6968):789-796

[13] Myers G. A dataset generator for whole genome shotgun sequencing. In: Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology. California: AAAI Press, 1999. 202-210

A particle swarm optimization algorithm for haplotype reconstruction based on minimum fragment removal model

Wu Jingli[*,**], Chen Jian'er[*], Wang Jianxin[*]

([*] School of Information Science and Engineering, Central South University, Changsha 410083)

([**] Department of Computer Science, Guangxi Normal University, Guilin 541004)

Abstract

The individual haplotype reconstruction problem was studied by using the minimum fragment removal (MFR) model. Owing to the NP-hardness of the MFR model, a practical algorithm based on particle swarm optimization (PSO) for haplotype reconstruction, named P-MFR, was presented. A kind of short particle code was designed for the P-MFR by taking advantage of the low heterozygous frequencies of single nucleotide polymorphisms (SNPs). The experiments were conducted by using the haplotypes on the chromosomes 1 of 60 individuals in the CEPH sample, which were released by the International HapMap Project. The results indicate that P-MFR can obtain higher reconstruction rate than previous algorithms when solving the MFR model. Moreover, this kind of short particle code makes P-MFR efficient even for reconstructing long haplotypes.

Key words: single nucleotide polymorphisms (SNPs), haplotype, minimum fragment removal (MFR), particle swarm optimization (PSO), code