博士学位论文

单体型组装问题参数化建模及算法研究

作 者 姓 名 ： 谢民主

学 科 专 业 ： 计算机应用技术

学院（系、所）：信息科学与工程学院

指 导 教 师 ： 陈建二教授

王建新教授

中 南 大 学

二〇〇八年五月

分类号 TP393　　学号 043310030

V D C　　密级

博士学位论文

单体型组装问题参数化建模及算法研究

Research on Parameterized Models and Algorithms for the Haplotype Assembly Problem

作者姓名：　谢民主

学科专业：　计算机应用技术

学院（系、所）：信息科学与工程学院

指导教师：　陈建二教授

副指导教师：　王建新教授

论文答辩日期　　答辩委员会主席

中 南 大 学

二〇〇八年五月

Central South University

Research on Parameterized Models and Algorithms for the Haplotype Assembly Problem

A Dissertation Submitted for

the Degree of Doctor of Philosophy

By

Minzhu XIE

Supervisor: Professor Jianer CHEN

Professor Jianxin WANG

May 2008

原创性声明

本人声明，所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了论文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得中南大学或其他单位的学位或证书而使用过的材料。与我共同工作的同志对本研究所作的贡献均已在论文中作了明确的说明。

作者签名：_____ 日期：_____年___月___日

学位论文版权使用授权书

本人了解中南大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并根据国家或湖南省有关部门规定送交学位论文，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以采用复印、缩印或其它手段保存学位论文。同时授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。

作者签名：_____ 导师签名_____ 日期：____年__月__日

摘　要

分析和识别单体型对复杂疾病致病基因的精确定位有重要作用。单体型组装问题是利用个体 DNA 测序片段数据推出该个体一对单体型的计算问题。根据不同的优化准则，单体型组装问题有 MSR、MFR、MEC 和 MEC/GI 等计算模型。单体型组装问题的绝大部分计算模型都被证明是 NP-难的，缺乏实用的精确算法。

在实际 DNA 片段数据中，一个片段所覆盖的最大 SNP 位点数 k_1 通常小于 10，覆盖一个 SNP 位点的最大片段数 k_2 通常不大于 19。基于以上事实，本文对 MSR 和 MFR 进行参数化建模。在此基础上，为求解无空隙的 MSR 和 MFR，本文设计了时间复杂度分别为 $O(nk_1k_2+mlogm+mk_1)$ 和 $O(mk_2^2+mk_1k_2+mlogm+nk_2)$ 的精确算法 P_MSR 和 P_MFR，其中 m 为片段数，n 为单体型的 SNP 位点数；为求解有空隙的 MSR 和 MFR，本文设计了时间复杂度分别为 $O(2^knk_1k_2+mlogm+nk_2+mk_1)$ 和 $O(2^kmk_1k_2+2^{3k}mk_2^2+mlogm+nk_2+mk_1)$ 的精确算法 PG_MSR 和 PG_MFR，其中 k 为片段中最大洞数。大量实验结果表明，在 Bafna 等的对应算法基础上，上述参数化算法的效率显著提高，适用于全基因组规模上的单体型组装。

针对长的 mate-pair 中洞的个数较多的情况，本文提出了求解 MSR 和 MFR 时间复杂度分别为 $O(nk_1k_22^{2h}+k_12^h+nk_2+mk_1)$ 和 $O(nk_23^{k_2}+m\log m+nk_2+mk_1)$的参数化精确算法 PM_MSR 和 PM_MFR，其中 h 为覆盖同一 SNP 位点且在该位点取空值的片段的最大数。在实际的 DNA 测序数据中，k_2 通常不大于 19，而 h 不大于 17，理论分析和实验结果均表明 PM_MSR 和 PM_MFR 算法所需的时间与片段中洞的个数的最大值 k 没有直接的关系，在片段数据中存在长 mate-pair 的情况下仍然能有效计算。

根据实际 DNA 测序片段数据的特点，本文对 MEC 和 MEC/GI 进行参数化建模，进而设计出求解这两个模型时间复杂度均为 $O(nk_22^{k_2}+mlogm+mk_1)$的精确算法 P_MEC 和 P_MEC/GI。实验结果表

明，在片段数达到 100，Wang 等提出的分支限界算法已无法运行的情况下，P_MEC、P_MEC/GI 和 Wang 等提出的遗传算法一样，仍然能快速运行。而作为精确算法，P_MEC 和 P_MEC/GI 在单体型重构精度上比 Wang 等对应的遗传算法有明显优势。

为了提高单体型的重构精度，本文提出了一个基于加权片段数据和有误差基因型的单体型组装问题计算模型 WMEC/GS，然后证明了即使片段中无空隙其也是 NP-难的。进而根据片段数据的特点，提出了求解该模型的时间复杂度为 $O(nk_2 2^{k_2}+mlogm+mk_1)$的参数化算法 P_WMEC/GS。对 MEC/GI、WMLF 和 WMEC/GS 三模型的大量实验表明 WMEC/GS 模型具有最高的单体型重构精度。

关键词： 单核苷酸多态性，单体型，基因型，NP-hard，参数化算法

ABSTRACT

Haplotyping plays an important role in locating complex disease susceptibility genes. The haplotype assembly problem is a computational problem that, given a set of DNA sequence fragment data of an individual, induces the corresponding haplotypes. For the problem, based on different optimal criteria, there are many different computational models, such as *Minimum SNP Removal* (MSR), *Minimum Fragment Removal* (MFR), *Minimum Error Correction* (MEC), *MEC with Genotype Information* (MEC/GI), etc. Most of the models for the problem have been proven to be NP-hard, and there are not practical exact algorithms for them.

Based on the observation that, for the real DNA sequence fragement data, the maximum number k_1 of SNPs that a fragment covers is usually smaller than 10, and the maximum number k_2 of the fragments covering a SNP site is usually not more than 19， we parameterize the MSR and MFR models. Based on the parameterized models, we propose two exact algorithms P_MSR and P_MFR to solve the gapless MSR and MFR models in the time complexity $O(nk_1k_2+mlogm+nk_2+mk_1)$ and $O(mk_2^2+mk_1k_2+mlogm+nk_2+mk_1)$ respectively. To solve the MSR and MFR models with gaps, we design two exact algorithms PG_MSR and PG_MFR of the time complexity $O(2^knk_1k_2+mlogm+nk_2+mk_1)$ and $O(2^kmk_1k_2+2^{3k}mk_2^2+mlogm+nk_2+mk_1)$ respectively. Extensive experiments show that, compared with Bafna et al.'s corresponding algorithms, the parameterized aglorthms above are more efficient and are practical in genome-wide haplotype assembly.

To deal with long mate-pairs, in which there may be many holes and k will be very large, we propose two parameterized exact algorithms PM_MSR and PM_MFR to solve both models in the time complexity $O(nk_1k_22^{2h}+k_12^h+nk_2+mk_1)$ and $O(nk_23^{k_2}+m\log m+nk_2+mk_1)$ respectively, where h is the maximum number of fragments covering a SNP site

whose value is unknown at the SNP site. In real DNA sequence fragment data, k_2 is usually not more than 19 and h is usually not more than 17. The theoretical complexity analysis and experimental results show that the running time of PM_MSR and PM_MFR is not directly related with k, and PM_MSR and PM_MFR algorithms can efficiently deal with MSR and MFR models with long mate-pairs respectively.

Baed on the characteristics of real DNA sequence fragment data, we parameterize MEC and MEC/GI models, and introduce two exact algorithms P_MEC and P_MEC/GI to solve them. The time complexity of both algorithms is $O(nk_2 2^{k_2}+mlogm+mk_1)$, and the experiments show that, when m is 100 and Wang et al.'s branch and bound algorithms are impractical, P_MEC and P_MEC/GI run as fast as Wang et al.'s genetic algorithms, and that, compared with Wang et al.'s genetic algorithms, P_MEC and P_MEC/GI are more accurate in haploype reconstruction.

To improve the accuracy of haplotype reconstruction, we propose a computational model WMEC/GS for the haplotype assembly problem, which is based on weighted fragments and genotype with errors. Then the model is proved to be NP-hard even with gapless fragments. To solve this model, based on the characteristics of fragment data, the paper introduces a parameterized exact algorithm P_WMEC/GS of time complexity $O(nk_2 2^{k_2}+mlogm+mk_1)$. Extensive experiments on MEC/GI, WMLF and WMEC/GS show that WMEC/GS is more accurate in haplotype reconstruction than other models.

KEY WORDS: single nucleotide polymorphisms (SNPs), haplotype, genotype, NP-hard, parameterized algorithm

目　　录

第一章 绪 论

1.1 研究背景

生命是自然界美妙绝伦的造化，每个生命从生到死都是自然界演奏的一篇华彩乐章。那么生命是如何起源和延续的？为什么会有如此多姿多彩的生命形态？几乎每个国家、每个民族都有自己关于人和生命起源的神话传说，从中国的女娲用黄土和水成泥造人到西方世界的上帝尘土造人，这些美丽的故事都世代流传，经久不衰。但是人类对生命奥秘的探索不仅仅停留在美妙的想象之中，漫长的人类实践培养了人类进行科学探索时重视观察和实验的实证精神。通过长达 5 年的对物种变化的环球考察，1859 年达尔文提出了进化论，以全新的生物进化思想动摇了生命“神创论”的根基。 1953 年，沃森（J. D. Watson）与克里克（F. H. C. Crick）发现了 DNA 双螺旋结构，这个发现阐明了生物遗传基因密码存在的形态，是 20 世纪最重大的三大发现之一，它标志着人类对生命的认识进入分子阶段，拉开了人类破解生命密码竞赛的大幕。

1990 年 10 月，以测定人类 DNA 约三十亿碱基对、识别出约 2 万个基因等为目标的人类基因组计划（Human Genome Project, HGP）启动。在美、英、日、德、法和中国等国家的科学家的努力下，2000 年 6 月人类基因组草图绘成[1]，2003 年 4 月人类基因组图谱基本完成[2]，至此人类基因组共性的一面被揭示出来，但人类个体多态性的一面没有得以充分体现。

不同的人具有不同的外貌、体格，对疾病有不同抵抗能力，对药物有不同的敏感性，从遗传上说，这是因为不同个体（除了同卵双胞胎外）的基因组不完全相同。两个人之间的 DNA 差异约占基因组的 0.5%[3]，单核苷酸多态性 SNPs（Single Nucleotide Polymorphisms）为人类染色体某个位点上的碱基变化[4-7]。SNPs 广泛分布在人类基因组中,在整个人类基因组中有几百万个 SNPs[3, 7, 8]。一个 SNP 位点指的是在一个物种的基因组 DNA 序列中不同个体可能出现不同碱基的位置。

单核苷酸多态性是一个物种中不同个体表型的主要遗传来源。识别 SNPs 对基因的精确定位、了解基因功能很有帮助，对遗传病等疾病的诊断和药物研究有重要作用，SNPs 可用于个体识别、亲子鉴定，亦可用于人类各群体的遗传

关系分析。Stephens 等采用单体型变异的方法研究人类 313 个基因中的 3899 个 SNPs,然后进行连锁不平衡分析,其结果支持了人类群体在近代扩张的说法[9]。Horikawa 等根据 SNPs 进行关联分析在墨西哥裔美国人中把 2 型糖尿病基因定位在 2 号染色体长臂，并发现 CAPN10 基因的 3 个 SNPs 和 2 型糖尿病相关[10]。

对于人类等二倍体生物，DNA 的载体——染色体是成对存在。在一条染色体 SNP 位点上的碱基序列叫做单体型（Haplotype）[11]。对于任何一个二倍体生物，都有二个单体型[12]。

为了构建一个高分辨率的人类 SNPs 位点的图谱，确定和编目人类遗传的相似性和差异性，建立一个将帮助研究者发现人类疾病及其对药物反应的相关基因的公众资源，由加拿大、中国、日本、尼日利亚、英国和美国共同资助和合作进行的项目国际人类基因组单体型图计划（HapMap）2002 年 10 月正式启动[13-15]。2005 年 10 月 26 日，HapMap 协作组①宣布其第一期工作已经完成，公布了其初步绘制的人类首张单体型图[11]。2007 年 10 月 18 日，第二期的 310 万个 SNPs 的单体型图已完成[16]。

HapMap 计划的研究成果可以帮助研究人员发现与人类健康、疾病以及对药物和环境因子的个体反应差异相关的基因，极大地推动了单体型的研究和应用。不幸的是在当前的实验技术下，测定一个个体的单体型既费钱又费时间，因此利用计算机技术来确定个体的单体型有极其重要的现实意义[12, 17, 18]。

本文主要研究如何确定个体的单体型的计算问题，下面对相关的背景知识进行详细的阐述。

1.2 单体型分型的重要意义

对人类自身认识不断深入的研究越来越表明除了与后天的环境因素有关外，人类的很多疾病都与家族遗传有关，个体罹患疾病的可能性与个体的遗传物质密切相关。

人类的遗传病可以分为以下三大类。

(1) 单基因病（monogenic disorder）：这种疾病由 DNA 序列的某个碱基对的改变所致，如血友病、白化病、红绿色盲、杜氏肌营养不良、镰刀型细胞贫血症和抗维生素 D 佝偻症等，这种疾病的遗传符合孟德尔遗传定律，因此也叫

① http://www.hapmap.org

孟德尔疾病（Mendelian disorder）。对单基因遗传病致病的基因的定位已有成熟的方法，每年都有新发现。

(2) 染色体疾病（chromosome disorder）：可进一步分为常染色体异常和性染色体异常两个类型，是由于染色体在数量或结构上的异常，如先天愚型、Edward 氏综合、先天性卵巢发育不全综合征、先天性睾丸发育不全综合征、超雌综合征、真性两性畸形等等，这类疾病很容易通过染色体检查发现。

(3) 多基因病（polygenic disorder）：这种疾病受两对以上等位基因的控制，一般还受环境等多种复杂因素的影响，也叫作人类复杂疾病（complex disease）。多基因病在人群中比较常见，并且发病率极高，如无脑儿、唇颚裂、原发性高血压、癌症、心脏病、哮喘、先天性幽门狭窄、精神分裂症和糖尿病等等。

在人类复杂疾病致病基因的研究上，研究人员发现研究单基因病的方法不再有效，人们必须从整体上研究多个基因及基因与环境之间的关系。在寻找复杂疾病致病基因的研究上，单体型起着非常重要的作用。

人类基因组大约有 30 亿个碱基对，其中约 99.5%是相同的，这样个体差异只占 30 亿个碱基序列的 0.5%。通常称在人群中所占的比例超过 1%个体碱基差异为多态性（polymorphism），最常见的多态性为单核苷酸多态性（single nucleotide polymorphisms, SNPs），大量研究表明人类基因组上大约有几百万个常见的 SNPs，平均分布密度为 1000 个碱基对就会出现 1 个 SNP[19, 20]，这样 SNPs 就成为寻找和定位致病基因的新一代的标记物（marker）。

单个 SNP 可用于单基因疾病致病基因的定位，也可以用于对疾病的关联分析，但是越来越多的研究表明，使用单体型比使用单个 SNP 在复杂疾病的关联研究上更加有效[21-27]。Akey 等[22]指出利用单个 SNP 进行疾病关联分析的能力因没有考虑两边的 SNPs 而受到局限，他们的统计分析说明使用单体型能显著提高致病基因定位的能力和鲁棒性。Clark[23]和 Tachmazidou 等[24]也指出在复杂疾病的关联分析中，单体型更有用。Browning 等[25]指出在疾病易感变异（disease-susceptibility variants）的频度在人群中不大于 5%时，基于单体型的测试比基于单个 SNP 的测试更有效。 Morris 等[26]和 Epstein 等[27]也指出，当多个与疾病相关的可疑等位基因（susceptibility alleles）之间缺乏连锁关系时，基于单体型的方法优于基于单个 SNP 的分析方法。正是因为这个原因，近年来有无数学者利用单体型进行复杂疾病的关联研究，取得了大量的研究成果[28-44]。显然在复杂疾病的关联分析和致病基因定位中，单体型起着非常重要的作用[45]。除此之外，单体型在人口结构分析、人类群体扩张等其他研究中[9, 46-51]也有重

要的应用。

1.3 单体型分型计算

如何利用高通量的生物测试技术和计算机技术、花费较少的人力和物力来确定个体的单体型是当前世界生物信息学的一个研究热点。自从 1990 年 Clark 在 Mol. Biol. Evolut.上发表论文提出用一群个体的基因型去推断出每个个体的单体型的思想[12]以来，国外发达国家有很多学者如 D. Gusfield[18, 52-57]、G. Lancia[17, 58, 59]、V. Bafna[60-62]、Rizzi [63] 、J. Li、T. Jiang[64-66]和 L. Wang, Y. Xu[67]等，国内主要有中科院数学与系统科学研究院的章祥荪教授领导的生物信息学研究中心[68-74]和中国科技大学的张强锋[75-80]等对单体型分型中的计算问题进行了大量研究, 提出了单体型检测的各种模型和算法，这些模型总的说来主要分为单体型组装和单体型推断两大类[70]。

1.3.1 单体型组装问题

单体型组装问题（The Haplotype Assembly Problem）也叫个体单体型问题（The Individual Haplotyping Problem）。对于人类等二倍体生物，染色体是成对存在, 都有二个单体型。在当前的技术条件下直接把一对染色体分开，然后对每一条染色体进行独立测序难度很大，花费的金钱和时间过分昂贵，因此，实验室的测出的 DNA 片段数据是来自于一对染色体，而单体型组装问题就是给定一组来自某对同源染色体的由 DNA 测序方法得到的 DNA 片段数据，根据片段上的 SNP 值组装出两条单体型。

Lancia 等[17]对这个问题进行开创性研究，提出了下面 3 个计算模型：

(1) 最少片段删除 (Minimum Fragment Removal, MFR): 对于一个个体的 DNA 片段测序数据，删除最少的 DNA 片段，使得剩下的片段可以分成两个子集，使得每个子集的片段能决定一个单体型；

(2) 最少 SNP 位点删除 (Minimum SNP Removal, MSR)：对于一个个体的 DNA 片段测序数据，删除最少的 SNP 位点上的值，使得所有的片段可以分成两个子集，使得每个子集的片段在剩下的 SNP 位点上能决定一个单体型；

(3) 最长单体型重建 (Longest Haplotype Reconstruction, LHR)：该问题是去掉最少的 DNA 片段，使得剩下的片段可以分成两个子集，使得每个子集的片段能决定一个单体型，且使获得的两个单体型的长度和最长。

Lippert 等[81]进一步提出了第 4 个计算模型：

(4) 最少错误更正 (Minimum Error Correction, MEC)：MEC 模型也叫做最少字符翻转 (Minimum Letter Flips, MLF)，该模型要求修改 DNA 片段上最少的 SNP 值, 使得修改后的 DNA 片段可以分成两个子集，使得每个子集的片段能决定一个单体型。

单体型组装问题的计算模型主要是以上 4 个，还有其他在此基础改进的模型，如加权最少字符翻转 (Weighted MLF, WMLF)[69, 82]等。

由于当前技术的局限性，DNA 测序存在错误和直接测序片段长度较短，从而很难确定那些 DNA 片段来自哪条染色体，单体型组装问题的上述计算模型无一例外都被证明为 NP-hard，因而缺乏实用的精确算法。

1.3.2 单体型推断问题

利用现有的生物实验技术如 SNP 芯片可以较容易得到个体的基因型，单体型推断问题（the Haplotype Inference Problem）就是如何从一个群体中多个个体的基因型来推断出这些个体的单体型。

目前单体型推断的主要计算模型概述如下：

(1) 基于节俭原则（Parsimony）的模型：这类模型基于人群中的单体型个数尽可能少的这种假设。Clark[12]最先提出了基于节俭原则的几条基本推断规则——Clark 规则，这些规则对大部分数据简单有效，但是有些情况下需要随机判断从而使最终的结果很有可能不是最节俭的。后来 Gusfield 等学者基于 Clark 规则提出了多种不同模型[18, 52, 83]。

(2) 基于进化史（Phylogeny）的模型：Gusfield, Bafna, Eskin 等[53, 61, 84, 85]提出了基于进化史对单体型进行推断的模型。

(3) 基于家族（Pedigree）的模型：这类计算模型是利用一个家族的基因组数据来推断个体的单体型[65, 66, 86]。

上述三类模型中的大多数问题也都被证明是 NP-难的。

(4) 基于统计（Statistics）的模型：单体型推断的统计模型以 HWE（Hardy-Weinberg Equilibrium）假设为前提，主要采用 EM 方法（Expectation-Maximization) [87, 88]、基于 Bayesian 估计和 Gibbs 取样的统计方法[89]和基于 Markov 链模型的方法[90]。基于 EM 方法、基于 Bayesian 估计和 Gibbs 取样的统计方法必须假设单体型片段在遗传中保持不变，需要对整条单体型估计频率，且无法对长的单体型进行分析，因而有很大的局限性，而基于 Markov 链模型

的算法时间复杂度高，而且这些算法都是启发式算法，容易收敛到局部最优，不能获得最优解。

从单体型的重建精度而言，一般说来单体型组装比单体型推断要高。本文主要研究单体型组装问题。

1.4 参数计算理论

利用计算机解决实际的工程应用中各种问题时，需要考虑的是所要求解的问题是易解的还是难解的。一般说来，所需的存储空间和运行时间在问题规模的多项式复杂度内，则该问题被看作易解的问题；而所需的存储空间和运行时间在问题规模的指数复杂度上，则该问题被看成难解的问题。研究问题计算复杂度的 NP-完全理论是理论计算机科学中的最伟大成就之一[91]，该理论把能在多项式时间内求解的问题归于 P 类问题；把给出问题的答案后，能在多项式时间验证该答案是否正确的问题归于 NP 类问题；一个属于 NP 类的问题，如果所有其他 NP 类问题都可以在多项式时间内归约成该问题，则该问题是 NP-完全问题；对于一个计算问题，如果存在着某个 NP-完全问题能在多项式时间内归约成该问题，则它是 NP-难的。NP-完全理论认为，NP-难问题不比 NP-完全问题容易，是不可能有多项式时间算法的，除非 P＝NP。

很多计算问题都是 NP-完全或 NP-难的，如最小顶点覆盖（Vertex Cover）问题：给定一个图 G，找出 G 中覆盖每条边的最小顶点子集。NP-完全或 NP-难问题用常规的算法在目前的计算资源下通常是不可行的，但是实际的应用迫切需要解决这类问题，国内外计算机领域大批知名学者几十年来对这类问题不断进行深入研究，在计算机理论和算法方面取得了大量的成果。目前处理 NP-难的计算问题通常有下面几类方法：近似算法[92, 93]、随机算法[94]和启发式算法[95]。但是有些问题很难设计出有意义的近似算法，或者近似解对这些问题毫无意义，而随机算法和启发式算法的近似程度很难保证[96]。参数计算理论作为一种处理 NP-难问题的新的方法和手段，近年来受到了国内外学者的广泛研究[97-114]。

根据参数计算理论，某些 NP-难问题可以进行参数化，即设计出算法复杂度为 $O(f(k)n^c)$ 的算法，其中 n 是问题的规模，c 是一个常数，k 是一个参数，且该参数在实际应用中只在一个小的范围内变化。这样如果 c 较小，这类参数

化算法就可以充分利用 k 是一个“小参数”这一特殊性质，快速地解决这些NP-难问题。

例如，上面的顶点覆盖问题可以如下参数化：给定一个图 $G(V, E)$和一个整数 k，$|V|=n$，判断图 G 是否存在一个至多为 k 个点的集合 $C \subseteq V$，使得图 G 中的每条边至少有一个端点在 C 中。对于参数化的顶点覆盖问题，Chen 等[115]设计了目前最好的参数化算法，其时间复杂度为 $O(1.286^k+kn)$。在有些特定的实际应用，k 的值不是很大，如苏黎士瑞士联邦工学院（Swiss Federal Institute of Technology, Zurich）的计算生物化学研究组（Computational Biochemistry Research Group (CBRG)）的达尔文项目组（Data Analysis and Retrieval with Indexed Nucleotide/Peptide Sequence (Darwin)）[116]将基因序列冲突转化成上述点覆盖问题[116, 117]，即求在一给定图中是否存在一个规模为 k 的点覆盖，而在其生物应用中，所需点覆盖的规模 k 不超过 60[117, 118]。在达尔文项目组中，Chen 等的参数化算法被实现，并能有效运行[119]。

除顶点覆盖问题，参数计算理论在求解实际问题的成功典范还有很多，如程序类型检测（ML Type-Checking）问题和瑕点覆盖（Fault Coverage）问题等。程序类型检测问题是指数时间难解的（因此比 NP-完全问题更难）。利用参数计算理论，对于长度为 n，嵌套深度为 k 的程序，程序类型检测问题可在时间 $O(2^k+n)$内解决[120]。注意，在一般计算科学中，程序的嵌套深度一般不超过 10。因此，虽然这一问题从理论上说是比 NP 完全问题更难解的问题，但利用其参数特殊性，这一问题在实际中是可以解决的。瑕点覆盖问题指的是给定一个有缺陷的存储器或处理器阵列，使用最少的备用行和备用列进行替换使存储器或处理器阵列恢复正常功能，该优化问题是 NP-完全问题[121]。基于实际芯片中备用的行数 k_u 和备用的列数 k_l 是有限的，实际的瑕点覆盖问题就转化成下面的二分图的受约束最小点覆盖（Min-CVCB）问题：给定一个二分图 G=(U, L, E) 以及两个正整数 k_u 和 k_l，构造图 G 中的一个最小点覆盖 C，使得 C 中包含至多 k_u 个 U 中结点和 k_l 个 L 中结点（或者报告没有这样的最小点覆盖存在）。Min-VCB 问题是 NP-完全问题[122]，Chen 等进一步提出了一个复杂度为 $O(1.26^k n)$ 的参数化算法[121]，其中参数 $k = k_u + k_l$，对于实际的瑕点覆盖问题转化成的 Min-CVCB 问题，典型的 k 值不超过 40，因而 Chen 等的参数化算法[122]又是一个求解 NP-难解问题的成功典范。

从以上几个例子可以看出，参数计算理论是将计算理论和计算实践有机地结合起来，从而解决实际中出现的计算难解问题。这一方法已在相当多领域中

得到应用，如数据库系统、芯片设计、计算生物学、人工智能、密码学等（参看文献[99-101, 123-126]）。

1.5 本文的主要研究内容及结构安排

本文主要研究单体型组装问题，研究如何利用参数计算理论将单体型组装问题进行参数化建模，进而提出进行精确求解且在实际应用中快速有效的参数算法, 最后在已有模型的基础上，结合 DNA 测序仪能提供的置信度信息和基因型的不确定性，提出一个高精度的单体型组装模型。本论文的具体组织如下。

第一章 绪论：本章对整个论文的研究背景和主要研究内容进行概述。首先简要介绍单体型分型计算的来源及意义。接下来，对单体分型计算进行了简单叙述，指出单体分型计算分为单体型组装问题和单体型推断问题。指出其中绝大部分计算模型均是 NP-难的。然后对目前处理 NP-难问题的一种方法参数计算理论进行了简介。最后给出全文结构安排。

第二章 单体型组装问题：本章对相关的分子遗传学背景知识及单体型组装问题的基本概念和研究现状进行介绍。首先对生物的遗传物质及遗传法则进行介绍，给出了单核苷酸多态性（Single Nucleotide Polymorphisms，SNP）、单体型（Haplotype）和基因型（Genotype）等相关概念，然后阐述了单体型组装问题的基本概念和定义，最后使用这些定义对单体型组装问题的计算模型进行重新表述，并对各模型的研究现状进行概述。

第三章 MSR 和 MFR 参数化模型和算法：本章首先分析了 DNA 测序实验数据的特点，根据一个片段所覆盖的最大 SNP 位点数 k_1 和覆盖一个 SNP 位点的最大片段数 k_2 均很小的特点，对 MSR 和 MFR 进行参数化建模，进而设计了时间复杂度分别为 $O(nk_1k_2+mlogm+mk_1)$和 $O(mk_2^2+mk_1k_2+mlogm+nk_2)$的精确算法 P_MSR 和 P_MFR 来求解无空隙的 MSR 和 MFR，其中 m 为片段数，n 为单体型的 SNP 位点数；然后进一步设计了时间复杂度分别为 $O(2^knk_1k_2+mlogm+nk_2+mk_1)$ 和 $O(2^kmk_1k_2+2^{3k}mk_2^2+mlogm+nk_2+mk_1)$ 的精确算法 PG_MSR 和 PG_MFR 求解有空隙 MSR 和 MFR 模型。最后通过大量的实验显示，本章提出的参数化算法在 Bafna 等的对应算法基础上，效率有了显著提高。

第四章 有长 Mate-Pair 时 MSR 和 MFR 参数化算法：在片段数据中有较长

mate-pair 的情况下, 复杂度以片段中洞的最大个数 k 为 2 的指数的算法已不再实用。本章针对长的 mate-pair 中洞的个数较多的情况，提出了求解 MSR 和 MFR 时间复杂度分别为 $O(nk_1k_2 2^{2h}+k_1 2^h+nk_2+mk_1)$和 $O(nk_2 3^{k_2}+m\log m+nk_2+mk_1)$的参数化精确算法，其中 h 为覆盖同一 SNP 位点且在该位点取空值的片段的最大数。然后用实验测试表明本章提出的算法在有长 mate-pair 片段数据的情况下仍然能有效地进行计算, 具有很好的可扩展性。

第五章 MEC 与 MEC/GI 参数化模型和算法：根据实际 DNA 测序片段数据的特点，本章对 MEC 和 MEC/GI 进行参数化建模，在此基础上设计出求解这两个模型的精确算法 P_MEC 和 P_MEC/GI。这两个算法的时间复杂度均为 $O(nk_2 2^{k_2}+mlogm+mk_1)$，空间复杂度均为 $O(mk_1 2^{k_2}+nk_2)$。然后通过实验测试表明，P_MEC 和 P_MEC/GI 和 Wang 等对应的分支限界算法具有相同的重构精度，而在片段数达到 100，Wang 等提出的分支限界算法已无法运行的情况下，P_MEC、P_MEC/GI 和 Wang 等提出的遗传算法一样，仍然能快速运行。作为精确算法，P_MEC 和 P_MEC/GI 在单体型重构精度上比 Wang 等对应的遗传算法有明显优势。

第六章 WMEC/GS 模型和参数化算法：利用 DNA 碱基的置信度信息及包含基因型误差信息的基因型谱（GenoSpectrum），本章提出了带误差基因型的加权最少错误纠正模型 WMEC/GS（Weighted Minimum Error Correction with GenoSpectrum），紧接着证明其是 NP-难的。进而根据片段数据的特点，提出了求解该模型的参数化精确算法。最后用大量的实验展示在 MEC/GI、WMLF 和 WMEC/GS 三模型中，WMEC/GS 模型具有最高的单体型重构精度。

第七章 总结：本章对全文的工作进行总结，并对下一步的研究方向进行展望。

第二章 单体型组装问题

2.1 遗传的物质基础及遗传法则

生命缤纷多彩，物种千变万化，但是种豆得豆，种瓜得瓜，是什么在控制生命的诞生和发展的进程？公元前五世纪希波克拉底（Hippocrates）认为子代具有亲代的特性是因为在精液或胚胎里集中了来自身体各部分的微小代表元素（Elememt）；100 年后，亚里斯多德（Aristotle）认为精液提供了后代的蓝图，生物的遗传是个体胚胎发育所需信息的传递。1865 年奥地利人孟德尔（G. J. Mendel）从他 8 年植物杂交实验的结果发现了生物每一个性状都是通过遗传因子来传递的，遗传因子是一些独立的遗传单位。1909 年，丹麦人约翰森（W. L. Johannsen）将孟德尔提出的“遗传因子”改称为后来广为流传的“基因”（Gene）。1910 年摩尔根（T. H. Morgan）等创立了连锁定律，1944 年艾弗里（O. T. Avery）确定遗传物质为脱氧核糖核酸（Deoxyribonucleic acid，DNA），1953 年沃森（J. D. Watson）和克里克（F. H. C. Crick）建立 DNA 分子的双螺旋结构模型。至此人们普遍认为生命个体的遗传信息包含在细胞中全部 DNA 所构成的基因组中，生命的表型和其他性状是由基因组决定的。

2.1.1 染色体

对于人类这样的真核生物，生命体的基本单位细胞（Cell）中有细胞核（Nucleus），在细胞核中有染色体（Chromosome）。1910 年摩尔根等创立连锁定律的同时也证实了遗传信息（基因）在染色体上以线状排列。

染色体的化学成分主要是 DNA 和蛋白质，在细胞发生有丝分裂时期容易被碱性染料着色，因此而得名。染色体是遗传物质的主要载体，细胞中大部分的 DNA 在染色体上，染色体中 DNA 含量稳定，是主要的遗传物质。

在无性繁殖物种中，生物体内所有细胞的染色体数目都一样。而在有性繁殖物种中，生物体的体细胞染色体成对分布，称为二倍体。人类体细胞中含有 22 对常染色体，1 对性染色体。男性的性染色体对由 X 和 Y 染色体构成（如图 2-1[①]），女性的性染色体由 2 条 X 染色体构成。其中每一对染色体中，一条来自

[①] http://www.imagewa.com/Photo/269/100.html

于母亲，另一条来自于父亲。

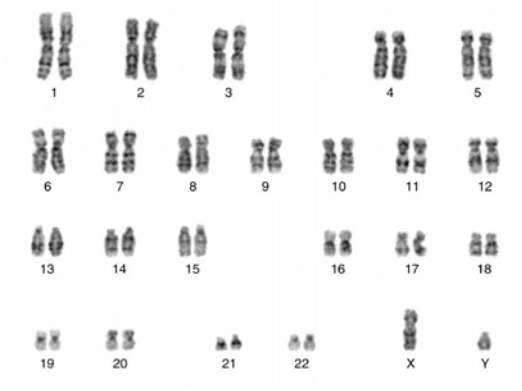


图 2-1 人类男性体细胞染色体

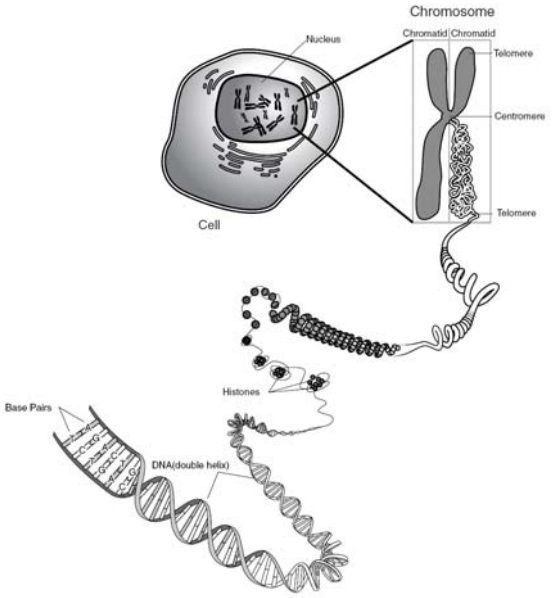


图 2- 2 染色体组成

在每一条染色单体中，1 个 DNA 分子像细长的丝带缠绕在组蛋白（Histone）分子表面形成所谓的绳珠结构，然后在此基础进一步形成了染色体的空间三维结构，如图 2- 2[①]所示。

2.1.2 DNA 分子与基因

构成 DNA 分子的基本单位是脱氧核苷酸（Deoxyribonucleotide），脱氧核苷

① http://www.accessexcellence.org/RC/VL/GG/chromosome.html

酸由磷酸（Phosphate）、脱氧核糖（Deoxyribose）和碱基（Base）构成，其中碱基有四种：腺嘌呤（Adenine，缩写为 A）、鸟嘌呤（Guanine，缩写为 G ）、

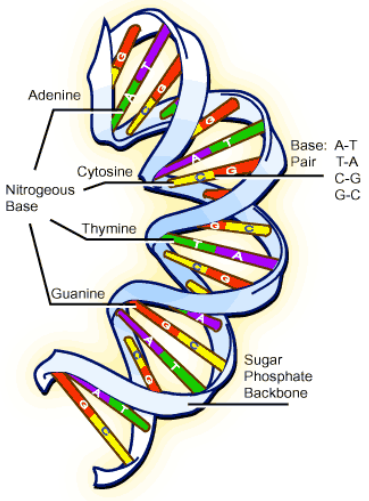


图 2-3 DNA 双螺旋结构

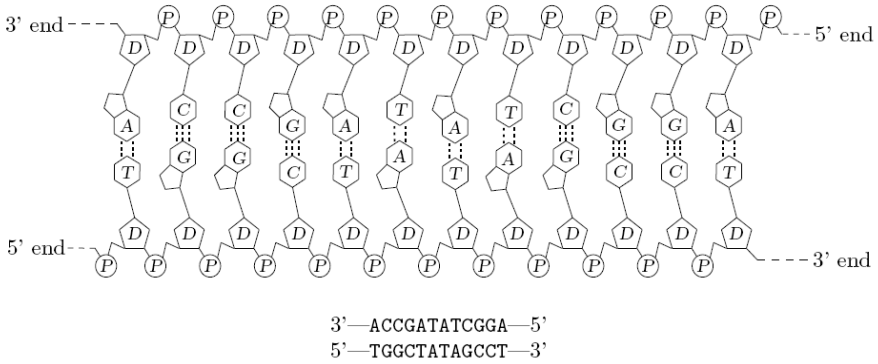


图 2-4 碱基互补配对

胞嘧啶（Cytosine, 缩写为 C ）和胸腺嘧啶（Thymine, 缩写为 T），由此脱氧核苷酸可分为四种，分别用 A、C、G、T 表示。脱氧核苷酸之间通过 3′、5′-磷酸二酯键相连形成脱氧核苷酸链，两条脱氧核苷酸链平行但反向盘旋成的规则的 DNA 分子双螺旋结构，两条链之间是通过互补碱基配对连接在一起，其中 A 与 T 以 2 个氢键相配对，C 与 G 之间以 3 个氢键配对。DNA 的双螺旋结构如图 2-3 所示①，碱基互补配对如图 2-4 所示[127]。在图 2-4 中，上面是 DNA 分子

① http://www.scq.ubc.ca/?p=263

的化学结构，下面是该分子的碱基对序列。在 DNA 分子的化学结构图中，P 表示磷酸，D 表示脱氧核糖，A、C、G 和 T 分别表示四种碱基，一个 DNA 单链开始于 3′端（脱氧核糖端），中止于 5′端（磷酸端）。

DNA 分子的一级结构指 DNA 分子中核苷酸的排列顺序，即核苷酸序列或碱基序列，DNA 分子的多样性是由碱基排列顺序的多样性决定的。

现代遗传学认为，基因是指位于染色体的特定位置、编码特异的蛋白质或 RNA 的一段核酸序列（通常是 DNA 序列），是遗传物质的结构和功能单位。对于真核生物而言，基因位于染色体上，并在染色体上呈线性排列。基因不仅可以通过复制把遗传信息传递给下一代，还可以使遗传信息得到表达，也就是使遗传信息以一定的方式反映到蛋白质的分子结构上，从而使后代表现出与亲代相似的性状。基因组(Genome)代表了一个生物细胞内的全部基因和染色体组成。

2.1.3 分子生物学的中心法则

1958 年，克里克提出了两个学说，奠定了分子生物学的理论基础。第一个

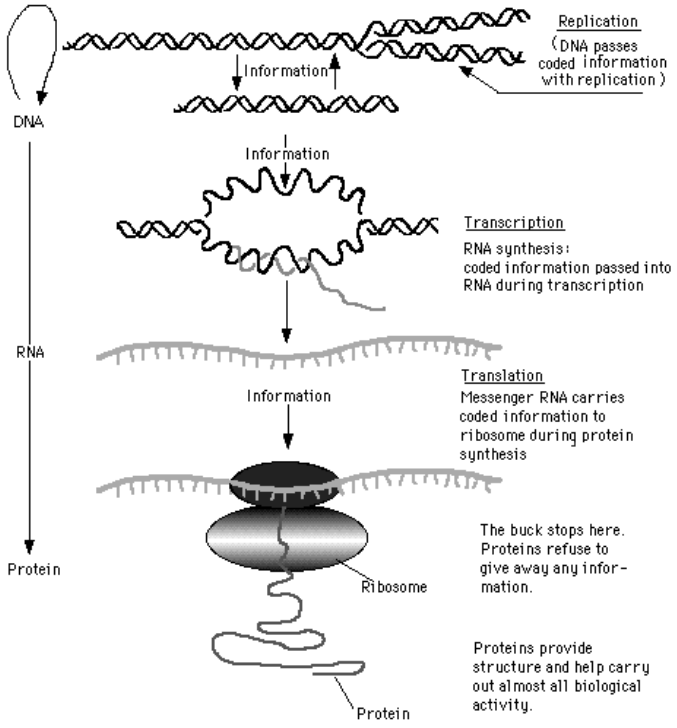


图 2-5 生物信息学的中心法则

学说是“序列假说”，它认为一段核酸的特殊性完全由它的碱基序列所决定，碱

基序列编码一个特定蛋白质的氨基酸序列，蛋白质的氨基酸序列决定了蛋白质的三维结构。第二个学说是“中心法则”，遗传信息只能从核酸传递给核酸，或核酸传递给蛋白质，而不能从蛋白质传递给蛋白质，或从蛋白质传回核酸。后来，沃森把“中心法则”更明确地表示为，遗传信息只能从 DNA 传到 RNA，再由 RNA 传到蛋白质。分子生物学的中心法则（Central dogma of molecular biology）如图 2-5[①]所示，从“中心法则”可以看出，遗传信息的一般流动方向是：遗传信息可以通过 DNA 的自我复制（Replication）从 DNA 流向 DNA，也可以通过转录过程（Tanscription）从 DNA 流向 RNA，进而通过翻译过程（Translation）从 RNA 流向蛋白质，最后通过具有空间结构的蛋白质完成细胞的各项生命活动。

2.2 单核苷酸多态性、单体型和基因型

人类不同个体有不同的外貌和体格，对疾病有不同的抵抗能力，从遗传上说是因为不同的人的基因组不完全相同。人类的基因组包含 23 对染色体上的 DNA 序列，总长度达 3 亿个碱基。不同人的 DNA 序列极为相似，若对两个不同的人的同源染色体进行比较，他们的 DNA 序列上可以连续数百个碱基都是相同的。最近的研究[3]表明两个不同的同源染色体的 DNA 序列中，99.5%是相同的，其他不同的部分可能是短的 DNA 片段的插入（Insertion）、删除（Deletion）和单个碱基的差异，其中主要是单个碱基的差异。在同一个物种中，这些遗传物质上的变化如果在群体中发生频率低于 1%则称为变异（Mutation），如果不小于 1%，则称多态现象（Polymorphism）。

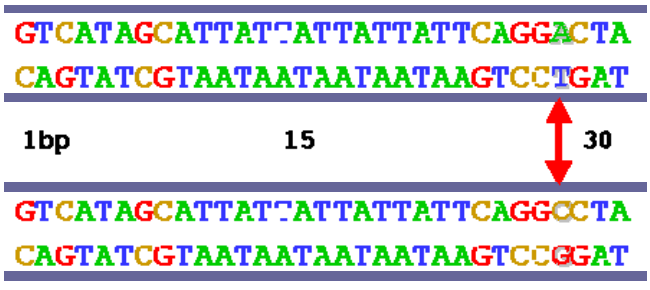


图 2-6 单核苷酸多态性

单核苷酸多态性（Single Nucleotide Polymorphism，SNP）：单核苷酸多态

① http://www.cbs.dtu.dk/staff/dave/roanoke/genetics980320f.htm#1.%20Digital%20River

性指染色体基因组上在单个核苷酸碱基尺度上的变化（这种变化最少在群体中的频率不小于 1%）而引起的 DNA 序列多态性。在图 2-6[①]中，第 30 个碱基处是一个 SNP 位点。虽然脱氧核苷酸碱基有四种，但 SNP 通常只是二等位基因（Biallelic）的变异，即只含两种等位基因型（碱基对）。

人类个体内超过 90%的多态现象都是单核苷酸多态性(SNPs)。由于在 DNA 序列中，大约 1000 个碱基中有 1 个 SNP，所以 SNP 作为分子标记广泛地用于致病基因定位等后基因组研究之中。

单体型（Haplotype）：单体型指的是一条染色体上或染色体一段区域内相关的 SNP 序列。在图 2-7 中，对应的三个单体型分别为 CTC、CAT 和 ATC。 确定染色体上的单体型叫单体型分型（Haplotyping）。

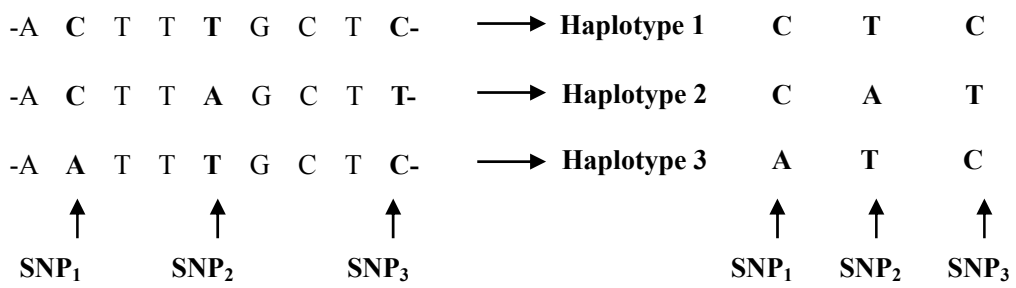


图 2-7 单体型

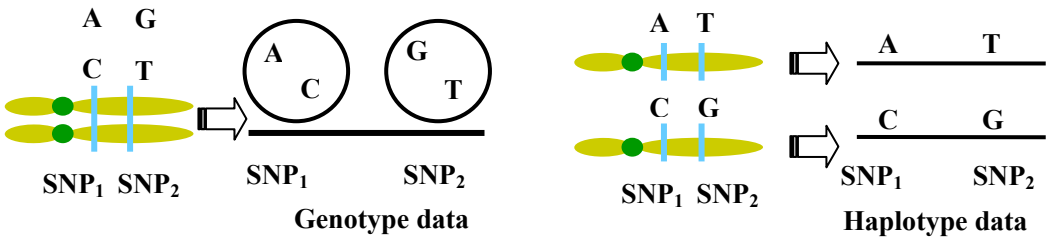


图 2-8 一对染色体的基因型及对应的单体型

基因型（Genotype）：我们常见的大多数动植物都是双倍体（Diploid），人类也是如此。人的细胞核内的染色体成对存在，成对存在的同源染色体其中一条来自父亲，另一条来自母亲。由于在 DNA 测序中很难把同源染色体分离，这样通常实验室里面得到的就是这两条同源染色体共同表现出来的 SNP 复合序

① http://cmbi.bjmu.edu.cn/cmbidata/snp/index00.htm

列，称之为基因型。在图 2-8 中，左边是某个体的一对同源染色体对应的基因型，表示为 A/C G/T；由左边的基因型信息是无法准确确定右边的两个单体型，因为由 AG 和 CT 这两条单体型构成的基因型也是 A/C G/T。

对于任意一个 SNP 位点来说，一对同源染色体上的碱基可以是相同的，也可以是不同的。同源染色体在某个 SNP 位点上的取相同碱基值的现象叫纯合(homozygous)，这时称这个 SNP 位点上的基因型为纯合子（homozygote）。同源染色体在某个 SNP 位点上的碱基值的不同的现象叫杂合(heterozygous) ，这时称这个 SNP 位点上的基因型为杂合子（heterozygote）。

个体的基因型和单体型可以通过生物实验技术测定[128]，但利用纯粹的生物实验技术测定单体型在时间和成本上过分昂贵。因此，利用计算技术确定单体型的研究具有重要的现实意义。

2.3 单体型组装问题

单体型组装问题（Haplotype Assembly Problem）最早是由 Lancia[17]提出，也叫个体单体型问题（Individual Haplotyping Problem）。对于人类等二倍体生物，染色体是成对存在, 都有二个单体型。在当前的技术条件下直接把一对染色体分开，然后对每一条染色体进行独立测序难度很大，花费的金钱和时间过分昂贵，因此，实验室的测出的 DNA 片段数据是来自于一对染色体，而单体型组装问题就是给定一组来自某对同源染色体的由 DNA 测序方法得到的 DNA 片段数据，根据片段上的 SNP 值组装出两条单体型。

图 2-9 是一个单体型组装问题的示例[81]，其中图 2-9(a)是来自于某个个体的联配的 DNA 测序片段(fragment)数据；而单体型组装问题只是关心 DNA 片段在 SNP 位点上的取值，图 2-9 (b)则标记出了对应的 SNP 值；图 2-9 (c)根据 SNP 的取值，把所有的 DNA 片段划分为两个集合，使得在同一个集合里的片段在对应的 SNP 位点上的取值均相等，即认为同一个集合里的片段来自于同一条染色体，这样就得出该个体的 1 对单体型为“CCACGAGT”和“GATTATCA”。当 DNA 测序过程没有错误时，单体型组装问题很容易得以解决，但是由于测序过程的错误是不可避免的[17]，因此实验室测得的 DNA 片段数据常常不可能划分为两个集合，使得在同一个集合里的片段在对应的 SNP 位点上的取值均相等，这样单体型组装问题在计算上就变得很困难。

国内外众多学者对单体型组装问题都有过深入研究，国外有 G. Lancia[17, 58, 59]、V. Bafna[60-62] 、Rizzi [63] 、Lippert[81]、Cilibrasi[129, 130]等，国内主要有中科院数学与系统科学研究院的章祥荪教授领导的生物信息学研究中心[68-74]，其中有王瑞省、吴凌云等，另外还有 Y. Wang[131]。他们根据不同的优化准则提出了众多的单体型组装计算模型，其中著名的有 MFR、MSR 和 MEC 模型及其变种。在对这些模型进行形式化描述前，下面首先介绍单体型组装问题的相关概念。

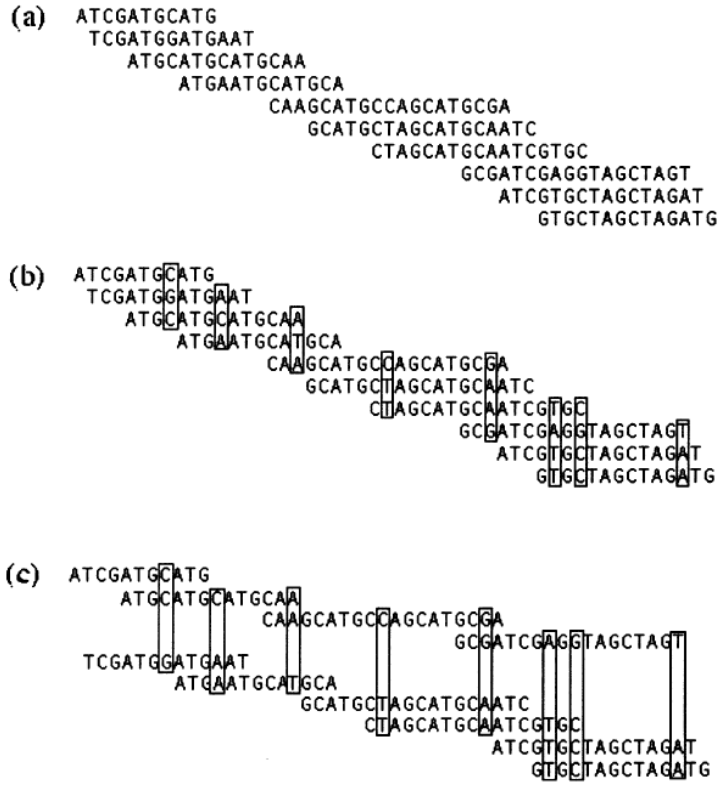


图 2-9 单体型组装问题：（a）个体的联配的 DNA 片段数据； (b)对 SNP 位点做了标记的联配 DNA 片段数据 ；（c）按照 SNP 的取值对 DNA 片段进行的划分

2.3.1 单体型组装问题相关概念

如前所述，单核苷酸多态性 SNPs 指的是在群体 1%以上个体中出现染色体单个位点上的碱基变异。人类基因组包含着广泛分布的几百万个 SNPs。人类等双倍体生物的 DNA 序列是按染色体成对出现的。一条染色体上某一区域的 SNP 位点上的碱基序列叫做单体型，而一对染色体上对应区域 SNP 位点上碱基对的序列叫做基因型。在图 2-10 中，该个体的单体型是(G, C, A, T, G)和(A, T, A, C,

G)；基因型为(A/G, C/T, A/A, C/T, G/G)。

一对染色体对应位点上 SNP 值可以是相同的，这种现象叫纯合(homozygous)；也可以是不同的，这种现象叫做杂合(heterozygous)。这样单体型就可以用字符集{0, 1}上的字符序列来表示, 不必用真正的碱基字符，其中‘0’通常表示人群中大部分单体型在该位点上的 SNP 值, 而‘1’则表示人群中少部分单体型在该位点上的 SNP 值。同理，基因型则可以用字符集{0, 1, 2}上的字符序列来表示，其中 0(或 1)是纯合子，表示两个单体型上对应的 SNP 值都为 0(或 1)，2 是杂合子。图 2-10 中的单体型可表示为 “0 1 0 1 1” 和“1 0 0 0 1” ，其中在第 1 个 SNP 位点‘0’表示‘G’, ‘1’表示‘A’; 在第 2 个 SNP 位点‘0’表示‘T’, ‘1’表示‘C’; 其余的类似。而基因型可表示为 “2 2 0 2 1”。

G ... C ... A ... T ... G .

A ... T ... A ... C ... G .

图 2- 10 单体型

SNP 位点

片段										
	-	-	-	-	-	0	1	-	1	0
	-	0	1	-	-	0	-	-	-	-
	0	1	1	0	-	-	-	-	-	-
	1	0	1	-	0	1	-	-	-	-
	-	1	0	-	-	-	-	-	-	-
	-	-	0	1	-	-	-	-	-	-
	-	-	-	0	1	0	-	-	-	-
	-	-	-	-	-	-	0	1	-	-
	-	-	-	-	-	-	1	0	0	1
	-	-	-	-	0	1	-	-	-	-
	-	-	-	-	-	0	0	-	-	-

图 2-11 SNP 矩阵

由于单体型组装问题只关心 DNA 片段在 SNP 位点上的取值，对于一对染色体某个对应区域内的 n 个 SNP 位点按其在染色体上的次序从左到右记作 S：{1，2，...，n}，m 个 DNA 片段记作 F：{1，2，...，m}。任意 SNP 位点应该被某些 DNA 片段覆盖，任意片段在它所覆盖的 SNP 位点的取值为{0，1，−}，其中‘−’为空值，表示该片段没有覆盖该 SNP 位点，或在该位点的取值未知。这样 DNA 片段的数据集可以表示为在{0，1，−}上的一个 $m \times n$ 的矩阵，叫做 SNP 矩

阵 $\mathbf{M}$[17]。图 2-11 是一个 11×10 的 SNP 矩阵。SNP 矩阵的列表示 SNP 位点，行表示片段在对应的 SNP 位点上的取值，$\mathbf{M}_{i,j}$ 表示第 i 个片段在第 j 个 SNP 位点上的取值。

下面是与 SNP 矩阵 **M** 相关的几个定义。

定义 2.1 如果$(\exists k(\mathbf{M}_{i,k} \neq \text{‘}-\text{’}) \wedge (k \leq j)) \wedge (\exists r\ (\mathbf{M}_{i,r} \neq \text{‘}-\text{’}) \wedge (j \leq r\ )))$，则称行 i **覆盖**列 j。

行 i 覆盖列 j 就是行 i 在列 j 上取值非空，或在列 j 前至少有一列，在列 j 后也至少有一列，使得行 i 在这两列上的取值均非空。在图 2-11 中行 2 覆盖列 2 到 6，但是行 2 没有覆盖列 1，也没有覆盖列 7 到 10。如果某一行覆盖某一列，但是该行在该列上的取值为空，则称该行在该列上有个**洞**(hole)。图 2-11 中行 4 在列 4 上有个洞。如果 **M** 的所有行都没有洞，则称 **M** 为**无空隙**的 SNP 矩阵。

M 的行覆盖的最左边和最右边的列号分别用函数 *left* 和 *right* 来表示，即行 i 覆盖的最左边的列是 $left(i)$，覆盖的最右边的列是 $right(i)$。在图 2-11 中，$left(2) = 2$，$right(2) = 6$。

为了叙述简便，本文中把取 **M** 的前 i 行构成的 SNP 矩阵记作 $\mathbf{M}(i, :)$，取 **M** 的前 j 列构成的 SNP 矩阵记作 $\mathbf{M}(:, j)$。

定义 2.2 如果两行在某一列上的值都不是空值，且这两行在该列上的值不相等，那么这两行在该列上**冲突**。

如果两行在所有的列上均不冲突，则这两行**兼容**。

如图 2-12 所示的 SNP 矩阵中，行 1 与行 3 在列 1、2 和 3 上冲突，在列 4 上不冲突；行 1 和行 2 在所有的列上均不冲突，它们兼容。

如果测序过程没有任何错误，代表来自于同一染色体的片段的行必定两两兼容。

定义 2.3 如果 SNP 矩阵 **M** 的所有行可以分成 2 个不相交的子集，每个子集中的所有行都相互兼容，则 **M** 是**可行**的。

如果 **M** 是可行的，则很容易找到一个划分，使划分在同一个子集中的所有行都兼容，进而通过同一个子集中的行很容易重建与这些行兼容的单体型。

一个 SNP 矩阵 **M** 是可行的当且仅当可以找到一对单体型，使得 **M** 中的任意行总是可以与其中的一个单体型兼容。这时，称 **M** 可由这对单体型导出。

如果测序过程没有任何错误，测序片段数据对应的SNP矩阵必定是可行的，

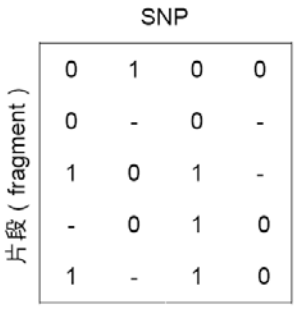


图 2-12 可行的 SNP 矩阵

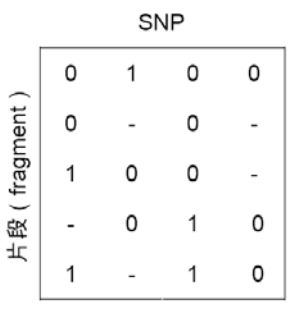


图 2-13 不可行的 SNP 矩阵

那么通过 DNA 测序片段数据很容易得出个体的一对单体型。如图 2-12 所示的 SNP 矩阵，该矩阵的所有行可以划分为{行 1，行 2}和{行 3，行 4，行 5}两个子集， 显然这样的划分能使同一个子集中的任意两行在所有的 SNP 位点上均不冲突，即互相兼容，而且很容易可以得出第一个子集中的片段可以确定一个单体型 “0 1 0 0”，第二个子集中的片段可以可以确定另一个单体型 “1 0 1 0”。可是由于测序过程的错误是不可避免的，因此实验室测得的片段数据对应的 SNP 矩阵通常是不可行的。图 2-12 和图 2-13 中的 SNP 矩阵只有第 3 行第 3 列的 SNP 值不同，可是图 2-13 所示的 SNP 矩阵是不可行的。

2.3.2 单体型组装问题计算模型及研究现状

基于不同的优化准则，单体型组装问题有多个不同的计算模型。

Lancia 等[17]在 2001 年的欧洲算法会议（ESA）上最先提出单体型组装问题，并引入了下面 3 个计算模型：

(1) 最少片段删除 (Minimum Fragment Removal, MFR): 给定一个 SNP 矩阵 **M**，删除最少的行使 **M** 可行；

(2) 最少 SNP 位点删除 (Minimum SNP Removal, MSR)：给定一个 SNP 矩阵 **M**，删除最少的列使 **M** 可行；

(3) 最长单体型重建 (Longest Haplotype Reconstruction, LHR)：给定一个 SNP 矩阵 **M**，删除最少的行使 **M** 可行，且使获得的两个单体型的长度和最长。

对于无空隙的 SNP 矩阵，在片段互不包含的情况下，Lancia 等[17]把 SNP 矩阵转化为有向图，然后把 MFR 转化为在该有向图中寻找两条没有公共顶点的最长有向路径问题，从而证明 MFR 是多项式可解的；随后 Lancia 等[17]通过相同的方法证明了对于无空隙的 SNP 矩阵，在片段互不包含的情况下，LHR 是多项式可解的。接着 Lancia 等[17]把无空隙的 SNP 矩阵 **M** 转化为 SNP 位点冲突图

G_S，证明了 **M** 可行当且仅当 G_S 是一个独立集（只有顶点没有边），从而把无空隙的 MSR 转化为最大独立集问题，然后证明 G_s 是一个完美图（Perfect Graph），最后基于在完美图中寻找最大独立集是多项式时间可解的事实证明了无空隙的 MSR 是有多项式时间算法的。

对于有空隙的 SNP 矩阵 **M**, 如果 **M** 满足连续 1 属性（Consecutive ones property, C1P），因为满足连续 1 属性的 SNP 矩阵可以在多项式时间内通过重排相关的列的次序调整为无空隙的 SNP 矩阵[132, 133]，因此对应的 MFR、MSR 和 LHR 也是多项式时间可解的。

对于一般的 SNP 矩阵 **M**，Lancia 等[17]把 **M** 转化成一个片段冲突图 G_F，**M** 可行当且仅当 G_F 是一个二部图。这样 MFR 就等价于删除 G_F 中最少的顶点使其二部化，而该问题是 NP-难的。然后进一步通过多项式时间归约于另外一个 NP-难 Max2SAT 问题证明了当 M 中的片段至多有一个空隙时，MFR 是 NP-难的。最后 Lancia 等[17]通过归约于 MAXCUT 问题可以证明当 M 中的片段至多有 2 个空隙时，MSR 是 NP-难的。

Rizzi 等[63]对 MSR 和 MFR 进一步进行深入研究，在 2002 年生物信息学算法会议（WABI）上提出了一些动态规划算法。对于 m 个 DNA 片段，n 个 SNP 位点的无空隙的 SNP 矩阵的 MSR 和 MFR 模型，Rizzi 等[63]提出了时间复杂度分别为 $O(mn^2)$和 $O(m^2n+m^3)$的多项式算法。对于有空隙的 SNP 矩阵，通过归约于图的最少边去除二部化（MinEdgeBipartizer）和最少顶点去除二部化（MinNodeBipartizer）这两个 APX-难问题，Rizzi 等[63]证明了 MFR 和 MSR 都是 APX-难。对于片段中洞的最大个数不超过 k 的 SNP 矩阵的 MSR 和 MFR 模型，Rizzi 等[63]提出了时间复杂度分别为 $O(mn^{2k+2})$和 $O(2^{2k}m^2n+2^{3k}m^3)$的算法。

2005 年，Bafna 等[62]在上述基础上，提出了类似的时间复杂度分别为 $O(mn^2)$和 $O(m^2n+m^3)$、空间复杂度分别为 $O(mn+n^2)$和 $O(mn+m^3)$的多项式算法求解无空隙的 MSR 和 MFR 模型；对于片段中洞的最大个数不超过 k 的 SNP 矩阵的 MSR 和 MFR 模型，Bafna 等[62]也提出了时间复杂度分别为 $O(mn^{2k+2})$和 $O(2^{2k}m^2n+2^{3k}m^3)$、空间复杂度分别为 $O(nm+2^kn^2)$和 $O(nm+2^{2k}m^3)$的动态规划算法。通过归约成顶点覆盖（Vertex cover）和最大割（Max cut）问题，Bafna 等[62]进一步证明了当片段中至多有一个空隙时，MSR 和 MFR 均是 APX-难的。

对于 LHR 模型，Cilibrasi 等[129, 130]证明了在有空隙的情况下，LHR 问题是 NP-hard 和 APX-hard。当 SNP 矩阵片段中没有空隙时，Cilibrasi[130]等设计了一个时间复杂度为 $O(n^2m + n^3)$的动态规划算法。

单体型组装问题除了上述三个计算模型外，Lippert 等[81]曾提出了下面第 4 个计算模型：

最少错误更正 (Minimum Error Correction, MEC) 或叫做最少字符翻转 (Minimum Letter Flips, MLF)：给定一个 SNP 矩阵 **M**，求翻转（'0'变成'1'，或'1'变成'0'）**M** 中的最少元素使 **M** 可行。

对于 MEC 模型，对于一般的 SNP 矩阵，Lippert 等[81]指出可以通过归约于最大割问题证明 MEC 是 NP-难的。Cilibrasi 等[130]则进一步证明了，即使对于无空隙的 SNP 矩阵，MEC 模型也是 NP-难的，并且证明了当片段中即使只有一个空隙时，MEC 模型也是 APX-hard。

为求解 MEC 模型的精确解，R. Wang 等[68, 73]曾设计过时间复杂度为 $O(2^m)$ 的分支限界算法。由于当片段数较多时，该精确算法缺乏实用性，因此有大批学者研究 MEC 模型的启发式算法。2004 年，Panconesi 等[134]提出一种快速的启发式算法（Fast hare），在其论文中 MEC 模型也叫做 MER（Minimum Element Removal）。2004 年和 2005 年，R. S. Wang 等[68, 73]曾提出过两种动态聚类算法和一种遗传算法。2008 年，Qian 等[135]提出一种粒子群算法。

在MEC（MLF）模型的基础上，Greenberg等[82]提出了下面第5个计算模型：

WMLF (weighted minimum letter flips) 模型：给定一个SNP矩阵**M**和一个对应的权值矩阵W，翻转**M**中的元素值（0变成1，或1变成0）使得**M**可行，且W中与翻转元素对应的权值之和最小。

对于WMLF模型，Zhao等[69]证明了其是NP-难的，并给出了一个动态聚类算法。

由于个体的基因型比较容易测定，2005年，Wang等[68]则在MEC模型的基础上加入个体的基因型信息，引入了第6个计算模型：

MEC/GI (MEC with genotype information)模型：给定一个SNP矩阵**M**和基因型G，翻转**M**中最少的元素（0变成1，或1变成0），目标是使得翻转后的SNP矩阵能由一对构成G的单体型导出。

对于MEC/GI模型，R. Wang等[68]曾设计一个遗传算法和一个时间复杂度为$O(2^m)$的分支限界算法。

2006年，Zhang等[72]提出了一个与MEC/GI类似的模型MCIH（Minimum Conflict Individual Haplotyping），证明了其是NP-难的，并提出了一个时间复杂度为$O(2^{2L}m)$的动态规划算法和一个神经网络算法FNN求解该模型，其中L为片段的最大长度。2007年，Y. Wang等[131]则提出了一个迭代的局部穷举搜索算法，

Genovese等[136]则针对高测序误差和低覆盖度提出了一个启发式算法。

与上述确定性计算模型不同，Li等[137]曾于2003年提出了一种统计方法以解决单体型重建问题，该统计算法以DNA片段的碱基组成是独立的，与周围碱基取值和所在区域的位置无关等6个假设作为前提，通过计算联配的片段上不同单体型的概率来确定相邻位点上的SNP值，然后组合连接成较长的单体型段。

从上面可以看出，对单体型组装问题的研究绝大部分是研究确定性计算模型。单体型组装问题的上述计算模型绝大部分被证明为 NP-难和 APX-难的，缺乏多项式时间的精确算法和有近似度保证的近视算法，而具有较快运行速度的启发式算法无法确保算法的精确度，单体型重建精度上往往与具体的实验方法密切相关，因此设计实用的精确算法具有重要的现实意义。

2.4 本章小结

本章对相关的分子遗传学背景知识及单体型组装问题的基本概念和研究现状进行介绍。首先对生物的遗传物质及遗传法则进行介绍，给出了单核苷酸多态性（Single Nucleotide Polymorphisms，SNP）、单体型（Haplotype）和基因型（Genotype）等相关概念，然后阐述了单体型组装问题相关的基本概念和定义，最后使用这些定义对单体型组装问题的计算模型进行重新表述，并对各模型的研究现状进行了概述。

第三章 MSR 和 MFR 参数化模型和算法

3.1 引言

Lancia 等[17]在引入最少 SNP 位点删除 (Minimum SNP Removal, MSR)和最少片段删除 (Minimum Fragment Removal, MFR)这两个单体型组装计算模型时就证明了当 SNP 矩阵无空隙时或满足连续 1 属性时，MSR 和 MFR 是多项式时间可解的；否则，MSR 和 MFR 是 NP-难的。后来 Bafna 等[62]对 MSR 和 MFR 进一步进行研究，对于 m 个 DNA 片段，n 个 SNP 位点的无空隙 SNP 矩阵的 MSR 和 MFR 模型，他们提出了时间复杂度分别为 $O(mn^2)$和 $O(m^2n+m^3)$、空间复杂度分别为 $O(mn+n^2)$和 $O(mn+m^3)$的多项式算法。对于有空隙的 SNP 矩阵，他们证明 MSR 和 MFR 均为 NP-难和 APX-难的。对于片段中洞的最大个数不超过 k 的 SNP 矩阵的 MSR 和 MFR 模型，Bafna 等[62]也提出了时间复杂度分别为 $O(mn^{2k+2})$和 $O(2^{2k}m^2n+2^{3k}m^3)$、空间复杂度分别为 $O(nm+2^kn^2)$和 $O(nm+2^{2k}m^3)$的动态规划算法。

当m和n不太大时，片段中洞很少时，Bafna等的上述算法是可行的，可是人有23对不同的染色体，染色体的平均长度约为150,000,000个碱基①，SNP的分布密度约为0.1%[1, 7]，这样一条染色体上的SNP位点数约为150,000。当从shotgun全基因组测序所得到的序列数据推出个体的单体型时，能直接测序的片段长度约为1000个碱基，在Celera公司的人类基因组测序工程中的片段数多达2727万[138]。在这种情况下，DNA片段数m和被测的SNP位点数n十分庞大，这时Bafna等的上述算法所需的时间和空间也将十分巨大，推断染色体一个区域的单体型时情况也是如此。因而根据目前基因组测序的技术现状，针对DNA测序数据和个体单体型问题的特点提出更高效的算法具有重要的现实意义。

3.2 MSR 和 MFR 参数化模型

目前最流行的SNPs探测方法是DNA直接测序[127, 139]，这种方法48小时可分析近百万个碱基对，杂合的SNPs探测率达95%以上，SNP联盟(SNP Consortium)

① http://www.ornl.gov/sci/techresources/Human_Genome/publicat/primer/prim1.html#3

用这种方法测得了上百万SNPs[7, 139]。当前DNA测序的主导方法是Sanger双脱氧链终止法[10]。采用Sanger双脱氧链终止法测序，一次能测定的DNA序列的长度仅为800-1200个碱基。各大测序中心使用的第三代测序仪如ABI 3730①、MageBACE②等可测片段长度约为1000碱基，而SNPs的平均分布密度约为1/1000[1, 7, 138]，虽然SNPs在整个染色体上的分布很不均匀，从已有的数据[47, 140, 141]来看，一个长度1000bp的片段上的SNP位点是极其有限的，通常在10个以内。

另外，因为测序的成本和所需的时间与覆盖度成正比，所以，实验室测序时片段的覆盖度(coverage)是不太大的。如Celera公司的人类基因组测序工程中的片段覆盖度为5.11[138]、国际人类基因组测序联盟（IHGSC）在人类基因组计划中的片段数据的覆盖度是4.5[1]。虽然在人类基因组的每个位置的片段覆盖程度并不完全相同，但是，Huson等[142]对Celera公司的人类基因组测序工程中的片段数据[138]进行了详细的分析，从其片段覆盖图上可看出绝大部分位点上的片段数为5左右，最大值没有超过19。从上可以看出，当需要测定某个体在一条染色体上的单体型时，覆盖一个SNP位点的DNA片段数远小于shotgun法测得的片段总数。因此，在DNA测序实验中，一个片段覆盖的SNP位点数 (通常小于10) 和覆盖一个SNP位点的片段数 (通常小于19) 与通常要探测的SNP位点数(n)及DNA片段总数(m)相比，均是很小的数。

基于以上事实，我们提出以下参数化条件：

定义3.1 (k_1, k_2)参数化条件：k_1、k_2是正整数，(k_1, k_2)参数化条件定义为片段覆盖的SNP位点数不超过k_1，覆盖任意SNP位点的片段数不超过k_2。

对一个无空隙的SNP矩阵**M**而言，(k_1, k_2)参数化条件等价为矩阵**M**的每一行最多只有一块连续非‘−’字符序列，该字符序列长度不超过k_1个；矩阵**M**每一列最多有k_2个行的取值为非‘−’字符。

根据(k_1, k_2)参数化条件，我们对无空隙的MSR问题和MFR问题参数化：

问题 3.1 P_MSR (Parameterized Minimum SNP Removal)：给定一个满足(k_1, k_2)参数化条件的无空隙的 SNP 矩阵 **M**，P_MSR 问题是要求删除最少的列，也就是保留最多的列使 **M** 可行。

问题 3.2 P_MFR (Parameterized Minimum Fragment Removal)：给定一个满足(k_1, k_2)参数化条件的无空隙的 SNP 矩阵 **M**，P_MFR 问题是要求删除最少的

① http://www.fungen.org/UseABI3700.htm
② http://www.ebiotrade.com/ custom/amersham/MegaBACE.htm

行，也就是保留最多的行使 **M** 可行。

本章中，一个无空隙的 SNP 矩阵 **M** 的 P_MSR 解指的是使 **M** 可行能保留下来的最多的列数，用 P_MSR(**M**)来表示；一个无空隙的 SNP 矩阵 **M** 的 P_MFR 解指的是使 **M** 可行能保留下的最多的行数，用 P_MFR(**M**)来表示。

在某些情况下，DNA 测序仪对 DNA 片段的某些碱基无法确定其值，如果这些碱基在 SNP 位点上，对应的 SNP 矩阵中的行在其覆盖的某些列上就会取空值。另外，如果在 DNA 测序中采用了 mate-pair，则 mate-pair 片段中间也会出现一段连续的空值。片段上的空值叫做洞（hole），而 1 个空隙（gap）指的是一段连续的洞。在图 3-1 中，行 2 对应的片段在列 4、5 上各有一个洞，这两个连续的洞形成一个空隙，也就是说行 2 有 2 个洞，但只有 1 个空隙。

对于有空隙的 SNP 矩阵，我们提出另一个参数化条件：

定义3.2　(k_1, k_2, k)参数化条件：k_1、k_2、k均为正整数，(k_1, k_2, k)参数化条件定义为片段覆盖的SNP位点数不超过k_1，覆盖任意SNP位点的片段数不超过k_2，任意片段中洞的个数不超过k。

对一个SNP矩阵**M**而言，(k_1, k_2, k)参数化条件等价为矩阵**M**的每一行覆盖的列数k_1，其中最多有不超过k列上的值是空值（‘–’），另外矩阵**M**每一列最多有k_2个行覆盖。例如，图3-1中的SNP矩阵满足（6, 6, 2）参数化条件，其中第2行和第4行覆盖6列，他们都有2个洞，第4列有6行覆盖，它们是行2到行7。

SNP 位点

片段								
-	-	-	-	-	1	0	1
-	1	0	-	-	0	1	-
0	1	0	0	-	-	-	-
1	-	1	1	-	1	-	-
-	0	1	1	-	-	-	-
-	-	0	1	-	-	-	-
-	-	-	0	1	0	1	0
-	-	-	-	-	1	0	1

图 3-1　SNP 矩阵

下面我们对有空隙的MSR问题和MFR问题参数化：

问题 3.3　PG_MSR (Parameterized Minimum SNP Removal with Gaps)：给定一个满足(k_1, k_2, k)参数化条件的 SNP 矩阵 **M**，PG_MSR 问题是要求删除最少的

列，也就是保留最多的列使 **M** 可行。

问题 3.4 PG_MFR (Parameterized Minimum Fragment Removal with Gaps)：给定一个满足(k_1, k_2, k)参数化条件的 SNP 矩阵 **M**，PG_MFR 问题是要求删除最少的行，也就是保留最多的行使 **M** 可行。

本章中，一个 SNP 矩阵 **M** 的 PG_MSR 解指的是使 **M** 可行能保留下来的最多的列数，用 PG_MSR(**M**)来表示；一个 SNP 矩阵 **M** 的 PG_MFR 解指的是使 **M** 可行能保留下的最多的行数，用 PG_MFR(**M**)来表示。

本文下面深入研究MSR和MFR的参数化算法。

3.3 MSR 和 MFR 参数化算法

3.3.1 预处理

求解问题 3.1~3.4 前，先对 SNP 矩阵 **M** 进行如下预处理：

Step 1 排序：调整 **M** 中各行的次序，使各行按其覆盖的最左边的列号即 *left* 值进行非降序排列。 同时对于任意列 j，计算出覆盖该列的行的有序集，记作 *rowset*(j)。

对于一个满足(k_1, k_2)或(k_1, k_2, k)参数化条件的 SNP 矩阵 **M**, 排序后如图 3-2 所示。

Step 2 去掉冗余列：对 **M** 的每一列 j，如果 *rowset*(j)中的所有行在该列的取值中没有出现‘0’或没有出现‘1’，那么该列就是冗余列，标记该列。最后去掉所有标记的列，调整剩下列的序号，修改对应行的 *left* 和 *right* 函数值。

Step 3 去掉冗余行：对 **M** 的每一行 i，如果该行在剩下的列上的取值均为空，则该列就是冗余行，标记该行。最后去掉所有标记的行，修改受影响的列的 *rowset* 集。

令 **M** 采用如下数据结构：对每一行 i，记下该行覆盖的最左边和最右边的列号，即 *left*(i)和 *right*(i)，记录该行从列 *left*(i)到列 *right*(i)的各列上的值。

预处理时间复杂度分析：排序所需的时间为 $O(mlogm)$，对排好序的行扫描一遍可得到各列的 *rowset* 有序集，这样 Step 1 所需时间为 $O(mlogm+mk_1)$。判断列 j 是否冗余，可以对 *rowset*(j)中的行在列 j 上的取值进行统计，所需的时间为 $O(k_2)$，这样标记出所有的冗余列需时间 $O(nk_2)$。去掉冗余列，可以先从列 1 到

n 扫描一遍，累计每一列前面有多少列被标记，所需时间为 $O(n)$，然后对所有的行扫描一遍，去掉被标记的列上的值，对片段的起始列号和终止列号进行修改所需的时间为 $O(mk_1)$。行的起始列号和终止列号的修改方法为：起始列号减去起始列前被标记的列数得到新的起始列号；如果终止列没有被标记，则终止列号减去终止列前被标记的列数得到新的终止列号，如果被标记，则再减 1，如果片段所覆盖的所有列都被标记，该片段新的终止列号就会比新的起始列号小 1。 这样，Step 2 所需时间为 $O(nk_2+mk_1)$。然后再对所有的行扫描一次，对冗余行进行标记所需的时间为 $O(mk_1)$，再检查每一列的 *rowset* 中的行，如果是冗余行，则从 *rowset* 中去掉，所需时间为 $O(nk_2)$。这样 Step 3 所需时间也是 $O(nk_2+mk_1)$。所以整个预处理所需的时间为 $O(mlogm+nk_2+mk_1)$。

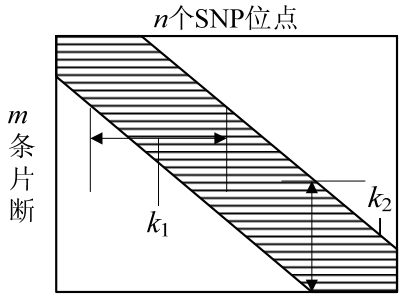


图 3-2 预处理后的 SNP 矩阵

定理 3.1 一个满足(k_1, k_2)或(k_1, k_2, k)参数化条件的 SNP 矩阵 **M** 经过以上预处理后仍然满足(k_1, k_2) 或(k_1, k_2, k)参数化条件。

证明：排序、去掉冗余列和冗余行的预处理显然不会增加某一行覆盖的列数、某一行中的洞数和覆盖某一列的行数。定理得证。 □

定理 3.2 一个 SNP 矩阵 **M** 经过以上预处理后得到的 SNP 矩阵记作 **M′**，去掉的冗余列集和冗余行集分别记作 **X** 和 **Y**，则 **M** 的 P_MSR（或 PG_MSR）解等于 **M′**的 P_MSR（或 PG_MSR）解与 **X** 的列数之和，**M** 的 P_MFR（或 PG_MFR）解等于 **M′**的对应解与 **Y** 的行数之和。

证明：首先证明 **M** 的 P_MSR（或 PG_MSR）解等于 **M′**的 P_MSR（或 PG_MSR）解与 **X** 的列数之和。因为去掉的列在所有的行上的取值只含有‘A’或‘B’，所以 **M** 的任意一行均不会在去掉的列上和其他行冲突。令 **S** 是 **M′**的一些或全部列构成的集合，容易看出，如果保留 **S** 中的所有列可使 **M′**可行，那么

保留 **S**∪**X** 中的所有列也可使 **M** 可行。由此可证 **M** 的 P_MSR（或 PG_MSR）解等于 **M′**的 P_MSR（或 PG_MSR）解与 **X** 的列数之和。

同理 **Y** 中的行不会与其他任何行冲突，因此 **M′**的 P_MFR（或 PG_ MFR）解中所保留下的行加上 **Y** 中的行所形成的 SNP 矩阵肯定是可行的，因此可以证明 **M** 的 P_MFR（或 PG_ MFR）解等于 **M′**的对应解与 **Y** 的行数之和。定理得证。　□

从定理 3.2 可知，求解一个 SNP 矩阵 **M** 的 P_MSR 和 P_MFR（或 PG_MSR 和 PG_MFR）解可以归结为求其预处理后的 SNP 矩阵 **M′**的对应解。一旦求出了 **M′**的 P_MSR 和 P_MFR（或 PG_MSR 和 PG_MFR）解，**M** 的对应解可在 $O(\max(m, n))$时间内得到。对于行列数为 0 的 SNP 矩阵，易知其 P_MSR 和 P_MFR（或 PG_MSR 和 PG_MFR）解均为 0。为了叙述的简洁，本章下面讨论的 SNP 矩阵 **M**，除特别声明外，均指预处理后的 SNP 矩阵，该矩阵具有如下特点：行列数 m、n>0；任何一个列总有一些行的值是‘0’，还有一些行的值为‘1’；对于行 $i<j$, 有 $left(i)\le left(j)$。

3.3.2　P_MSR 参数化算法

在阐述求解问题 3.1 即 P_MSR 问题的参数化算法前，本文先引入列之间**冲突、兼容**的相关概念。

定义 3.3　对于一个 SNP 矩阵 **M**（不管有无空隙，是否经过预处理）的两列 j_1 和 j_2，如果 **M** 的所有行都可以划分到两个子集中，使得划分在同一个子集中的任意两行在列 j_1 和 j_2 上均不冲突，则列 j_1 和 j_2 兼容，满足上述条件的划分叫做在列 j_1 和 j_2 上兼容的划分；如果不存在着在列 j_1 和 j_2 上兼容的划分，则列 j_1 和 j_2 冲突。

SNP 位点

片段								
	-	-	-	-	-	1	0	1
	-	1	0	0	-	0	1	-
	0	1	0	0	-	-	-	-
	1	0	1	1	0	1	-	-
	-	0	1	1	-	-	-	-
	-	-	0	1	-	-	-	-
	-	-	-	0	1	0	1	0
	-	-	-	-	-	1	0	1

图 3-3　SNP 矩阵

在图 3-3 所示的 SNP 矩阵中，如果要在列 3 上不冲突，行 2、3 和 6 必须划分到同一子集中，行 4 和 5 则必须划分到另一子集中；但是这样划分的话，行 6 与行 2 在列 4 上冲突，由此可见，列 3 和 4 冲突。对于列 1 和列 2 而言，把行 2 和 3 分到一个子集中，行 4 和 5 分到另一个子集中的任意划分都是在列 1 和 2 上兼容的划分，所以列 1 和列 2 兼容。

定理 3.3　对于一个 SNP 矩阵 **M**（不管有无空隙，是否经过预处理），只有‘0’或‘1’值的任意列不会和其他列冲突。对于既有‘0’又有‘1’值的列 j_1、j_2，列 j_1、j_2 冲突当且仅当存在着两行 i_1、i_2：$i_1 \neq i_2$，有 $\mathbf{M}_{i_1,j_1}$、$\mathbf{M}_{i_1,j_2}$、$\mathbf{M}_{i_2,j_1}$ 和 $\mathbf{M}_{i_2,j_2}$ 均不为空值，且这 4 个值之中有 3 个相同、1 个不同。

证明：不失一般性，令列 j_1 只有‘0’值，对其他任意列 j_2，**M** 中的所有行均可如下划分：在列 j_2 上取‘0’值的行划分到第一个集合中，在列 j_2 上取‘1’值的行划分到第二个集合中，其他的行随机划分到这两个集合中，容易验证，这样划分到同一个集合中的任意两行在列 j_1 和 j_2 上不会冲突。因此如果一列只有‘0’或‘1’值，该列不会和其他任何列冲突。

对于既有‘0’又有‘1’值的列 j，如果要使划分在同一个子集中的行在列 j 上不冲突，则在列 j 上取值为‘0’的所有行必须划分在同一子集中，在列 j 上取值为‘1’的行必须划分在另一个子集中。对于既有‘0’又有‘1’值的两列 j_1、j_2，如果存在着两行 i_1、i_2，有 $\mathbf{M}_{i_1,j_1}$、$\mathbf{M}_{i_1,j_2}$、$\mathbf{M}_{i_2,j_1}$ 和 $\mathbf{M}_{i_2,j_2}$ 均不为空值，且这 4 个值之中有 3 个相同、1 个不同，不失一般性，令 $\mathbf{M}_{i_1,j_1}=\mathbf{M}_{i_2,j_1}=\mathbf{M}_{i_1,j_2}=$‘0’、$\mathbf{M}_{i_2,j_2}=$‘1’，那么按列 j_1 上的取值，行 i_1 和 i_2 应该划分到同一个子集中；而按列 j_2 上的取值，行 i_1 和 i_2 应该划分到不同的子集中，这相互矛盾，说明不存在列 j_1 和 j_2 上兼容的划分。因此列 j_1, j_2 冲突。

反过来，对于既有‘0’又有‘1’值的两列 j_1、j_2，如果列 j_1 和 j_2 不冲突，那么存在着一个划分使 **M** 的所有行划分到两个子集中，使得相同子集中的任意两行在列 j_1 和 j_2 上不冲突。这样对任意两行 i_1, i_2, 如果有 $\mathbf{M}_{i_1,j_1}$、$\mathbf{M}_{i_1,j_2}$、$\mathbf{M}_{i_2,j_1}$ 和 $\mathbf{M}_{i_2,j_2}$ 均不为空值，即不是‘0’值就是‘1’值，那么如果这两行被划分到同一个子集中，则这两行在列 j_1 上的值相等，在列 j_2 上的值也应相等；如果这两行被划分到不同的子集中，则这两行在列 j_1 上的值不相等，在列 j_2 上的值也应不相等。这样这 4 个值之中就不可能有 3 个相同、1 个不同的情况。定理得证。 □

定理 3.4 [17] 一个无空隙的 SNP 矩阵 **M** 是可行的当且仅当其任意两列均不

冲突。

定理 3.4 的证明请参看文献[17]。由定理 3.4 可知 P_MSR 问题等价于保留最多的列，使保留的列中的任意两列均不冲突。

令 C 是 **M** 一个列的子集，如果 C 中的列彼此兼容，且 C 中的列的最大列号为 j，则 C 称为列 j 的兼容列集。对于列 j 的任意兼容列集 C，容易看出保留 C 中的所有列可使 **M** 可行。令 $K(j)$表示列 j 的最大兼容列集。显然：

$$K(1) = \{1\} \tag{3-1}$$

由文献[17]的引理 2 可知对于一个无空隙的 SNP 矩阵 **M** 的任意 3 列 $j_1<j_2<j_3$，如果 j_1 与 j_2 兼容，j_2 与 j_3 兼容，则一定有 j_1 与 j_3 兼容。因此对于任意两列 r、j：$r<j$，如果列 r 和 j 兼容，则 $K(r) \cup \{j\}$一定是列 j 的兼容列集，由此容易证明 $K(j) = \{j\} \cup \max_{r:1\le r<j,\text{列}r\text{和}j\text{兼容}} K(r)$，其中对集合进行的 max 运算是取元素最多的集合，即 $\max_{r:1\le r<j,\text{列}r\text{和}j\text{兼容}} K(r)$ 为满足 $1\le r<j$ 且列 r 和 j 兼容的所有 $K(r)$中的最大集合，如果没有满足这些条件的 $K(r)$，则为空集∅，下文也是如此。

对于列 j 和它前面的列 r，下面判断列 j 是否和 r 相容：

Case 1 $r \le j-k_1$： 因为 **M** 满足(k_1, k_2)参数化条件，所以 **M** 的任一行覆盖的列数不会超过 k_1，这样就不可能存在着两行 i_1、i_2：$i_1 \ne i_2$，且使得 $\mathbf{M}_{i_1, j_1}$、$\mathbf{M}_{i_1, j_2}$、$\mathbf{M}_{i_2, j_1}$ 和 $\mathbf{M}_{i_2, j_2}$ 均不为空值。根据定理 3.3，列 r 与 j 兼容。

Case 2 $r > j-k_1$： 因为 **M** 经过了预处理，所以任意一列上既有‘0’又有‘1’。根据定理 3.3，只需对在列 r 与 j 上取值均非空的行进行观察，这些行的个数不大于 k_2。如果这些行在这两列上的取值只有“00”、“11”两种类型，或只有“01”、“10”两种类型，那么列 r 与 j 兼容；否则就不兼容。

令 $A(j)$表示 $K(1)$到 $K(j)$的最大值, 如果 $j<0$, 规定 $A(j)=\varnothing$。

令 $OK(j) = \{ r \mid j > r > j-k_1$ 且列 r 与 j 兼容$\}$。由上面的讨论，易知下面的公式成立：

$$K(j) = \{j\} \cup \max(A(j-k_1), \max_{r \in OK(j)}(K(r))) \tag{3-2}$$

由 P_MSR 问题的定义可知:

$$\text{P\_MSR}(\mathbf{M}) = \max(|A(n-k_1)|, |K(n-k_1+1)|, \ldots, |K(n)|) \tag{3-3}$$

在预处理的基础上，根据公式(3-1)到(3-3)，可得到下面求解无空隙 SNP 矩

阵 **M** 的 P_MSR 问题的动态规划算法。

P_MSR 算法：

Input：预处理后得到的无空隙 SNP 矩阵 **M**，令其行数为 m, 列数为 n

Output：**M** 的 P_MSR 解

Step 1. 扫描 **M** 得到一行覆盖的最大列数 k_1 和覆盖一列的最大行数 k_2；
$K(1)= A(1)=\{1\}$; //公式(3-1)

Step 2. **for** $j = 2 .. n$ **do** //根据公式(3-2)递推

Step 2.1. **if** $j \le k_1$ **then** $Kp = \varnothing$; **else** $Kp = A(j-k_1)$;

Step 2.2. **for** $r = \max(1, j-k_1) .. j-1$ **do**

Step 2.2.1. **if** 列 r 与 j 兼容 **and** $|Kp| < |K(r)|$ **then** $Kp= K(r)$;

Step 2.3. $K(j) = \{j\} \cup Kp$;

Step 3. **if** $n \le k_1$ **then** $Kp = \varnothing$ **else** $Kp = A(n-k_1)$;

Step 4. **for** $r = \max(1, n-k_1) .. n$ **do** //公式(3-3)

Step 4.1. **if** $|Kp|<|K(r)|$ **then** $Kp= K(r)$;

Step 5. **return** $|Kp|$;

定理 3.5 对于一个 $m\times n$ 满足(k_1, k_2)参数化条件的无空隙 SNP 矩阵 **M**，P_MSR 算法是正确的，加上预处理其时间复杂度为 $O(nk_1k_2+mlogm+mk_1)$，空间复杂度为 $O(mk_1+nk_1)$。

证明: P_MSR 算法的正确性由公式(3-1)~(3-3)的正确性来保证，而这些公式的正确性在引入时已经得到了证明。下面分析其复杂度。

时间复杂度分析：Step 1 中扫描 **M** 得到 k_1 和 k_2 所需的时间为 $O(mk_1)$；Step 2.2.1 中判断列 r 与 j 是否兼容所需时间为 $O(k_2)$，Step 2.2.1 最多执行 nk_1 次，由此可知 Step 2 的时间复杂度为 $O(nk_1k_2)$；Step 3 的时间复杂度为 $O(1)$；Step 4 的时间复杂度为 $O(k_1)$。这样加上预处理的时间整个算法的时间复杂度为 $O(nk_1k_2+mlogm+mk_1)$。

空间复杂度分析：SNP 矩阵所需的空间为 $O(mk_1)$, 计算 $K(j)$只需要 $A(j-k_1)$、$K(j-k_1+1)$、... 、$K(j-1)$的值，因此 A 和 K 均可采用大小为 k_1 的循环队列，需要的空间 $O(nk_1)$，所以整个算法的空间复杂度为 $O(mk_1+nk_1)$。定理证毕。 □

3.3.3 P_MFR 参数化算法

本节阐述求解问题 3.2 即 P_MFR 的参数化算法。

考虑 $m\times n$ 的 SNP 矩阵 **M** 的前 i 行构成的矩阵 **M**(i, :)，令 R 为 **M**(i, :)的行的子集，如果保留 R 中的所有行能使 **M**(i, :)可行，那么 R 就称 **M**(i, :)的一个兼容行集。

如果 R 是 **M**(i, :)的一个兼容行集，则必定可以把 R 划分成两个子集 R_1 和 R_2，使得划分在同一子集的行彼此兼容。对于 R_j：j=1,2, 找出具有以下特性的行 r_j 作其代表：(1) $r_j\in R_j$; (2) 对于任意行 $r\in R_j$，有 $right(r)<right(r_j)$，或者 $right(r)=right(r_j)$且 $r<r_j$。在“$right(r_1)<right(r_2)$” 或 “$right(r_1)=right(r_2)$且 $r_1<r_2$”的情况下，R 叫做行 r_1, r_2 代表的 **M**(i, :)的兼容行集；否则 R 叫做行 r_2, r_1 代表的 **M**(i, :)的兼容行集。

划分得来的子集在极端情况下可能是空集，为了上述概念的完整性，规定空集的代表行为 0，而且 $left(0)=-1$，$right(0)=-2$，行 0 与所有其他行均兼容。显然空集也应是 **M**(i, :)的一个兼容行集，所以进一步规定空集是行 0, 0 代表的 **M**(i, :)的兼容行集。令 $R(d, k, i)$表示以行 d, k 为代表的 **M**(i, :)的最大兼容行集。易知：

$$R(0, 0, 1)=\varnothing,\quad R(0, 1, 1)=\{1\} \tag{3-4}$$

令 $R(*, k, i)$表示所有满足 $right(d)<left(i+1)$的最大 $R(d, k, i)$，即 $R(*, k, i)=\max_{d:right(d)<left(i+1)} R(d,k,i)$。令 $R(*, *, i)$表示所有满足 $right(k)<left(i+1)$的最大 $R(*, k, i)$，即 $R(*, *, i)=\max_{k:right(k)<left(i+1)} R(*,k,i)$。这样就有：

$$R(*, 0, 1)=\varnothing;\ R(*, 1, 1)=\{1\};\ R(*,*, 1)=\begin{cases}\{1\}, & \text{if } right(1)<left(2)\\ \varnothing, & \text{else}\end{cases} \tag{3-5}$$

为了使 $R(*, k, i)$、$R(*,*, i)$在 $i=m$ 时有意义，引入行 m+1, 规定 $left(m+1)=n+1$。这样，根据 P_MFR 的定义，显然有：

$$\text{P\_MFR}(\mathbf{M})=|\,R(*,*, m)\,|,\ \text{即 } R(*,*, m)\text{中的行数} \tag{3-6}$$

定理 3.6 对于一个预处理后的无空隙 SNP 矩阵 **M**，行 i+1 与 $R(d, k, i)$中以行 d（或 k）为代表的子集中所有行兼容当且仅当行 i+1 与行 d（或 k）兼容。

证明：如果行 i+1 与 $R(d, k, i)$中以行 d（或 k）为代表的子集中所有行兼容，那么显然行 i+1 与行 d（或 k）兼容。

下面证明如果行 i+1 与行 d 兼容，则行 i+1 与 $R(d, k, i)$中以行 d 为代表的子集中所有行兼容：对于 $R(d, k, i)$中以行 d 为代表的子集中任意一行 r, 必有 $right(r) \le right(d)$，且行 r 与 d 在列 $\max(left(r), left(d))$到列 $right(r)$中的任一列上取值必定非空（无空隙），而且相等。由于 **M** 是经过排序的，且 $r, d < i$+1, 所以 $\max(left(r),left(d)) \le left(i+1)$。由于 **M** 是无空隙的 SNP 矩阵，且行 i+1 与行 d 兼容，所以行 i+1 与行 d 在列 $left(i+1)$到列 $\min(right(i+1), right(d))$中的任一列上取值必定相等，而且非空。这样行 i+1 与行 r 在列 $left(i+1)$到列 $\min(right(i+1), right(r))$中的任一列上取值必定相等，而在其他的列，总有一行在该列的值为空，因此行 i+1 与行 r 在任意列上均不冲突，行 i+1 与行 r 兼容。

同样可以证明如果行 i+1 与行 k 兼容，则行 i+1 与 $R(d, k, i)$中以行 k 为代表的子集中所有行兼容。定理得证。 □

为了行文简洁，满足下列 2 个条件的行 k 的集合记作 $S_k(i)$：

(1) $k < i$;

(2) $right(k) \ge left(i)$。

满足下列 4 个条件的(d,k)的集合记作 $S_{d,k}(i)$：

(1) $d, k < i$;

(2) $right(d) \ge left(i)$； (3) $right(k) \ge left(i)$;

(4) “$right(d) < right(k)$”或“$right(d) = right(k)$且 $d < k$”。

令 $C(i) = \{ r \mid r \in S_k(i),$ 且行 r 与 i 兼容$\}$。

下面讨论对于行 i: $2 \le i \le m$, 在 $R(*, *, i-1)$、所有的 $R(*, k, i-1)$：$k \in S_k(i)$和所有的 $R(d, k, i-1)$：$(d, k) \in S_{d,k}(i)$都已知的条件下，如何求出 $R(*, *, i)$、所有可能的 $R(*, k, i)$：$k \in S_k(i+1)$和所有的 $R(d, k, i)$：$(d, k) \in S_{d,k}(i+1)$。

首先对于所有的 $(d, k) \in S_{d,k}(i)$, 计算 $R(d, k, i)$：

$R(d, k, i-1)$显然是一个以行 d, k 为代表的 $\mathbf{M}(i, :)$的兼容行集，而 $\mathbf{M}(i,:)$比 $\mathbf{M}(i-1, :)$多的行只有行 i, 易知 $R(d, k, i)$比 $R(d, k, i-1)$最多只会增加一个元素，增加的元素只可能是行 i。在满足“i 与 d 兼容， $right(i) < right(d)$”或“i 与 k 兼容，$right(i) < right(k)$”的条件下，根据定理 3.6，$\{i\} \cup R(d, k, i-1)$显然是一个以行 d, k 为代表的 $\mathbf{M}(i, :)$的兼容行集，所以有：

$$R(d, k, i) = \{i\} \cup R(d, k, i\text{-}1); \tag{3-7}$$

否则，必定有：

$$R(d, k, i) = R(d, k, i-1)。 \tag{3-8}$$

这是因为如下事实：条件“i 与 d 兼容, $right(i) < right(d)$”得不到满足，则行 i 如果划分到行 d 代表的子集中，那么或者 d 所在的子集中的行不再相互兼容，或者 d 所在的子集的代表不再是 d 而应该是 i。而条件“i 与 k 兼容, $right(i) < right(k)$”得不到满足，那么或者 k 所在的子集中的行不再相互兼容，或者 k 所在的子集的代表不再是 k 而应该是 i。这两个条件都得不到满足，则行 i 必定不能在以 d, k 为代表的 $\mathbf{M}(i, :)$的兼容行集之中。

对于所有的 $r \in S_k(i)$：

如果 $right(r) \le right(i)$，有下面的公式成立(证明见定理 3.7)：

$$R(r,i, i) = \{i\} \cup \max\Big(R(*, r, i-1), \max_{d: d\in C(i), (d,r)\in S_{d,k}(i)} R(d,r,i-1), \max_{k: k\in C(i), (r,k)\in S_{d,k}(i), right(k)\le right(i)} R(r,k,i-1) \Big) \tag{3-9}$$

否则，即 $right(r) > right(i)$, 有下面的公式成立(证明见定理 3.7)：

$$R(i,r, i) = \{i\} \cup \max\Big(R(*, r, i-1), \max_{d:d\in C(i),(d,r)\in S_{d,k}(i),right(d)\le right(i)} R(d,r,i-1) \Big) \tag{3-10}$$

对于所有的 $k \in S_k(i)$，计算 $R(*, k, i)$：

令 I 表示 $\max_{d:right(d)<left(i)} R(d,k,i)$，$I_{d,k}(i)$表示 $S_{d,k}(i) \cup \{ (i, k) \mid k \in S_k(i), right(k) > right(i) \}$。根据 $R(*, k, i)$的定义：$R(*, k, i) = \max(\mathrm{I}, \max_{d:left(i)\le right(d)<left(i+1)} R(d,k,i))$ $= \max(\mathrm{I}, \max_{d:(d,k)\in I_{d,k}(i),right(d)<left(i+1)} R(d,k,i))$。对于所有的 $d < left(i)$, 如果 k 与 i 兼容且 $right(k) > right(i)$，有 $R(d, k, i) = R(d, k, i\text{-}1) \cup \{i\}$ (理由与公式(3-7)相同)，$\mathrm{I} = \max_{d:right(d)<left(i)} R(d,k,i-1) \cup \{i\} = \{i\} \cup R(*, k, i-1)$；否则有 $R(d, k, i) = R(d, k, i\text{-}1)$

(理由与公式(3-8)相同)， $I = \max_{d:right(d)<left(i)} R(d,k,i-1) = R(*,k,i-1)$。 因此有下面公式成立。

如果 k 与 i 兼容且 $right(k) > right(i)$：

$$R(*, k, i) = \max(R(*,k,i-1) \cup \{i\}, \max_{d:(d,k)\in I_{d,k}(i),right(d)<left(i+1)} R(d,k,i)) \quad (3\text{-}11)$$

否则，必定有：

$$R(*, k, i) = \max(R(*, k, i-1), \max_{d:(d,k)\in I_{d,k}(i),right(d)<left(i+1)} R(d,k,i)) \quad (3\text{-}12)$$

最后计算 $R(*, i, i)$和 $R(*, *, i)$ (证明见定理 3.7)：

$$R(*, i, i) = \max(\{i\} \cup R(*, *, i-1), \{i\} \cup \max_{k:k\in C(i),right(k)\le right(i)} R(*,k,i-1), \max_{d:d\in S_k(i),right(d)\le right(i),right(d)<left(i+1)} R(d,i,i)) \quad (3\text{-}13)$$

$$R(*, *, i) = \max(R(*, *, i-1), \max_{k:k\in S_k(i)\cup\{i\},right(k)<left(i+1)} R(*,k,i)) \quad (3\text{-}14)$$

由于 $S_k(i+1)$与 $S_k(i)$的差集或者是空或者是$\{i\}$，$S_{d,k}(i+1)$与 $S_{d,k}(i)$的差集最多是$\{ (d, i) \mid d \in S_k(i), right(d) \le right(i) \} \cup \{ (i, k) \mid k \in S_k(i), right(i) < right(k) \}$，所以通过公式(3-7) ~ (3-14)可以求出 $R(*, *, i)$、所有可能的 $R(*, k, i)$：$k \in S_k(i+1)$ 和所有的 $R(d, k, i)$：$(d, k) \in S_{d,k}(i+1)$。

根据公式(3-4) ~ (3-14), 可得出下面的 P_MFR 动态规划算法。

P_MFR 算法：

Input：预处理后的无空隙 SNP 矩阵 **M**，令其行数为 m, 列数为 n

Output：**M** 的 P_MFR 的解

Step 1. $R(0, 0, 1) = \varnothing$; $R(0, 1, 1) = \{1\}$; //公式(3-4)

$R(*, 0, 1) = \varnothing$; $R(*, 1, 1) = \{1\}$; //公式(3-5)

Step 2. **if** $right(1)<left(2)$ **then** $R(*, *, 1)=\{1\}$;

else $R(*, *, 1)= \varnothing$; //公式(3-5)

Step 3. **for** $i = 2 .. m$ **do**

Step 3.1. **for** each $k \in S_k(i)$ **do**

Step 3.1.1. **if** $right(k) \le right(i)$ **then** $R(k,i,i) = \{i\} \cup R(*,k,i-1)$;

//用公式(3-9)第 1 项初始化

else $R(i,k,i) = \{i\} \cup R(*,k,i-1)$; //用公式(3-10)第 1 项初始化

Step 3.1.2. $R(*,k,i) = R(*,k,i-1)$; //用公式(3-12)第 1 项初始化

Step 3.1.3. **if** 行 k 与 i 兼容 **then** $C(k)=1$; **else** $C(k)=0$;

Step 3.2. $R(*,i,i) = \{i\} \cup R(*,*,i-1)$; //用公式(3-13)第 1 项初始化

$R(*,*,i) = R(*,*,i-1)$; //用公式(3-14) 第 1 项初始化

Step 3.3. **for** each $(d,k) \in S_{d,k}(i)$ **do**

Step 3.3.1. **if** ($C(d) = 1$, $right(i) < right(d)$) or ($C(k) = 1$, $right(i) < right(k)$)

then $R(d,k,i) = \{i\} \cup R(d,k,i-1)$; //公式(3-7)

else $R(d,k,i) = R(d,k,i-1)$; //公式(3-8)

Step 3.3.2. **if** $C(d) = 1$, $right(k) \le right(i)$ **then**

$R(k,i,i) = \max(R(k,i,i), \{i\} \cup R(d,k,i-1))$; //公式(3-9)第 2 项

Step 3.3.3. **if** $C(k) = 1$, $right(k) \le right(i)$ **then**

$R(d,i,i) = \max(R(d,i,i), \{i\} \cup R(d,k,i-1))$; //公式(3-9)第 3 项

Step 3.3.4. **if** $C(d) = 1$, $right(d) \le right(i) < right(k)$ **then**

$R(i,k,i) = \max(R(i,k,i), \{i\} \cup R(d,k,i-1))$; //公式(3-10)第 2 项

Step 3.4. **for** each $k \in S_k(i)$ **do**

Step 3.4.1. **if** $C(k) = 1$, $right(k) > right(i)$ **then**

$R(*,k,i) = \{i\} \cup R(*,k,i-1)$; //公式(3-11)第 1 项

Step 3.4.2. **if** $C(k) = 1$, $right(k) \le right(i)$ **then** //公式(3-13)第 2 项

$R(*,i,i) = \max(R(*,i,i), \{i\} \cup R(*,k,i-1))$;

Step 3.5. **for** each $(d,k) \in S_{d,k}(i)$ **do**

Step 3.5.1. **if** $right(d) < left(i+1)$ **then** //公式(3-11)和(3-12)第 2 项

$R(*,k,i) = \max(R(*,k,i), R(d,k,i))$;

Step 3.6. **for** each $r \in S_k(i)$ **do**

Step 3.6.1. **if** $right(i) < right(r)$, $right(i) < left(i+1)$ **then**

$R(*,r,i) = \max(R(*,r,i), R(i,r,i))$; //公式(3-11)和(3-12)第 2 项

Step 3.6.2. **if** $right(r) \le right(i)$, $right(r) < left(i+1)$ **then**

$R(*,i,i) = \max(R(*,i,i), R(r,i,i))$; //公式(3-13)第 3 项

Step 3.7. **for** each $k \in S_k(i) \cup \{i\}$ **do**

Step 3.7.1. **if** *right*(*k*) < *left*(*i*+1) **then**

R(*,*,*i*)=max(*R*(*,*, *i*), *R*(*,*k*, *i*));　　//公式(3-14)第 2 项

Step 4. **return** | *R*(*, *, *m*) |;　　//公式(3-6)

定理 3.7　对于一个预处理后满足(k_1, k_2)参数化条件的无空隙 $m\times n$ SNP 矩阵 **M**，P_MFR 算法是正确的，加上预处理，其时间复杂度为 $O(mk_2^2+mk_1k_2+mlogm+nk_2)$，空间复杂度为 $O(mk_1+mk_2^2)$。

证明: P_MFR 算法的正确性取决于公式(3-4) ~ (3-14)的正确性。公式(3-4) ~ (3-8)，(3-11)和(3-12)的证明在公式的引入时已经给出，下面证明其他公式。

在公式(3-9)中，显然只有当 *right*(*r*) ≤ *right*(*i*)时，*R*(*r, i, i*)才会有意义。

令 I_1 表示 *R*(*, *r*, *i*−1)，I_2 表示 $\max_{d:d\in C(i),(d,r)\in S_{d,k}(i)} R(d,r,i-1)$，$I_3$ 表示 $\max_{k:k\in C(i),(r,k)\in S_{d,k}(i),right(k)\le right(i)} R(r,k,i-1)$。

首先证明$\{i\}\cup I_1$，$\{i\}\cup I_2$，$\{i\}\cup I_2$ 都是以 *r, i* 为代表的 M(:,*i*)的兼容行集。

根据定义，I_1 即 *R*(*, *r*, *i*−1) 存在着一个划分，使得划分在同一个子集中的行彼此兼容，且其中一个子集的代表是行 *r*，另一个子集的代表为行 *d* 且 *right*(*d*) < *left*(*i*)。显然行 *i* 与行 *d* 没有共同覆盖的列，所以行 *i* 与 *d* 兼容，根据定理 3.6，行 *i* 与 *d* 所代表的子集中的所有行均兼容。这样行 *i* 加入到 *d* 所在的子集后，行 *i* 将取代 *d* 作为该子集的代表，这样$\{i\}\cup I_1$ 就是一个以 *r*, *i* 为代表的 M(:,*i*)的兼容行集。

对于任意$(d,r)\in S_{dk}(i)$, 同样存在着 *R*(*d, r, i*−1)的一个划分，使得划分在同一个子集中的行彼此兼容，且其中一个子集的代表行是 *r*，另一个子集的代表为行 *d*。如果 $d\in C(i)$，即行 *i* 与 *d* 兼容，那么根据定理 3.6，行 *i* 与 *d* 所代表的子集中的所有行均兼容。行 *i* 加入到 *d* 所在的子集后，由于 *right*(*d*)≤*right*(*r*)≤*right*(*i*)，行 *i* 将取代 *d* 作为该子集的代表，显然$\{i\}\cup R(d,\ r,\ i-1)$是一个以 *r*, *i* 为代表的 M(:,*i*)的兼容行集，因此$\{i\}\cup I_2$ 也是。

同理，对于任意$(r,k)\in S_{dk}(i)$, *R*(*r, k, i*−1) 存在着一个划分，使得划分到同一个子集中的行彼此兼容，且其中一个子集的代表是行 *r*，另一个为 *k*。如果 $k\in C(i)$，即行 *i* 与 *k* 兼容，根据定理 5，行 *i* 与 *k* 代表的子集中的所有行均兼容。行 *i* 加入到 *k* 所在的子集后，如果 *right*(*k*)≤*right*(*i*)，行 *i* 将取代 *k* 作为该子集的

代表行，这样$\{i\}\cup R(r,\ k,\ i-1)$就是一个以 $r,\ i$ 为代表的 M(:,i)的兼容行集，因此 $\{i\}\cup I_3$ 也是。

现在证明$\{i\} \cup \max(I_1, I_2, I_2)$是一个以 r, i 为代表的 M(:, i)的最大兼容行集。

采用反证法，假设$\{i\} \cup \max(I_1, I_2, I_3)$不是 **M**(:, i)的一个以 r, i 为代表的最大兼容行集，这就是说 $R(r, i, i)$中的行数比$\{i\} \cup I_1$、$\{i\} \cup I_2$和$\{i\} \cup I_3$中的行数都要多。根据定义，$R(r, i, i)$中的行可以划分成两个子集，同一个子集的行互相兼容，且这两个子集中的行的代表分别是 $r,\ i$。令 i 所在的子集去掉行 i 后的代表行为 l，显然 l 与 i 兼容。

如果 $right(l)<left(i)$，则 $R(r, i, i)$去掉行 i 后得到的行集 $R(r,\ i,\ i)-\{i\}$是以 l, r 为代表的 **M**(:, i-1)的一个兼容行集，如果假设成立，则 $R(r,\ i,\ i)-\{i\}$比 I_1 大，这与 $R(*, r, i-1)$的定义矛盾。

如果 $right(l)\geq left(i)$，那么或者$(l, r)\in S_{d,k}(i)$或者$(r, l)\in S_{d,k}(i)$（因为 $r\in S_k(i)$）。如果$(l, r)\in S_{d,k}(i)$，则 $R(r, i, i)-\{i\}$是以 l, r 为代表的 **M**(:, i-1)的一个兼容行集。如果假设成立，则 $R(r, i, i)-\{i\}$比 I_2 大，这与 $R(l, r, i-1)$的定义矛盾；如果$(r, l) \in S_{d,k}(i)$，则 $R(r,\ i,\ i)-\{i\}$是以 r, l 为代表的 **M**(:, i-1)的一个兼容行集。如果假设成立，则 $R(r, i, i)-\{i\}$比 I_3 大，这与 $R(r, l, i-1)$的定义矛盾。

由此公式(3-9)对于所有满足 $right(r) \leq right(i)$的(r, i)都成立，用同样的方法可以证明公式(3-10)成立。

公式(3-13)的证明：

令 I_1 表示 $\max\limits_{d:\ right(d)<left(i)} R(d,i,i)$，$I_2$ 表示 $\max\limits_{d:\ d\in S_k(i),\ right(d)\leq right(i),\ right(d)<left(i+1)} R(d,i,i)$，

根据定义，$R(*, i, i) = \max(I_1, I_2)$，这样证明公式(3-13)只需证明 $I_1 = \max(\{i\} \cup R(*,\ *,\ i-1)\ ,\ \{i\} \cup \max\limits_{k:\ k\in C(i),\ right(k)\leq right(i)} R(*,k,i-1)) = \{i\} \cup \max(R(*,\ *,\ i-1)\ ,\ \max\limits_{k:\ k\in C(i),\ right(k)\leq right(i)} R(*,k,i-1))$。

对于满足条件“$right(d)<left(i)$”的任意 d，用证明公式(3-9)的方法可以证明下式成立：$R(d,i, i) = \{i\} \cup \max(R(*, d, i-1),\ \max\limits_{k:\ k\in C(i),\ right(k)\leq right(i)} R(d,k,i-1))$，因此

$I_1 = \{i\} \cup \max(\max\limits_{d:\ right(d)<left(i)} R(*,d,i-1),\ \max\limits_{k:\ k\in C(i),\ right(k)\leq right(i)} (\max\limits_{d:\ right(d)<left(i)} R(d,k,i-1)))$

$= \{i\} \cup \max(R(*, *, i-1), \max_{k:\, k\in C(i), right(k)\le right(i)} R(*,k,i-1))$。公式(3-13)成立。

公式(3-14)的证明：

令 I_1 代表 $\max_{k:\, right(k)<left(i)} R(*,k,i)$，$I_2$ 代表 $\max_{k:\, k\in S_k(i)\cup\{i\}, right(k)<left(i+1)} R(*,k,i)$，根据定义，$R(*, *, i) = \max(I_1, I_2)$。对于满足 $right(k)<left(i)$的任意 $R(d, k, i)$而言，有 $right(d)\le right(k)<left(i)$，因此有 $R(d, k, i) = R(d, k, i-1)$ (理由同公式(3-8))，所以

$I_1 = \max_{k:\, right(k)<left(i)} (\max_{d:right(d)<left(i)} R(d,k,i)) = \max_{k:right(k)<left(i)} (\max_{d:right(d)<left(i)} R(d,k,i-1))$

$= \max_{k:\, right(k)<left(i)} R(*,k,i-1) = R(*, *, i-1)$。公式(3-14)成立。

算法的时间复杂度：算法的主体部分是 Step 3。由于 **M** 经过了预处理，那么 $k<i$ 意味着 $left(k)\le left(i)$, 这时如果还有 $right(k)\ge left(i)$，则可以肯定行 k 覆盖列 $left(i)$，所以 $S_k(i)$中的元素个数$|S_k(i)|$不会超过覆盖列 $left(i)$的行数。由于 **M** 满足(k_1, k_2)参数化条件，所以$|S_k(i)|\le k_2$。这样 Step 3.1 循环不超过$(m-1)k_2$ 次，在 Step 3.1.3 中，判断两行是否兼容只需检查两行在它们共同覆盖的列上是否冲突，Step 3.1.3 执行一次所需时间为 $O(k_1)$，所以 Step 3.1 所需时间为 $O(mk_1k_2)$；可以看出 $S_{dk}(i)$中的元组个数不会超过 k_2^2，所以 Step 3.3 和 Step 3.5 循环不超过$(m-1)k_2^2$ 次，所需时间为 $O(mk_2^2)$；剩下的 Step 3.4、Step 3.6 和 Step 3.7 所需时间为 $O(mk_1)$。因此加上预处理，整个算法的时间复杂度为 $O(mk_2^2+mk_1k_2+mlogm+nk_2)$。

算法的空间复杂度：SNP 矩阵所需的空间为 $O(mk_1)$, 记录所有的 $R(d, k, i)$需要的空间 $O(mk_2^2)$ (只需记录相邻的两组 R 值，即 i-1 和 i)，所以算法的空间复杂度为 $O(mk_1+mk_2^2)$。定理得证。 □

3.3.4 PG_MSR 参数化算法

本节阐述求解问题 3.3 即 PG_MSR 的参数化算法。首先对行冲突、兼容的概念进行扩展。

给定 1 个 $m\times n$ SNP 矩阵的某一行 r 和 1 个长为 n 单体型 H，令 r_j 表示行 r 在列 j 上的值，H_j 表示 H 在对应的第 j 个 SNP 位点上的值（即第 j 个字符的值）。对于 j：$1 \le j \le n$， 如果 $r_j \ne$ ‘-’ 且 $r_j \ne H_j$，那么行 r 和单体型 H 在列 j 上冲突。如果行 r 和单体型 H 在任意列 j（$1 \le j \le n$）上均不冲突，则行 r 和单体型 H 兼容。

下面进一步介绍一些与文献[62]相同的一些概念和记号，并重新给予定义。

定义 3.4 给定一个经过预处理的 $m\times n$ 的 SNP 矩阵 **M** 和一个长度为 l 的两个单体型 $\alpha, \beta\in\{0, 1\}^l$，如果 **M** 的行存在着一个划分（$R_1$，$R_2$）使得被划分在 R_1（或 R_2）中的任意行在最后 l 列上与 α（或 β）不冲突（如果 $n < l$，则在 **M** 的第 1 列前补足 l-n 列空值），则 **M** 是 α, β 可行的。

定理 3.8 给定一个经过预处理的且每一行的洞数不超过 k 的 $m\times n$ SNP 矩阵 **M**, 删除 **M** 的最后 1 列得到一个 $m\times(n\text{-}1)$的 SNP 矩阵 **M'**，令 M*是由 **M** 的最后 k+2 列构成的 $m\times(k+2)$ SNP 矩阵（如果 $n < k+2$, 则令 **M***=**M**），单体型 $\alpha', \beta'\in\{0, 1\}^{n-1}$，单体型 $\alpha^*, \beta^*\in\{0, 1\}^{k+1}$。如果 **M'**是 α', β'可行的，**M***是 α^*, β^*可行的，且 α', β'的最后 k 字符构成的部分和 α^*, β^*最前 k 字符构成的部分相同，则 **M** 是 α, β 可行的，且 α 和 β 在每个对应 SNP 位点上的取值均不相同并不为空值，其中 α, β 是在 α', β'后分别添上的 α^*, β^*最后一个字符构成。

证明： 如果 $n<k+2$，**M***=**M**，α, β 分别是 α^*, β^*最后 n 个字符构成，因为 **M***是 α^*, β^*可行的，所以 **M** 必定是 α, β 可行的。

下面考虑 $n\geq k+2$ 情况。

因为 **M** 是经过预处理的，所以 **M**、**M'**和 **M***的每一列总有一些行的值是‘0’，还有一些行的值为‘1’，这样如果 **M'**是 α', β'可行的，**M***是 α^*, β^*可行的，则 α' 和 β'在每个对应 SNP 位点上的取值均不相同，且都不为空值，α^*和 β^*进而 α 和 β 也是如此。

令 $r[i\text{:}\ j]$表示 **M** 中的行 r 从列 i 到 j 的部分，$\alpha[i\text{:}\ j]$ 表示单体型 α 从第 i 字符到到第 j 个的部分。下面用反证法来证明定理 3.8。

假定 **M** 不是 α, β 可行的，则 **M** 中必定存在一行 r 和 α, β 都冲突。因为 **M'**是 α', β'可行的，则 $r[1\text{:}\ n\text{-}1]$应与 α'和 β'中的一个兼容，不失一般性，令 $r[1\text{:}\ n\text{-}1]$与 α'（即 $\alpha[1\text{:}\ n\text{-}1]$）兼容。既然 r 和 α 冲突，则 r 的最后一列的值必定不为空，且与 α 最后一个字符（即 α^*最后一个字符）不同，这样就必定与 β 最后一个字符（即 β^*最后一个字符）相同。既然 **M***是 α^*, β^*可行的，则 $r[n\text{-}k\text{-}1\text{:}\ n]$必定与 β^* 兼容，这样 $r[n\text{-}k\text{-}1\text{:}\ n\text{-}1]$与 $\alpha[n\text{-}k\text{-}1\text{:}\ n\text{-}1]$ 和 $\beta[n\text{-}k\text{-}1\text{:}\ n\text{-}1]$都兼容。由于 $\alpha[n\text{-}k\text{-}1\text{:}\ n\text{-}1]$和 $\beta[n\text{-}k\text{-}1\text{:}\ n\text{-}1]$在每个对应的位点上的值均不相等，且不为空，这样行 r 在从列 n-k-1 到 n-1 这 k+1 列均为空值，如果存在某一列 $j < n\text{-}k\text{-}1$ 使得行 r 在该列取非空值，则行 r 的洞数超过了 k，这与定理中的前提条件不符；如果不存在这样的列 j，则 r 应和 β 兼容，这与假定相矛盾。定理得证。 □

为了叙述简洁，α, β 可以看成二进制数（单体型 “1001” 看成二进制数 1001）来比较大小。令 $\alpha \lhd \beta$ 表示 α 比 β 小且在每个对应位置均不相同。$\alpha \lhd \beta$ 隐含着 α 的第一个字符必须是 0，β 的第一个字符必须是 1。令 $\min(\alpha, \beta)$ 和 $\max(\alpha, \beta)$ 分别表示取较小和较大的单体型。

在本节下面的内容中，**M** 是满足 (k_1, k_2, k) 参数化条件的 SNP 矩阵，$h=k+1$，$\alpha, \beta \in \{0, 1\}^h$, $\alpha \lhd \beta$，满足这样约束的不同的单体型对 (α, β) 的数目为 2^k。

考虑 **M** 的前 j 列构成的矩阵 $\mathbf{M}(:, j)$。$l = \max(j, h)$，$\alpha', \beta' \in \{0, 1\}^l$，且 α', β' 在每个对应的位置均不相同。令删除 $\mathbf{M}(:, j)$ 中除列 j 外的一些列后得到的 SNP 矩阵如果是 α', β' 可行的，则该 α', β' 可行矩阵记作 $\mathrm{F}(j; \alpha', \beta')$。

定义 3.5 对于给定的 α, β，考虑满足条件 α', β' 最后 h 个位点分别和 α, β 对应的 h 个位点相同的所有 $\mathrm{F}(j; \alpha', \beta')$ 中，具有列数最多的 SNP 矩阵记作 $\mathrm{M}(j; \alpha, \beta)$，$\mathrm{M}(j; \alpha, \beta)$ 的列数定义为 $K[j; \alpha, \beta]$。

显然，下列公式成立：

$$K[1;\ \alpha,\ \beta] = 1 \tag{3-15}$$

令 $[\mathrm{M}(i; \alpha, \beta) \mid c_j]$ 表示在 $\mathrm{M}(i; \alpha, \beta)$ 的最后一列后附加 **M** 的列 j 后得到的 SNP 矩阵。

定义 3.6 $OK(j; \alpha, \beta)$ 定义为满足下列条件的三元组 (i, α', β') 的集合：

(1) $j - k_1 < i < j$;

(2) $\alpha', \beta' \in \{0, 1\}^{k+1}$ 且 $\alpha' \lhd \beta'$;

(3) α' 最后 k 个字符和 α（或 β）的最前 k 个字符相同，β' 最后 k 个字符和 α、β 中余下的那个的最前 k 个字符相同；

(4) $[\mathrm{M}(i; \alpha', \beta') \mid c_j]$ 是 α, β 可行的。

显然 $OK(j; \alpha, \beta)$ 中的三元组的个数即 $|OK(j; \alpha, \beta)|$ 不会超过 $2k_1$。因为 **M** 满足 (k_1, k_2, k) 参数化条件，检查 $[\mathrm{M}(i; \alpha', \beta') \mid c_j]$ 是否 α, β 可行需要时 $O(k_1 k_2)$。

由于 **M** 满足 (k_1, k_2, k) 参数化条件，**M** 中没有行既覆盖列 j，又能覆盖列号小于或等于 $j-k_1$ 的列。令 $\alpha^* = \alpha[1: k]$, $\beta^* = \beta[1: k]$, 显然对于任意 $l \le j-k_1$，$[\mathrm{M}(l; 0\alpha^*, 1\beta^*) \mid c_j]$ 和 $[\mathrm{M}(l; 0\beta^*, 1\alpha^*) \mid c_j]$ 都是 α, β 可行的，其中 0α 表示在 a 前加上 ‘0’ 得到的单体型，其它 1β 等类似。令 $A[j-k_1; \alpha^*, \beta^*]$ 表示所有 $l \le j-k_1$ 的 $K[l; 0\alpha^*, 1\beta^*]$、$K[l; 0\beta^*, 1\alpha^*]$ 这 $2(j-k_1)$ 个值中的最大值；如果 $j-k_1<1$，则 $A[j-k_1; \alpha^*, \beta^*] = 0$。

根据定义 3.5，$K[j; \alpha, \beta] \geq A[j-k_1; \alpha^*, \beta^*] + 1$。再根据定义 3.6 和定理 3.8 可以证明下面公式成立：

$$K[j;\alpha,\beta] = \max(A(j-k_1;\alpha^*,\beta^*), \max_{(i;\ \alpha',\beta')\in OK(j;\alpha,\beta)} K[i;\ \alpha',\beta']) + 1 \tag{3-16}$$

$A[i; \alpha^*, \beta^*]$可以通过下面的方式计算:

$$A[i;\alpha^*,\beta^*] = \begin{cases} 0, & \text{if } i < 1; \\ \max(A[i-1;\alpha^*,\beta^*],\ K[i;0\alpha^*,1\beta^*],\ K[i;0\beta^*,1\alpha^*]), & \text{otherwise.} \end{cases} \tag{3-17}$$

通过公式(3-17)计算出所有的 $A[i; \alpha^*, \beta^*]$需时间 $O(2^k n)$。

对于一个 $m \times n$ SNP 矩阵 **M**，有下列公式成立：

$$\text{PG\_MSR}(\mathbf{M}) = \max_{\alpha,\beta\in\{0,\ 1\}^k, \alpha\lhd\beta} A[n;\alpha,\beta] \tag{3-18}$$

这样利用公式(3-15)~(3-18)，我们得到了求解 PG_MSR 的参数化算法，在下面的算法中，(α, β)用一个长为 k 的 2 进制数 h 编码：在 h 最高位加上 0 就得到 α，这种操作记作 $0h$；对 α 按位取反（0 变 1 或 1 变 0）就得到 β，这种操作记作$\sim\alpha$；同样，(α', β')用一个长为 k 的 2 进制数 h'编码; (α^*, β^*)用一个长为 k-1 的 2 进制数 h^*编码。$\alpha/2$ 表示取 h 的前 k 位构成的 k 位 2 进制数。

PG_MSR 算法：

Input：预处理后得到的 $m \times n$ SNP 矩阵 **M**

Output：**M** 的 PG_MSR 解

Step 1. 扫描 **M** 得到参数 k_1、k_2、k 和覆盖列的行数的向量 H;

//$H[j]$表示覆盖列 j 的行数

Step 2. **for** $h = 0..2^k$-1 **do**

Step 2.1. $M(1; h)= \{1\}$; $c = h$ 的二进制最低位; //$M(1; h)$表示保留的列集合

$K[1; h] = 1$; //公式(3-15), h 编码 α, β

Step 2.2. **for** l=1..$H[1]$ **do**

取 *rowset*(1)第 l 行的行号给 r

if $\mathbf{M}_{r,1} = c$ **then** $C_{1,l}$=0; // $C_{1,l}$=0 表示行 r 与 α 兼容

else $C_{1,l}$=1; // $C_{1,l}$=1 表示行 r 与 β 兼容

Step 3. **for** $h^* = 0..2^{k-1}$-1 **do** //公式(3-17), $A[1; h^*]$表示 $A[1; \alpha^*, \beta^*]$

$M_A(1; h^*)=\{1\}$;　$A[1; h^*] = 1$; // M_A 表示保留的列集合

Step 4. **for** $j = 2 .. n$ **do**　　//根据公式(3-16)递推

Step 4.1. $j'=j \bmod k$;　　// mod：取模运算

Step 4.2. **for** $h^* = 0..2^{k-1}-1$ **do**　　//初始化，$A[j'; h^*]$表示 $A[j; \alpha^*, \beta^*]$

$M_A(j'; h^*)=\{j\}$;　$A[j'; h^*] = 1$; // M_A 表示保留的列集合

Step 4.3. **for** $h = 0..2^{k}-1$ **do**

Step 4.3.1. $K[j'; h] = 1$; $M(j'; h) =\{j\}$;　　//$K[j'; h]$ 表示 $K[j; \alpha, \beta]$

Step 4.3.2. $maxK=maxA=0$;　$c = h$ 的二进制最低位;

Step 4.3.3. **for** $i = \max(1, j-k_1+1) .. j-1$ **do**

Step 4.3.3.1.　$i'= i \bmod k_1$;

Step 4.3.3.2.　$h'=(0h)/2$;　$conflict=0$;

Step 4.3.3.3. **for** $l=1..H[i]$ **do**

{　取 $rowset(i)$第 l 行的行号给 r;

if $\mathbf{M}_{r,j} \neq -$ **then**

{　**if** $C_{i',l}=0$, $\mathbf{M}_{r,j} \neq c$ **then** $conflict=1$;

if $C_{i',l}=1$, $\mathbf{M}_{r,j} = c$ **then** $conflict=1$;　}　}

Step 4.3.3.4. **if** $conflict=0$ **then**　//定义 3.6 条件(4)成立, $(i; \alpha', \beta') \in OK(j; \alpha, \beta)$

if $maxK<K[i'; h']$ **then**

$maxK = K[i'; h']$; $maxi= i$; $maxh= h'$;

Step 4.3.3.5. $h'=(\sim(0h))/2$;　$conflict=0$;

Step 4.3.3.6. **for** $l=1..H[i]$ **do**

取 $rowset(i)$第 l 行的行号给 r;

if $\mathbf{M}_{r,j} \neq -$ **then**

{　**if** $C_{i',l}=0$, $\mathbf{M}_{r,j} = c$ **then** $conflict=1$;

if $C_{i',l}=1$, $\mathbf{M}_{r,j} \neq c$ **then** $conflict=1$;　}

Step 4.3.3.7. **if** $conflict=0$ **then**　//定义 3.6 条件(4)成立, $(i; \alpha', \beta') \in OK(j; \alpha, \beta)$

if $maxK<K[i'; h']$ **then**

$maxK = K[i'; h']$; $maxi= i$; $maxh= h'$;

Step 4.3.4. **if** $j-k_1>0$ **then**

$h^*= (0h)/2$;　$l=(j-k_1) \bmod k_1$;

if $A[l; h^*] > maxK$ **then**　　//公式(3-16)

maxA=1; *K*[*j*'; *h*]= *A*[*l*; *h**]+1; *M*(*j*'; *h*) =M_A(1; *h**)∪{*j*};

Step 4.3.5. **if** *maxA* =1 or *maxK* = 0 **then**

for *l*=1..*H*[*j*] **do**

{ 取 *rowset*(*j*)第 *l* 行的行号给 *r* ; $C_{j',l}$= − ;

if $\mathbf{M}_{r,j}$= *c* **then** $C_{j',l}$= 0; // $C_{1,l}$= 0 表示行 *r* 与 *α* 兼容

else if $\mathbf{M}_{r,j}$ ≠ − **then** $C_{j',l}$= 1; }// $C_{1,l}$= 1 表示行 *r* 与 *β* 兼容

Step 4.3.6. **if** *maxA* = 0 and *maxK* > 0 **then**

i'= *maxi* mod k_1; *l*'=1; *M*(*j*'; *h*) = *M* (*i*'; *maxh*)∪{*j*};

K[*j*'; *h*] = *K*[*i*'; *maxh*]+1; //公式(3-16)

for *l*=1..*H*[*j*] **do**

{ 取 *rowset*(*j*)第 *l* 行的行号给 *r* ; $C_{j',l}$= − ;

if $\mathbf{M}_{r,j}$= *c* **then** $C_{j',l}$= 0; // $C_{1,l}$= 0 表示行 *r* 与 *α* 兼容

else if $\mathbf{M}_{r,j}$ ≠ − **then** $C_{j',l}$= 1; // $C_{1,l}$= 1 表示行 *r* 与 *β* 兼容

else

{ 取 *rowset*(*maxi*)第 *l*'行的行号给 *r*' ;

while *r*' < *r* and *l*' < *H*[*maxi*] **do**

l'= *l*'+1; 取 *rowset*(*maxi*)第 *l*'行的行号给 *r*' ;

if *r*' = *r* **then** $C_{j',l}$= $C_{i',l'}$; } }

Step 4.3.7. *h** = min(*h*, ~*h*) ;

Step 4.3.8. **if** *A*[*j*'; *h**] < *K*[*j*'; *h*] **then** //公式(3-17)

M_A(*j*'; *h**) = *M*(*j*'; *h*); *A*[*j*'; *h**] = *K*[*j*'; *h*];

Step 5. *maxh* = 0;

Step 6. **for** *h** = 1..2^{k-1}-1 **do**

if *A*[*j*'; *maxh*] < *A*[*j*'; *h**] **then** *maxh* = *h**; //公式(3-18)

Step 7. **return** *A*[*j*'; *maxh*]; // 对应保留的列在 M_A(*j*'; *maxh*)中

PG_MSR 算法的时间复杂度分析：Step 1 中扫描 **M** 所需的时间为 $O(mk_1)$；Step 2 所需的时间为 $O(k_2 2^k)$；Step 3 所需的时间为 $O(2^k)$；Step 4.3.3.3 和 Step 4.3.3.6 所需的时间都为 $O(k_2)$，所以 Step 4.3.3 所需的时间都为 $O(k_1k_2)$，Step 4.3 所需的时间都为 $O(k_1k_2 2^k)$，进而 Step 4 的时间复杂度为 $O(nk_1k_2 2^k)$。这样加上预处理的时间整个算法的时间复杂度为 $O(nk_1k_2 2^k+mlogm+nk_2+mk_1)$。

空间复杂度分析：SNP 矩阵 **M** 所需的空间为 $O(mk_1)$, *A* 和 *K* 需要空间

$O(k_1 2^k)$，M_A 和 M 需要空间 $O(nk_1 2^k)$，C 需要空间 $O(k_2 k_1 2^k)$，所以整个算法的空间复杂度为 $O(nk_1 2^k + k_2 k_1 2^k + mk_1)$。

3.3.5 PG_MFR 参数化算法

本节阐述求解问题 3.4 即 PG_MFR 的参数化算法。

假定 **M** 是一个满足(k_1, k_2, k)参数化条件的经过预处理的 $m \times n$ SNP 矩阵，d 是 **M** 中的一行，$\alpha \in \{0, 1\}^k$。令 $d[\alpha]$表示用 α 中的字符依次填充 f 中的洞所得到的无空隙的行（行 d 最多有 k 个洞，所以 α 中的字符总是够用的）。给定 $\alpha, \beta \in \{0, 1\}^k$，**M** 中的两行 d 和 l，令 $\mathbf{M}_{d[\alpha], l[\beta]}$表示行 d 和 l 分别用 $d[\alpha]$和 $l[\beta]$替换后的 SNP 矩阵 **M**。

显然当 $m \leq 2$ 时，**M** 肯定是可行的，无须删除任何行，这样很容易求出 **M** 的 MFR 问题的解。下面假定 $m > 2$。

在本节下文中，如果没有特别声明，则 $\alpha, \beta \in \{0, 1\}^k$。

给定行 i（$1 \leq i \leq m$）、α、β 和满足 $right(d) \leq right(l)$且不大于 i 的两行 (d, l)，考虑 $m \times n$ 的 SNP 矩阵 **M** 的前 i 行构成的矩阵 $\mathbf{M}(i, :)$， $\mathbf{M}(i, :)$中行 d 和 l 行分别用 $d[\alpha]$和 $l[\beta]$替换后得到的 SNP 矩阵记作 $\mathbf{M}_{d[\alpha], l[\beta]}(i, :)$。

令 R 为 $\mathbf{M}_{d[\alpha], l[\beta]}(i, :)$的行的子集，如果保留 R 中的所有行能使 $\mathbf{M}_{d[\alpha], l[\beta]}(i, :)$可行，那么 R 就称 $\mathbf{M}_{d[\alpha], l[\beta]}(i, :)$的一个兼容行集。如果 R 是 $\mathbf{M}_{d[\alpha], l[\beta]}(i, :)$的一个兼容行集，则必定可以把 R 划分成两个子集 R_1 和 R_2，使得划分在同一子集的行彼此兼容，这样的划分叫做 R 的兼容划分。

定义 3.7 $R[d, \alpha; l, \beta; i]$ 定义为 $\mathbf{M}_{d[\alpha], l[\beta]}(i, :)$的满足下列条件的最大兼容子集 R：

(1)$d, l \in R$且对任意$r \in R$，如果$r \neq l$，则（$right(r) > right(l)$）或（$right(r) = right(l)$且$r < l$）；

(2)存在R的一个兼容划分(R_1, R_2), 使得$d \in R_1, l \in R_2$；

(3)对任意$r \in R_1$，如果$r \neq d$，则（$right(d) > right(r)$）或（$right(d) = right(r)$且$r < d$）。

在上述定义中，l称为R_1的代表行，d称为R_2的代表行。

$R[d, \alpha; l, \beta; i]$ 称为以(d, l)为代表行$\mathbf{M}_{d[\alpha], l[\beta]}(i, :)$的最大兼容子集。

定义3.8 $K[d, \alpha; l, \beta; i]$定义为$R[d, \alpha; l, \beta; i]$中的行数，即$K[d, \alpha; l, \beta; i] = |R[d, \alpha; l, \beta; i]|$。

令$K[*; l, \beta; i]$为d, α满足: $right(d) < left(i+1), \alpha \in \{0, 1\}^k$的所有有定义的$K[d, \alpha;$

$l, \beta; i]$ 的最大值，即:

$$K[*;l,\beta;i] = \max_{right(d)<right(i+1),\alpha\in\{0,1\}^k} K[d,\alpha;l,\beta;i]$$

令$K[*, *; i]$为d, α, l, β满足: $right(d)<left(i+1) \wedge right(l)<left(i+1), \alpha,\beta\in\{0, 1\}^k$ 的所有有定义的$K[d, \alpha; l, \beta; i]$ 的最大值，即:

$$K[*;*;i] = \max_{right(d)<right(i+1),right(l)<left(i+1),\alpha,\beta\in\{0,1\}^k} K[d,\alpha;l,\beta;i]。$$

如果在 **M** 的最后一行后添加虚拟行 m+1，令 $left(m+1) = n+1$, 那么可通过下列公式计算 **M** 的 PG_MFR 的解：

$$\text{PG\_MFR}(\mathbf{M}) = K[*; *; m] \tag{3-19}$$

当$i = 1$时，为了使上面的定义和记号有意义，在**M**的第一行前还添加了虚拟行0和-1，并令它们与其他所有行均兼容，$left(-1) = -1$，$right(-1) = -2$，$left(0) = 0$，$right(0) = -1$。但是当对行集合的元素计数时，这两行不考虑在内。

这样按照定义3.7，当$i = 1$时，(d, l)的值只能取(-1,0)、(-1,1) 和(0, 1)，对应的K值如下：

$$K[-1, \alpha; 0, \beta; 1] = 0，\ K[-1, \alpha; 1, \beta; 1] = K[0, \alpha; 1, \beta; 1] = 1。$$

在此基础上，进一步可以得到：

$$K[*; 1, \beta; 1] = 1 \tag{3-20}$$

$$K[*; *; 1] = \begin{cases} 1, & \text{if } right(1) < left(2); \\ 0, & \text{otherwise.} \end{cases} \tag{3-21}$$

为了叙述简洁，如果 d 和 l 小于 i，且满足下述条件之一，则称为$(d, l)\in \boldsymbol{T}$：

(1) $left(i) \le right(d) < right(l)$;

(2) $left(i) \le right(d) = right(l)$ 且 $d < l$。

如果 d 和 l 小于或等于 i，且满足上述条件之一，则称为$(d, l)\in \boldsymbol{T}_i$：

对于预处理，且满足(k_1, k_2, k) 参数化条件的 $m\times n$ SNP 矩阵 **M**，对于任意行 i（$1 \le i \le m$），由于覆盖列 $left(i)$的行数不会超过 k_2，而满足上述条件的行 d 和 l 均应覆盖列 $left(i)$，所以满足上述条件的 d 和 l 的数目不会超过 k_2，$\boldsymbol{T}_i$ 和 $\boldsymbol{T}$ 中的元素个数不会超过 k_2^2。

令i是**M**中的一行（$2 \le i \le m$）， 有下面公式成立：

(1) 对于$(d, l) \in \boldsymbol{T}$，如果行$i$和$d[\alpha]$兼容且 $right(i) < right(d)$，或者i和$l[\beta]$兼容且$right(i) < right(l)$, 则:

$$K[d,\alpha;l,\beta;i]=K[d,\alpha;l,\beta;i-1]+1; \tag{3-22}$$

否则： $K[d,\alpha;l,\beta;i]=K[d,\alpha;l,\beta;i-1]$。 (3-23)

(2) 对于 l: $1\le l<i$ 且 $left(i)\le right(l)$，如果 $right(i)<right(l)$且行 i 和 $l[\beta]$兼容，则：

$$K[*;l,\beta;i]=\max(K[*;l,\beta;i-1]+1,I_0); \tag{3-24}$$

如果$right(i)\ge right(l)$或i和$l[\beta]$冲突，则：

$$K[*;l,\beta;i]=\max(K[*;l,\beta;i-1],I_0) \tag{3-25}$$

其中： $I_0=\max\limits_{(d,l)\in \boldsymbol{T}_i,\ left(i)\le right(d)<left(i+1),\ \alpha\in\{0,1\}^k} K[d,\alpha;l,\beta;i])$。

(3) 对于 d: $1\le d<i$ 且 $left(i)\le right(d)$，如果 $right(i)\ge right(d)$，则：

$$K[d,\alpha;i,\beta;i]=1+\max(\ K(*;d,\alpha;i-1),\ I_1,\ I_2\ ) \tag{3-26}$$

上述公式中，$I_1=\max\limits_{(d,j)\in \boldsymbol{T},(j,\beta_j)\in R_{OK}(i,\beta),right(j)\le right(i)} K[d,\alpha;j,\beta_j;i-1]$；

$$I_2=\max_{(j,d)\in \boldsymbol{T},(j,\beta_j)\in R_{OK}(i,\beta)} K[j,\beta_j;d,\alpha;i-1]。$$

上式及下文中，$R_{OK}(i,\alpha)$表示满足下面条件的二元组(j,β)的集合：$j<i$，$right(j)\ge left(i)$且$j[\beta]$和$i[\alpha]$兼容。

(4) 对于l: $1\le l<i$且$left(i)\le right(l)$，如果$right(i)<right(l)$，则：

$$K[i,\alpha;l,\beta;i]=1+\max(\ K(*;l,\beta;i-1),\ I_3\ ) \tag{3-27}$$

上述公式中，$I_3=\max\limits_{(j,\alpha_j)\in R_{OK}(i,\alpha),(j,l)\in \boldsymbol{T},right(j)\le right(i)} K[j,\alpha_j;l,\beta;i-1]$。

(7) $K[*,i,\beta;i]=\max(\ K[*,*,i-1]+1,\ I_4,\ I_5\ )$ (3-28)

上述公式中，$I_4=\max\limits_{d<i,\ left(i)\le right(d)<left(i+1)} K[d,\alpha;i,\beta;i])$；

$$I_5=\max_{l<i,\ right(i)\ge right(l)\ge left(i),\ (l,\beta_l)\in R_{OK}(i,\beta)} K[*;l,\beta_l;i-1]+1。$$

(8) $K[*,*;i]=\max(\ K[*,*,i-1],I_6\ )$ (3-29)

其中，$I_6=\max_{l\le i,\ left(i)\le right(l)<left(i+1),\ \beta\in\{0,1\}^k} K[*;l,\beta;i]$。

根据公式(3-19) ~ (3-29), 我们得到了如下求解有空隙的 MFR 问题的参数化算法 PG_MFR。

PG_MFR 算法：

Input: 预处理后得到的$m\times n$ SNP矩阵**M**

Output: **M**的PG_MFR解

Step 1. 扫描 **M** 得到参数 k_1、k_2、k;

for $\alpha\in\{0,1\}^k$ **do** $K[*;1,\alpha;1]=1$; //公式(3-20)

if $right(1)<left(2)$ **then** $K[*;*;1]=1$; **else** $K[*;*;1]=0$; //公式(3-21)

$i=2$; //从第 2 行开始向后递推

Step 2. **while** $i\le m$ **do**

Step 2.1. $K[*,*;i]=K[*,*;i\text{-}1]$; //用公式(3-29)第一项初始化

for $\beta\in\{0,1\}^k$ **do** $K[*;i,\beta;i]=0$;

for $\beta\in\{0,1\}^k$ and l: $l\le i$, $left(i)\le right(l)$ **do**

{ $K[*;l,\beta;i]=0$;

// C 记录记录兼容信息，$C[l,\beta]=0$ 表示行 i 和 $l[\beta]$兼容

if 行 i 和 $l[\beta]$不兼容 **then** $C[l,\beta]=1$; **else** $C[l,\beta]=0$;

for $\alpha\in\{0,1\}^k$ **do** // $C_i[l,\beta;\alpha]=0$ 表示行 $i[\alpha]$和 $l[\beta]$兼容

if $i[\alpha]$和 $l[\beta]$ 不兼容 **then** $C_i[l,\beta;\alpha]=1$; **else** $C_i[l,\beta;\alpha]=0$; }

Step 2.2. **for** $\alpha,\beta\in\{0,1\}^k$, l: $right(l)>right(i)$, $l<i$ **do** $K[i,\alpha;l,\beta;i]=0$;

for $\alpha,\beta\in\{0,1\}^k$, d: $left(i)\le right(d)\le right(i)$, $l<i$ **do** $K[d,\alpha;i,\beta;i]=0$;

Step 2.3. **for** $\alpha,\beta\in\{0,1\}^k$ and d, l: $(d,l)\in \boldsymbol{T}$ **do**

Step 2.3.1. **if** ($C[d,\alpha]=0$ and $right(i)<right(d)$) or

($C[l,\beta]=0$ and $right(i)<right(l)$) **then**

$K[d,\alpha;l,\beta;i]=K[d,\alpha;l,\beta;i\text{-}1]+1$; // 公式(3-22)

else $K[d,\alpha;l,\beta;i]=K[d,\alpha;l,\beta;i\text{-}1]$; // 公式(3-23)

Step 2.3.2. **if** $C[d,\alpha]=0$ and $right(l)>right(i)\ge right(d)$ **then**

for $\alpha_i\in\{0,1\}^k$ **do**

if $C_i[d,\alpha;\alpha_i]=0$ **then** // $i[\alpha_i]$和$d[\alpha]$兼容, 公式(3-27)中的I_3

$K[i,\alpha_i;l,\beta;i]=\max(K[d,\alpha;l,\beta;i\text{-}1]+1, K[i,\alpha_i;l,\beta;i])$;

Step 2.3.3. **if** $C[d, \alpha] = 0$ and $right(i) \geq right(l)$ **then**

for $\alpha_i \in \{0, 1\}^k$ **do**

if $C_i[d, \alpha; \alpha_i] = 0$ **then** // 公式(3-26)中的 I_2

$K[l, \beta; i, \alpha_i; i] = \max(K[d, \alpha; l, \beta; i\text{-}1] + 1, K[l, \beta; i, \alpha_i; i])$;

Step 2.3.4. **if** $C[l, \beta] = 0$ and $right(i) > right(l)$ **then**

for $\beta_i \in \{0, 1\}^k$ **do**

if $C_i[l, \beta; \beta_i] = 0$ **then** // $i[\beta_i]$和 $l[\beta]$兼容，公式(3-26)中 I_1

$K[d, \alpha; i, \beta_i; i] = \max(K[d, \alpha; l, \beta; i\text{-}1] + 1, \boldsymbol{K}[d, \alpha; i, \beta_i; i])$;

Step 2.4. **for** $\beta \in \{0, 1\}^k$ and l: $1 \leq l \leq i$, $left(i) \leq right(l)$ **do**

Step 2.4.1. **if** $C[l, \beta] = 0$ and $right(i) < right(l)$ **then**

$K[*; l, \beta; i] = K[*; l, \beta; i\text{-}1] + 1$; // 公式(3-24)第一项

else $K[*; l, \beta; i] = K[*; l, \beta; i\text{-}1]$; // 公式(3-25)第一项

Step 2.4.2. **if** $C[l, \beta] = 0$ and $right(i) > right(l)$ **then**

for $\beta_i \in \{0, 1\}^k$ **do**

if $C_i[l, \beta; \beta_i]$ **then** // 公式(3-28)中的 I_5

$K[*; i, \beta_i; i] = \max(K[*; l, \beta; i\text{-}1] + 1, K[*; i, \beta_i; i])$;

Step 2.4.3. **for** $\alpha \in \{0, 1\}^k$ **do**

if $right(i) < right(l)$ **then** // 公式(3-27)第一项

$K[i, \alpha; l, \beta; i] = \max(K[*; l, \beta; i\text{-}1] + 1，K[i, \alpha; l, \beta; i])$;

else // 公式(3-26)第一项

$K[l, \beta; i, \alpha; i] = \max(K[*; l, \beta; i\text{-}1] + 1，K[l, \beta; i, \alpha; i])$;

Step 2.5. **for** $\alpha \in \{0, 1\}^k$ **do** // 公式(3-28)第一项

$K[*; i, \alpha; i] = \max(K[*, *; i\text{-}1] + 1，K[*; i, \alpha; i])$;

Step 2.6. **for** $\alpha,\beta \in \{0, 1\}^k$, $(d, l) \in \boldsymbol{T}_i$ **do**

if $right(d) < left(i+1)$ **then**

// 公式(3-24)和(3-25)中的 I_0，公式(3-28) 中的 I_4

$K[*; l, \beta; i] = \max(K[*; l, \beta; i], K[d, \alpha; l, \beta; i])$;

Step 2.7. **for** $\beta \in \{0, 1\}^k$ and l: $1 \leq l \leq i$, $right(l) \geq left(i)$ **do**

if $right(l) < left(i+1)$ **then** // 公式(3-29)中的 I_6

$K[*; *; i] = \max(K[*, *; i], K[*; l, \beta; i])$

Step 2.8. $i = i+1$;

Step 3. return $K[*; *; i\text{-}1]$; // 公式(3-19)

定理 3.8 对于一个预处理后满足(k_1, k_2, k)参数化条件的 $m\times n$ SNP 矩阵 **M**，PG_MFR 算法能正确地为 **M** 求解 MFR 问题，加上预处理，其时间复杂度为 $O(mk_1k_2 2^{2k}+mk_2^2 2^{3k}+mlogm+nk_2+mk_1)$，空间复杂度为 $O(mk_1+ mk_2^2 2^{2k})$。

证明：PG_MFR 算法是基于公式(3-19)~(3-29)。根据公式(3-29)，为了计算 $K[*,*; i]$，需要知道 $K[*,*; i\text{-}1]$，及满足下列条件的 $K[*; l, \beta; i]$的值：$left(i) \le right(l) < left(i+1)$。知道了下面的值就可以根据公式(3-24)、(3-25)和(3-28)计算上述的 $K[*; l, \beta; i]$值：$K[*,*; i\text{-}1]$，$K[*; l', \beta'; i\text{-}1]$（其中 $l'<i, left(i) \le right(l')$），$K[d, \alpha; l, \beta; i]$ （其中$(d, l) \in \boldsymbol{T}_i$）。而上述 $K[d, \alpha; l, \beta; i]$ 可利用公式(3-22)、(3-23)、(3-26)和(3-27)利用下面的值进行计算得到：$K[d', \alpha; l', \beta; i\text{-}1]$（其中$(d', l') \in \boldsymbol{T}$），$K[*; l', \beta; i\text{-}1]$ （其中 $l'<i, left(i) \le right(l')$）。这样以此类推，可以得知最终所需的初始值就是公式(3-20)和(3-21)提供的两个值。

下面证明上述公式的正确性。

首先证明公式(3-19)的正确性：$K[*; *; m]$ 取所有有定义的 $K[d, \alpha; l, \beta; m]$的最大值，根据 K 的定义，容易验证 $K[*; *; m]$就是使 **M** 可行能保留的最大的行数。

公式(3-20)和(3-21)是显然的。

公式(3-22)和(3-23)的证明：从$K[d, \alpha; l, \beta; i\text{-}1]$求$K[d, \alpha; l, \beta; i]$，只须考虑从$M_{d[\alpha],l[\beta]}(i\text{-}1, :)$到$M_{d[\alpha],l[\beta]}(i, :)$所增加的行$i$。考虑$R[d, \alpha; l, \beta; i\text{-}1]$中的所有行可以划分成以 l所代表的R_1和以 d所代表的R_2，如果行i和$d[\alpha]$兼容且 $right(i) < right(d)$，或者i和$l[\beta]$兼容且$right(i) < right(l)$, 则行i可以加入R_2或R_1中，而使得代表行仍然是d和l，这样保留$R[d, \alpha; l, \beta; i\text{-}1]$中的行及行$i$能使$M_{d[\alpha],l[\beta]}(i, :)$可行, 因此$R[d, \alpha; l, \beta; i\text{-}1]$是以$d, l$为代表的$M_{d[\alpha],l[\beta]}(i\text{-}1, :)$的最大兼容子集当且仅当$R[d, \alpha; l, \beta; i\text{-}1] \cup \{i\}$是以$d, l$为代表的$M_{d[\alpha],l[\beta]}(i\text{-}1, :)$的最大兼容子集，公式(3-22)成立。

如果条件“行i和$d[\alpha]$兼容且 $right(i) < right(d)$，或者i和$l[\beta]$兼容且$right(i) < right(l)$”不能满足，显然对于以(d, l)为代表的$M_{d[\alpha],l[\beta]}(i\text{-}1, :)$的任意兼容子集$R$来说，若行$i$加入$d$（或$l$）为代表的$R_1$（或$R_2$）中，则使$R_1$（或$R_2$）中的行不再彼此兼容，或者$R_1$（或$R_2$）的代表行应是$i$而不再是$d$（或$l$），因此以$d, l$为代表的$M_{d[\alpha],l[\beta]}(i, :)$的兼容子集不应该包含行$i$。公式(3-23)成立。

其他公式可以按照类似的方法进行证明，由于证明过程更加冗长，在此不再赘述。

算法的时间复杂度分析：加上扫描 **M** 所需的时间 $O(mk_1)$，Step 1 所需时间 $O(mk_1+2^k)$。Step 2.1 所需的时间为 $O(k_1k_2 2^{2k})$；Step 2.2 所需的时间为 $O(k_2 2^k)$；

Step 2.3.2、2.3.3 和 2.3.4 所需的时间为 $O(2^k)$，这样 Step 2.3 所需的时间为 $O(k_2^2 2^{3k})$；Step 2.4.2 所需的时间为 $O(2^k)$，这样 Step 2.4 所需的时间为 $O(k_2 2^{2k})$；Step 2.5 所需的时间为 $O(2^k)$；Step 2.6 所需的时间为 $O(k_2^2 2^{2k})$；Step 2.7 所需的时间为 $O(k_2 2^k)$；由此可知 Step 2 中所需时间 $O(mk_1k_2 2^{2k} + mk_2^2 2^{3k})$。这样加上预处理的时间整个算法的时间复杂度为 $O(mk_1k_2 2^{2k} + mk_2^2 2^{3k} + mlogm + nk_2 + mk_1)$。

空间复杂度分析：算法向后递推到行 i 时，只需要行 i-1 的相关值，因此算法所需的空间如下：SNP 矩阵 **M** 所需的空间为 $O(mk_1)$，$K[i, \alpha; l, \beta; i]$需要空间最大为 $O(k_2 2^{2k})$，如果还要记录对应保留的行号，则需要空间 $O(mk_2 2^{2k})$，这样整个算法的空间复杂度为 $O(mk_1 + mk_2^2 2^{2k})$。 □

3.4 实验结果

我们用 C++语言实现了 P_MFR、P_MSR、PG_MFR 和 PG_MSR 算法，Bafna 等[62]的求解无空隙的 MFR 和 MSR 算法的 C 语言源程序来自文献[134]作者提供的 Fast hare 源程序，这两个算法在本节中记作 MFR 和 MSR。Bafna 等[62]的求解有空隙的 MFR 和 MSR 算法的 C 语言实现是通过对 Fast hare 源程序中的对应程序修改而成，这两个算法在本节中分别记作 G_MFR 和 G_MSR。我们在一台 Linux 服务器（4 个 Intel Xeon 3.6G CPU，4G RAM）上对这些算法的运行时间(Running Time)和重建率(Reconstruction Rate，RR)进行了比较。

令 $\boldsymbol{h} = (h_1, h_2)$是个体一对真实的单体型，$\hat{\boldsymbol{h}} = (\hat{h}_1, \hat{h}_2)$ 是通过算法在个体的片段数据上重建出来的一对单体型，重建率 RR 定义[68] [69]如下：

$$\mathrm{RR}(\boldsymbol{h}, \hat{\boldsymbol{h}}) = 1 - \frac{\min(r_{1,1} + r_{2,2},\ r_{1,2} + r_{2,1})}{2n},$$

其中当i, j 等于1或2时，$r_{i,j} = \sum_{l=1,\ldots,n} d(h_i[l], \hat{h}_j[l])$，而$h_i[l]$、$\hat{h}_j[l]$分别表示这两个单体型第$l$个SNP位点上的值。对两个SNP值$x$和$y$：

$$d(x,y) = \begin{cases} 1, & \text{if } x \neq \text{-}, y \neq \text{-} \text{ and } x \neq y; \\ 0, & \text{otherwise.} \end{cases}$$

从上述定义可以看出，算法得出的单体型中错误的 SNP 数占整个两个单体型的总 SNP 数 $2n$ 的比例越小，重建率 RR 就越高。

实验中的单体型采用 2 种方式得到，第一种与文献[68]相同，采用来自公

开数据库的真实的单体型，本文实验采用的真实单体型数据来自于国际人类基因组单体型图计划[13]2006 年 7 月发布的数据文件 genotypes_chr1_CEU_r21_nr_fwd_phased.gz①，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 个个体的单体型，每个单体型有 SNP 位点 193333 个，本文实验随机选择一个个体指定长度的一对单体型。第二种跟文献[127]、[134]和[143]一样用计算机模拟生成，即首先随机生成指定长度的单体型，根据指定的两个单体型的差异率来随机生成另一个单体型，本文采用差异率与文献[134]一样，为 20%。

由于原始的 DNA 片段测序数据很难得到，在得到一对单体型的基础上，上述文献均根据指定的参数利用计算机来随机生成片段数据集。实验室中，Sanger 双脱氧链终止法的 DNA 测序误差约为 1%[144]，片段的覆盖度约为 5[1, 138]。为了使模拟生成的片段数据能很好的反映真实情况，与文献[134]一样，本文采用著名的 shotgun 测序模拟数据生成器 Celsim[145]。生成的片段数则按（单体型长度×片段覆盖度/片段平均长度）设置。最后在 Celsim 生成的片段数据的基础上，根据指定的测序误差 e 对片段的 SNP 值进行随机翻转，植入测序错误。模拟数据生成器的详细情况请参照文献[134]和[145]。

3.4.1 无空隙 MSR 和 MFR 算法性能比较

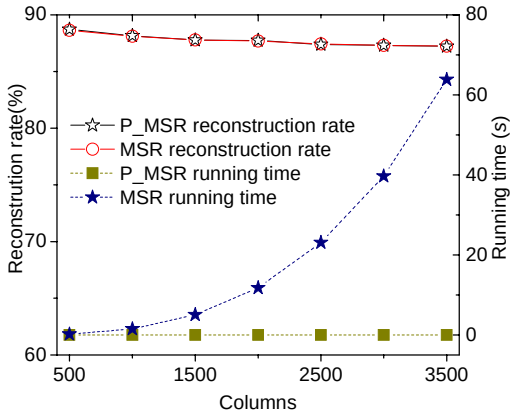
本小节对适用于无空隙 SNP 矩阵的四个算法 MSR、MFR、P_MSR 和 P_MFR 进行比较。

生成无空隙的片段数据采用参数如下：测序误差为 1%，生成片段的最小长度为 3；在没有特别说明的情况下，片段的最大长度为 7，片段覆盖度跟文献[134]一样设为 10。

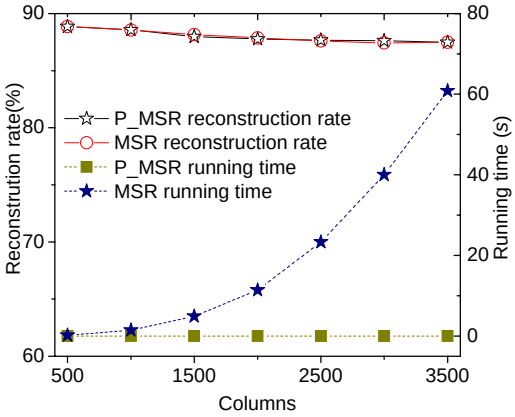
图 3-4 到图 3-9 的每一个点均为 100 次重复测试的平均值。下面各图中，图(a)均为在真实单体型数据上的实验结果，图(b)均为在模拟的单体型数据上的实验结果。因为在模拟的单体型数据上的实验结果与在真实单体型数据上的实验结果基本相同，所以下面主要讨论在真实单体型数据上的实验结果。每个子图中，左边的 Y 轴表示算法的重构率，右边的 Y 轴表示算法的运行时间。

图 3-4 和图 3-5 显示算法性能随 SNP 位点数（对应 SNP 矩阵的列数）变化的情况。图 3-4 (a) 中，当位点数为 500 时，P_MSR 和 MSR 算法的重构率分别是 88.71%和 88.64%，运行时间为 0.0001 和 0.191 秒；当位点数增加到 3500 时，

① 从 http://www.hapmap.org/downloads/phasing/2006-07_phaseII/phased/ 下载而来



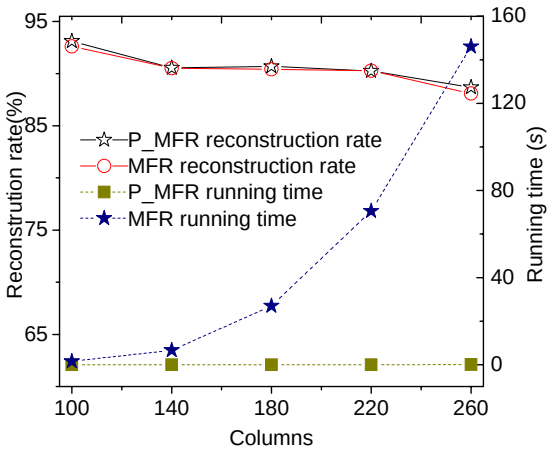
(a) 真实的单体型数据



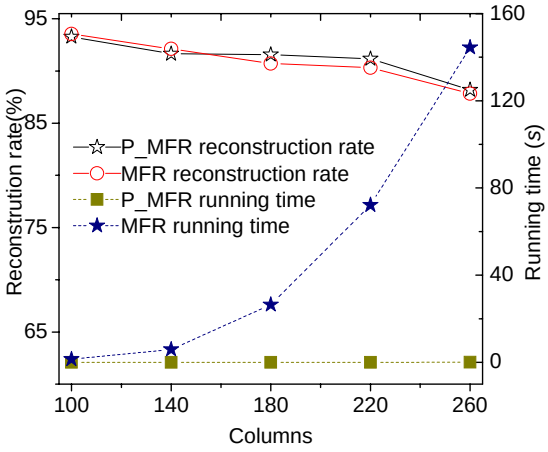
(b) 模拟的单体型数据

图 3-4 SNP 矩阵的列数变化时 P_MSR 和 MSR 的性能对比

P_MSR 和 MSR 算法的重构率分别是 87.22%和 87.24%，运行时间分别是 0.004 和 63.82 秒。



(a) 真实的单体型数据



(b) 模拟的单体型数据

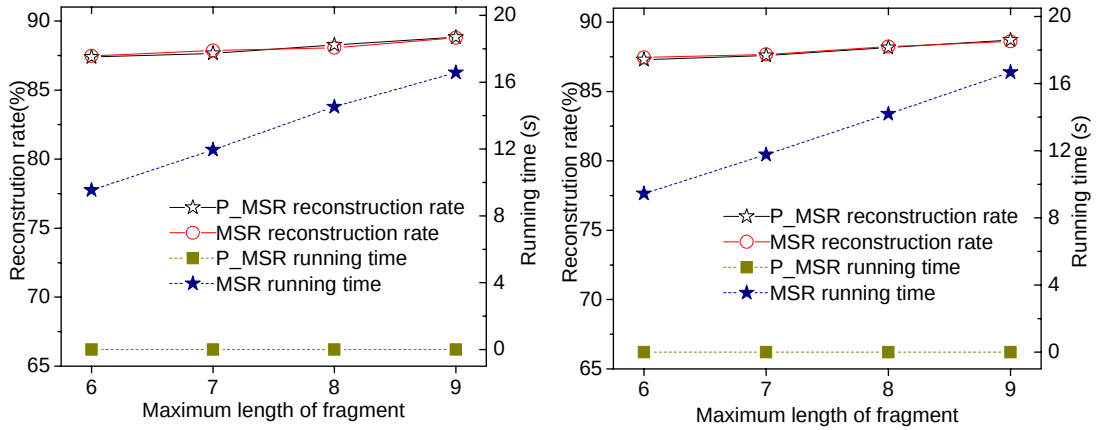
图 3-5 SNP 矩阵的列数变化时 P_MFR 和 MFR 的性能对比

图 3-5(a) 中，当位点数为 100 时，P_MFR 和 MFR 算法的重构率分别是 93.11%和 92.62%，运行时间分别是 0.009 和 0.67 秒；当位点数增加到 260 时，P_MFR 和 MFR 算法的重构率分别是 88.68%和 88.10%，运行时间分别是 0.055 和 146.05 秒。

从图 3-4 和图 3-5 可以看出，随着 SNP 位点数的增加，算法的单体型重构精度有下降趋势；MFR 和 MSR 运行时间显著增长，这是因为当 SNP 位点数增

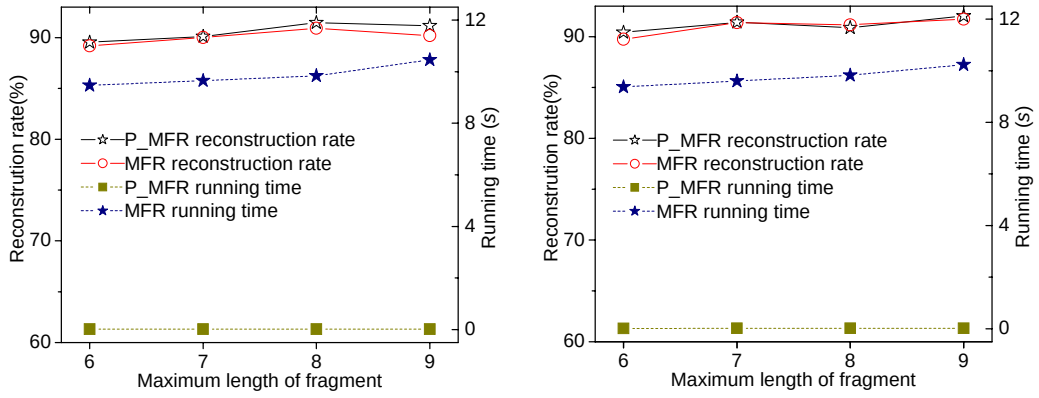
加，覆盖度不变时，片段数也随着增加，所以 MFR 和 MSR 运行时间的增长速度是位点数增长速度的 3 次方，而 P_MFR 和 P_MSR 的时间则基本成线性增长。

在片段数和其他参数保持不变的的条件下，图 3-6 和图 3-7 通过改变片段的最大长度和 SNP 位点数来比较各算法的性能。 图 3-6 的片段数保持 4000 不变。图 3-6(a) 中，在片段最大长度为 6、SNP 位点数为 1800 时，P_MSR 和 MSR 算法的重构率分别是 87.39%和 87.47%，运行时间分别是 0.001 和 9.5 秒；当片段最大长度为 9、SNP 位点数为 2400 时，P_MSR 和 MSR 算法的重构率分别是 88.82%和 88.77%，运行时间分别是 0.001 和 16.6 秒。



(a) 真实的单体型数据 (b) 模拟的单体型数据

图 3-6 片段最大长度变化时 P_MSR 和 MSR 算法性能对比



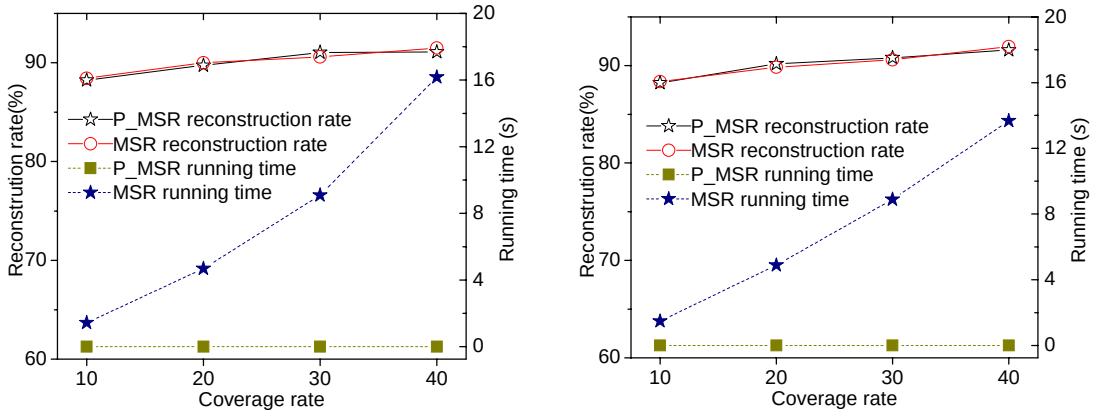
(a) 真实的单体型数据 (b) 模拟的单体型数据

图 3-7 片段最大长度变化时 P_MFR 和 MFR 算法性能对比

图 3-7 的片段数保持 300 不变。图 3-7 (a) 中，在片段最大长度为 6、SNP

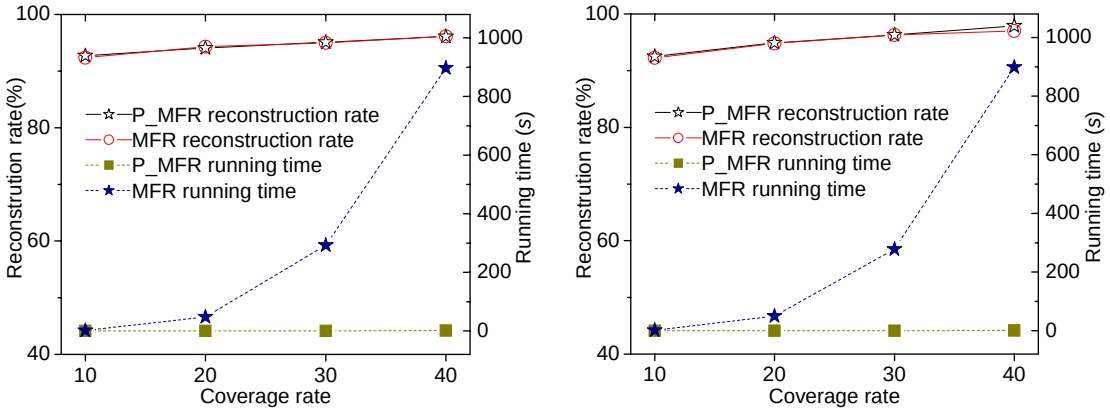
位点数为 135 时，P_MFR 和 MFR 算法的重构率分别是 89.53%和 89.19%，运行时间分别是 0.019 和 9.47 秒；当片段最大长度为 9、 SNP 位点数为 180 时，P_MFR 和 MFR 算法的重构率分别是 91.17%和 90.19%，运行时间分别是 0.025 和 10.46 秒。

图 3-6 和图 3-7 的实验结果说明，当片段的最大长度和 SNP 位点数增加而片段数不变时，算法的单体型重构精度有上升趋势；在运行时间方面，MFR、P_MFR 及 P_MSR 的时间变化不大，而 MSR 有较大的变化，这是因为从时间复杂度理论分析上看，MSR 的运行时间应与位点数 n 的 2 次方成正比，而其他算法应与位点数 n 或片段的最大长度 k_1 成正比。



(a) 真实的单体型数据　　(b) 模拟的单体型数据

图 3-8　片段覆盖度变化时 P_MSR 和 MSR 的性能对比



(a) 真实的单体型数据　　(b) 模拟的单体型数据

图 3-9　片段覆盖度变化时 P_MFR 和 MFR 的性能对比

图 3-8 和图 3-9 则是在 SNP 位点数和其他参数保持不变的条件下，通过改

变片段的覆盖度来改变片段数，从而比较各算法的性能。在图 3-8 (a)中，SNP 位点数保持 1000 不变。当片段覆盖度为 10，片段数为 2000 时，P_MSR 和 MSR 算法的重构率分别是 88.2%和 88.4%，运行时间为 0.001 和 1.44 秒；当片段覆盖度增加到 40，片段数为 8000 时，P_MSR 和 MSR 算法的重构率分别是 91.1%和 91.5%，运行时间分别是 0.004 和 16.17 秒。

在图 3-9 (a)中，SNP 位点数保持 100 不变。当片段覆盖度为 10 时，片段数为 200 时，P_MFR 和 MFR 算法的重构率分别是 92.72%和 92.38%，运行时间为 0.01 和 1.50 秒；当片段覆盖度增加到 40，片段数为 800 时，P_MFR 和 MFR 算法的重构率分别是 96.10%和 96.13%，运行时间分别是 0.99 和 897.14 秒。

图 3-8 和图 3-9 的实验结果说明，当位点数保持不变的条件下，片段覆盖度增加，片段数增加，SNP 位点的最大覆盖度（k_2）也会随之增加，算法单体型重构精度上升；在运行时间方面，P_MSR 及 P_MFR 的时间变化不大，MSR 基本上呈线性变化，但 MFR 的运行时间显著上升。这次实验结果也说明了算法实际运行时间与其时间复杂度理论分析是吻合的（MFR 的运行时间与片段数 m 的 3 次方成正比，MSR 和 P_MSR 只与片段数 m 或最大覆盖度 k_2 成正比，而 P_MFR 与最大覆盖度 k_2 的平方成正比，但由于 k_2 的值比片段数小得多，所以 P_MFR 所需的时间仍然很小）。

上述实验结果表明，无论是对于真实的单体型数据还是模拟的单体型数据，P_MFR（或 P_MSR）在运算速度上明显优于 MFR（或 MSR）；在单体型重构精度上，P_MFR 和 MFR（或 P_MSR 和 MSR）没有实质的差别，这是因为 P_MFR 和 MFR（或 P_MSR 和 MSR）都是精确算法，即对于保留的行数（或列数）而言，它们的结果是完全相等的，尽管在保留的具体行（或列）上可能存在随机性，算法的单体型重构精度不完全耦合，但是总体上来说没有优劣之分。

3.4.2 有空隙 MSR 和 MFR 算法性能比较

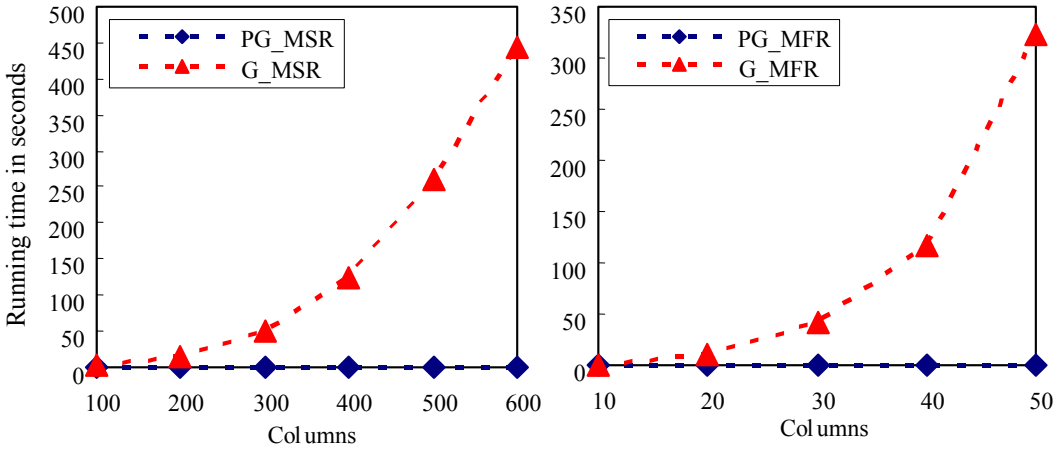
本小节对适用有空隙 SNP 矩阵的四个算法 G_MSR、G_MFR、PG_MSR 和 PG_MFR 进行比较。作为精确算法，在单体型重构精度上，PG_MFR 和 G_MFR（或 PG_MSR 和 G_MSR）没有实质的差别，因此下面只列出算法运行时间对比的结果。

有空隙的片段数据的生成方法如下：在一对模拟生成的单体型数据基础上，采用著名的 shotgun 测序模拟数据生成器 Celsim[145]来生成 m_1 条长度在 3 和 6 之间随机变化的片段和 m_2 条中间有 k 个连续的洞的长为 $6\times2+k$ 的片段，后面

类型的片段称为 mate-pair。最后在 Celsim 生成的片段数据的基础上，采用大小为 1%测序误差对片段的 SNP 值进行随机翻转，植入测序错误。

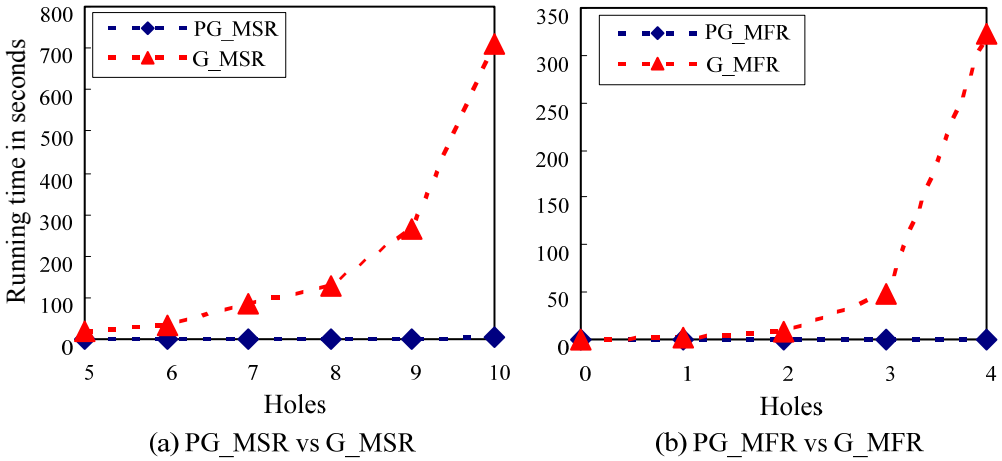
实验中的参数 k 和单体型长度即 SNP 位点数 n，在一个区间内变化，m_1、m_2 的大小相应发生变化，使得 mate-pair 的覆盖度为 2，普通片段的覆盖度为 10，其中覆盖度=片段平均长度×片段数/单体型长度。

下面的图中每一个数据点均为 100 次重复测试的平均值。



(a) PG_MSR vs G_MSR (b) PG_MFR vs G_MFR

图 3-10 单体型长度增大时算法的运行时间对比



(a) PG_MSR vs G_MSR (b) PG_MFR vs G_MFR

图 3-11 片段中最大洞的个数增大时算法的运行时间对比

图 3-10 中的实验结果显示单体型长度（即 SNP 矩阵的列数 n）增加，而片段覆盖度保持不变时，G_MSR 和 G_MFR 算法的运行时间显著上升。图 3-10 (a) 显示：当 $n = 100$, 普通片段数 $m_1 = 232$，mate-pair 片段数 $m_2 = 10$, mate-pair

片段中洞的个数 k = 8 时, G_MSR 和 PG_MSR 的运行时间分别是 1.58 和 0.10 秒； 当 n 增大到 600, m_1 和 m_2 分别增大到 1392 和 60, G_MSR 和 PG_MSR 的运行时间分别增大到 445.11 和 1.03 秒。

图 3-10 (b) 显示：当 n = 10, m_1 和 m_2 分别是 22 和 0 时, G_MFR 和 PG_MFR 算法的运行时间都是 0.03 秒, 当 n 增大到 50, m_1 和 m_2 分别增大到 110 和 6, k = 4 时, G_MFR 和 PG_MFR 的运行时间分别增大到 323.10 和 0.71 秒。从算法的时间复杂度分析可以看出，当 n 增加，覆盖度不变时，片段数 m=m_1+m_2 也随着增加，G_MSR 和 G_MFR 运行时间的增长速度是 n 增长速度的 3 次方，而 PG_MFR 和 PG_MSR 的时间则基本上成线性增长。

图 3-11 中的实验结果显示片段中最大洞的个数 k 增加，而 n 和片段覆盖度保持不变时，各算法的运行时间变化情况。图 3-11 (a) 显示当 k 从 5 增大到 10，mate-pair 片段数 m_2 从 46 减少到 36，而 n 保持 400、普通片段数 m_1 保持 930 不变时，G_MSR 和 PG_MSR 算法的运行时间变化情况：当 k＝5 时，G_MSR 和 PG_MSR 的运行时间分别是 17.97 和 0.08 秒；当 k 增大到 10 时, G_MSR 和 PG_MSR 的运行时间分别增大到 707.88 和 3.74 秒。

图 3-11 (b) 显示当 k 从 0 增大到 4，而 n 保持 50、mate-pair 片段数 m_2 保持 6、普通片段数 m_1 保持 22 不变时，G_MFR 和 PG_MFR 算法的运行时间变化情况：当 k＝0 时，G_MFR 和 PG_MFR 的运行时间分别是 0.61 和 0.01 秒；当 k 增大到 4 时, G_MFR 和 PG_MFR 的运行时间分别增大到 323.10 和 0.71 秒。

从时间复杂度分析来说，当片段中的最大洞的个数 k 增加时，四个算法的时间均呈指数增长。从图 3-11 的实验结果可以看出，G_MSR 和 G_MFR 的运行时间增长明显，而 PG_MSR 和 PG_MFR 算法运行的时间还是比较小的。

3.5 本章小结

对于 m 个 DNA 片段，n 个 SNP 位点的无空隙的 SNP 矩阵的 MSR 和 MFR 问题，Bafna 等[62]提出过时间复杂度分别为 $O(mn^2)$和 $O(m^2n+m^3)$、空间复杂度分别为 $O(mn+n^2)$和 $O(mn+m^3)$的多项式算法。当 DNA 测序仪对 DNA 片段的某些碱基无法确定其值，这时 DNA 片段上就会出现空隙。另外，如果在 DNA 测序中采用了 mate-pair，则 mate-pair 片段中间也会出现空隙。有空隙的 MSR 和 MFR 问题是 NP-难的[17]，也是 APX-hard[62]。为了求解有空隙的 MSR 和 MFR

问题，Bafna 等[62]曾提出过时间复杂度分别为是 $O(2^k n^2 m)$和 $O(nm^2 2^{2k}+2^{3k}m^3)$、空间复杂度分别为 $O(nm+2^k n^2)$ 和 $O(nm+2^{2k}m^3)$的算法，其中 k 为片段中最大洞（hole）数。

目前的 DNA 测序技术能直接测定的 DNA 片段所覆盖的最大 SNP 位点数 k_1 通常小于 10。为了节约时间和金钱，实际的 DNA 测序中覆盖一个 SNP 位点的最大片段数 k_2 也不是很大，通常小于 19。与要测定的单体型 SNP 位点总数 n 及所测序的 DNA 片段总数 m 相比，k_1 和 k_2 均很小。

根据以上事实，本章对 MSR 和 MFR 问题进行了参数化建模，设计了时间复杂度分别为 $O(nk_1k_2+mlogm+mk_1)$和 $O(mk_2^2+mk_1k_2+mlogm+nk_2)$和空间复杂度分别为 $O(mk_1+nk_1)$ 和 $O(mk_1+mk_2^2)$的精确算法 P_MSR 和 P_MFR 来求解无空隙 MSR 和 MFR。为求解有空隙的 MSR 和 MFR，本章设计了时间复杂度分别为 $O(2^k nk_1k_2+mlogm+nk_2+mk_1)$和 $O(2^k mk_1k_2+2^{3k}mk_2^2+mlogm+nk_2+mk_1)$、空间复杂度分别为 $O(nk_1 2^k+k_2k_1 2^k+mk_1)$ 和 $O(nm+k_2^2 m2^{2k})$的精确算法 PG_MSR 和 PG_MFR。大量实验结果表明，在 Bafna 等[62]的对应算法基础上，上述算法的效率显著提高，适用于全基因组规模上的单体型组装。

第四章　有长 Mate-Pair 时 MSR 和 MFR 参数化算法

4.1　引言

在使用 shotgun 进行全基因组 DNA 测序常常使用 mate-pair[3, 138, 142]来提高 DNA 组装的精度。Mate-pair 是采用双管测序法从两端对一段较长的 DNA 克隆进行测序，得到位于克隆两端的长度在 1kb 以内，彼此间的距离已知的两个读段，这两个读段构成一个 mate-pair。图 4-1 给出了一对平均距离为 μ,标准差为 σ 的片段构成的一个 mate-pair 示意图。

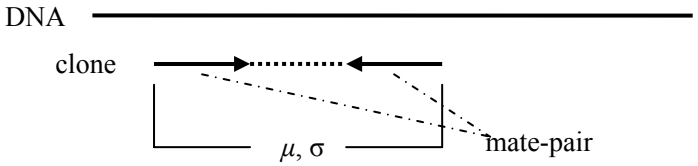


图 4-1 由一对平均距离为 μ,标准差为 σ 的片段构成的 Mate-pair

根据克隆（clone）的载体不同，mate-pair 的典型平均长度有 2kb、10kb、50kb 和 150kb[142]。如果片段测序数据中出现的 mate-pair 的长度达到了 150kb, 在这种情况下，片段中的洞的个数则会达到 100 个以上(平均 1kb 有 1 个 SNP 位点[1, 7, 138])。

虽然，对于片段中有洞的情况，Bafna 等[62]曾提出过时间复杂度分别为 $O(2^k n^2 m)$和 $O(nm^2 2^{2k}+2^{3k}m^3)$、空间复杂度分别为 $O(nm+2^k n^2)$ 和 $O(nm+2^{2k}m^3)$ 的算法求解 MSR 和 MFR 问题，其中 k 为片段中洞的最大个数，但当片段数据中包含长的 mate-pair 时，k 的取值将较大，上述算法将不再可行；本论文上一章提出的算法也是如此。因此需要进一步研究含有长 mate-pair 片段数据的 MSR 和 MFR 问题的实用算法。

本章组织如下：4.2 节研究有长 mate-pair 的 MSR 参数化模型和算法，4.3 节研究对应的 MFR 参数化模型和算法，4.4 节对本章小结。

4.2　有长 Mate-Pair 时 MSR 参数化算法

4.2.1　有长 Mate-Pair 片段时 MSR 参数化模型

在有mate-pair片段数据的情况下，虽然mate-pair片段中洞的最大数可达100个，但是，如前一章所叙，因为测序的成本和所需的时间与覆盖度成正比，所以，实验室测序时片段的覆盖度(coverage)是不太大的。Celera公司的人类基因组测序工程中的片段覆盖度为5.11[138]、国际人类基因组测序联盟（IHGSC）在人类基因组计划中的片段数据的覆盖度是4.5[1]。在这两个人类全基因组测序工程中，都采用了mate-pair片段数据。虽然在人类基因组的每个位置的片段覆盖程度并不完全相同，但是，Huson等[142]对Celera公司的人类基因组测序工程中的片段数据[138]进行了详细的分析，从其片段覆盖图上可看出绝大部分位点上的片段数为5左右，最大值没有超过19。从上可以看出，覆盖一个SNP位点的片段数远小于片段总数。覆盖某一SNP位点的所有片段中，在该位点上取空值的片段数肯定不大于覆盖该位点的片段总数;另外，一个片段覆盖的SNP位点数一般比SNP位点总数小（尽管SNP分布不均匀，从已有的数据[47, 140, 141]来看，长为150kb的mate-pair，其覆盖的SNP位点数约为150个左右）。

根据以上事实，特提出以下参数化条件:

定义 4.1　$(k_1, k_2; h)$**参数化条件**:　k_1, k_2, h是正整数，$(k_1, k_2; h)$参数化条件定义为任意片段覆盖的SNP位点数不超过k_1（从第一个非‘–’值SNP位点到最后一个非‘–’值SNP位点之间最多只有k_1–2个位点，中间允许出现任意多个洞），对于任意一个SNP位点，覆盖该位点的片段数不超过k_2，而其中在该位点上取空值的片段数不会超过h。

对于一个SNP矩阵，满足$(k_1, k_2; h)$参数化条件就是该矩阵的任意一行覆盖的列数不超过k_1，即该行取非‘–’值的两列之间最多有k_1–2列，覆盖任意一列的行数不超过k_2，其中在该列上的值为‘–’值的行数最多为h。

图 4-2 给出了一个满足(9, 6; 2)参数化条件的11行×10列的SNP矩阵，其中，覆盖列数最多的为第2行，覆盖了从列2到列10的9个列; 覆盖第6列的行数最多，为6; 对于任意一列，覆盖该列的行中最多只有2行在该列上取‘–’值。

对于任意一个$m \times n$的SNP矩阵$\mathbf{M}$，对$\mathbf{M}$的所有行进行一次扫描，就可以统计出每行覆盖的列数、覆盖每列的行数及覆盖每列且在该列取‘–’值的行数，从每行覆盖的列数中取最大值就得到k_1 $(k_1 \le n)$，从覆盖每列的行数中取最大值就得到

k_2 ($k_2 \le m$)，同样从覆盖每列且在该列取‘–’值的行数中取最大值就得到h ($h \le k_2$)，这样，**M**就满足(k_1, k_2; h)参数化条件。在极端的情况下，$k_1 = n$，$k_2 = m$，$h = k_2$-2; 但根据本节第一自然段对已有的基因组测序片段数据的分析，从文献[141]可知，对Celera公司的这些片段数据对应的SNP矩阵而言，k_2=19，而h不会大于k_2-2。

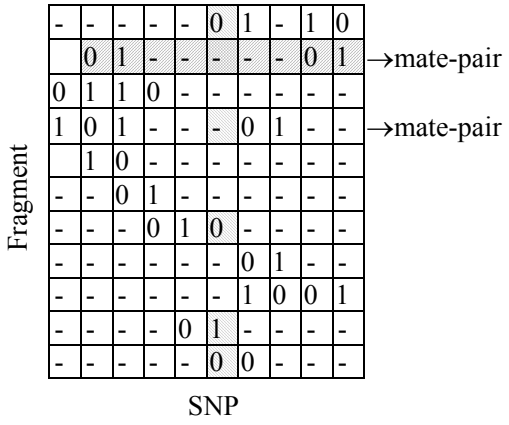


图 4-2　一个满足(9, 6; 2)参数化条件的 SNP 矩阵

为了对包含长mate-pair的片段数据有效求解，下面对MSR问题进行参数化:

问题 4.1　PM_MSR (parameterized minimum SNP removal with mate-pair): 给定一个满足(k_1, k_2; h)参数化条件的SNP矩阵**M**，PM_MSR问题是要求删除最少的列 (SNP位点)，也就是保留最多的列使**M**可行。

本节下面所述的PM_MSR的解指的是能保留下来的最大列数。

4.2.2　预处理

SNP 矩阵 **M** 仍然采用如下数据结构：对每一行 i，记下该行覆盖的最左边和最右边的列号，即 *left*(i)和 *right*(i)，记录该行从列 *left*(i)到列 *right*(i)的各列上的值。我们先对 SNP 矩阵 **M** 进行预处理，预处理的方法与 3.3.1 基本相同，但不需要对行进行排序，即：

Step 1 对每一列 j，得出覆盖它的行号的有序集 *rowset*(j)。

Step 2 去掉冗余列，修改受影响行的 *left* 和 *right* 函数值。

Step 3 去掉冗余行，修改受影响的有序集 *rowset*。

预处理所需的时间为 $O(nk_2+mk_1)$。

定理 4.1 一个满足(k_1, k_2; h)参数化条件SNP矩阵**M**经过以上的预处理后得到的 SNP 矩阵 **M**'满足(k_1, k_2; h')参数化条件, 其中 h' = min(max(k_2−2, 0), h)。

证明：去掉冗余列和冗余行的预处理显然不会增加对应的片段的长度、覆盖某一 SNP 位点的片段数和覆盖某一位点并在该位点取空值的片段数。而经过预处理以后，任何一列对应的 SNP 位点上至少应有一个片段取‘0’值，还有一个取‘1’值，所以覆盖任意 SNP 位点的取‘−’值的片段数不会大于 k_2-2，当然也不会比 0 小。　□

定理 4.2 令一个 SNP 矩阵 **M** 经过以上的预处理得到的 SNP 矩阵为 **M'**，其中去掉的列的集合为 **X**，则 **M** 的 PM_MSR 的解（这里指能保留的最大列数）等于 **M'**的 PM_MSR 的解与 **X** 的列数之和。

用证明定理 3.2 的方法可以证明定理 4.2，在此不再赘述。

从定理 4.2 可知，求解一个 SNP 矩阵的 PM_MSR 的解可以归结为求其预处理后形成的矩阵的对应解。

为了叙述的简便，我们称一个经过上述预处理后的 SNP 矩阵为精简的 SNP 矩阵。

本节下面的算法的输入均为预处理后得到的精简的 SNP 矩阵 **M**，该矩阵具有如下特点：任何一个列总有一些行的值是‘0’，还有一些行的值为‘1’。

4.2.3　PM_MSR 算法

对于一个精简的 SNP 矩阵 **M**，因为在列 j 上，有些行取‘0’值也有些行取‘1’值，所以如果列 j 被保留，要使 **M** 可行，则覆盖列 j 的行肯定必须划分成两个子集，并且在列 j 上的取值为‘0’的行应该划分到同一个子集；而取值为‘1’的行肯定应该划分到另一个子集中。这样只有在该列取‘−’值的行的划分才是不确定的。显然对于一个满足$(k_1, k_2; h)$参数化条件的精简的 SNP 矩阵 **M**，覆盖列 j 且在列 j 上的值为‘−’的片段数最多为 h, 那么覆盖列 j 的全体片段上所有可行的划分数最多是 2^h。

定义 4.2 划分函数：对某一列 j，令覆盖列 j 的所有行的序号集为 F，列 j 上的划分函数 P 定义为满足如下条件的映射 F→{0, 1}：对于覆盖列 j 的行 i，如果 $\mathbf{M}_{i,j}$ = ‘0’则 $P(j) = 0$；如果 $\mathbf{M}_{i,j}$ = ‘1’则 $P(j) = 1$；如果 $\mathbf{M}_{i,j}$ = ‘−’，则 $P(j)$可在 0 和 1 任选一个。

对于一个满足$(k_1, k_2; h)$参数化条件的精简的 SNP 矩阵 **M**，任意列上的划分函数的个数最多是 2^h。

从定义 4.2 可以看出，可以按照 P 函数值对覆盖列 j 的行进行划分，函数

值相同的行划分给同一子集。反过来，对覆盖列 j 的行的任意划分，如果划分出来的每个子集中的行互不冲突，则该划分必定对应着列 j 上的一个划分函数。

定义 4.3　$K[j, P]$和$Column[j, P]$: P为列j上的划分函数, $K[j, P]$定义为满足下列条件从列1到列j中能保留下的最多列数, 对应的保留下来的列的集合定义为$Column[j, P]$:

(1) 列j必须保留;

(2) 对$\mathbf{M}(:, j)$中的所有行存在着一个划分, 使得划分在同一子集中的行在$Column[j, P]$中的所有列上均不冲突, 且划分在同一子集中的行,如果其覆盖列j, 则其P函数值必相同。

显然,保留$Column[j, P]$中的列可使$\mathbf{M}(:, j)$可行。对于一个精简的SNP矩阵第1列而言, 覆盖第1列的行在第1列的值都是确定的, 所以片段的划分函数P是唯一的，且有

$$K[1, P] = 1; \quad Column[1, P] = \{1\} \tag{4-1}$$

定义 4.4 $OK(j, P)$: P为列j上的划分函数, $OK(j, P)$定义为所有满足下列条件的(j', P')的集合:

(1) P'为列j'上的划分函数且$\max(0, j-k_1) < j' < j$;

(2) P'和P兼容, 就是P'和P对既覆盖j'又覆盖j的行的划分不冲突, 即对任意既覆盖j'又覆盖j的两行, 如果它们的P'函数值相等, 则P函数值必相等, 这表示由P'划分在同一子集中的行, 按P来划分也必须在同一子集中。

对于一个满足$(k_1, k_2; h)$参数化条件的精简的SNP矩阵$\mathbf{M}$, 根据$K[j, P]$的定义，有下面的递推式（正确性见定理4.3的证明）

$$K[j, P] = \max(A[j - k_1], \max_{(j', P') \in OK(j, P)}(K[j', P'])) + 1 \tag{4-2}$$

其中, $A[j]$为满足$j' \le j$的所有$K[j', P']$的最大值, 即

$$A[j] = \begin{cases} 0, & \text{if } j < 1; \\ 1, & \text{if } j = 1; \\ \max\limits_{\text{all possible } K[j', P'] \text{ with } j' \le j} K[j', P'], & \text{if } j > 1. \end{cases} \tag{4-3}$$

从公式(4-1)~(4-3)可以看出, 对任意j, $A[j]$不会比0小, 且$m \times n$矩阵$\mathbf{M}$的PM_MSR解就是$A[n]$。

根据公式(4-1)~(4-3)可以得到如下的PM_MSR算法。

PM_MSR 算法：

Input:　一个精简的 SNP 矩阵 **M**;

Output: **M** 的 MSR 解.

Step 1:　对 **M** 中所有的片段扫描一次, 得到 **M** 的行列数 m, n 及其 3 个参数 k_1, k_2, h;

Setp 2:　//初始化

Setp 2.1: **for** j = 0, ..., k_1 **do**

Setp 2.1.1:　$A[j] = 0$; $H[j] = 0$;

Setp 2.1.2:　**for** P = 0, ..., 2^h-1 **do** $K[j, P] = 0$;

//根据公式(4-1), *curColum* = 0 表示第 1 列

Setp 2.2: *curColum* = 0,　$j = 0$,　$K[0,0] = 1$;

Step 3: **while** *curColum* < $n-1$ **do**　//循环 $n-1$ 次

Step 3.1:　*curColum*++,　$j = (j+1) \bmod (k_1+1)$;

Step 3.2:　$H[j]$ = 覆盖列 *curColum* 的行中在 *curColum* 上取'−'值的行数;

Step 3.3:　**for** $P = 0, ..., 2^{H[j]}-1$ **do**　//P 是一个在 *curColum* 列上的划分函数

Step 3.3.1:　$K[j, P] = A[(j+k_1+1-k_1) \bmod (k_1+1)] + 1$;

Step 3.3.2:　**for** pre = 1, …, (k_1-1) **do**

Step 3.3.2.1:　$q = (j+k_1+1-pre) \bmod (k_1+1)$;

// p 是一个在 *curColum*−*pre* 列上的划分函数

Step 3.3.2.2:　**for** $p = 0, \ldots, 2^{H[q]}-1$ **do**

Step 3.3.2.2.1:　**if** $K[j, P] > K[q, p]$ **continue**;

//判断列 *curColum* 的划分函数 P 和列(*curColumn*−*pre*)的划分函数 p 是否兼容

Step 3.3.2.2.2:　**else if** *Compatible*(*curColum*, *curColumn*−*pre*, P, p)

Step 3.3.2.2.2.1:　$K[j, P] = \max(K[j, P], K[q, p]+1)$;

Step 3.3.2.2.2.2:　$A[j] = \max(A[j], K[j, P])$; //目前看到的最大值;

Step 4:　**M** 的 MSR 解就是 $A[i]$.

从公式(4-2)可以看出, 计算$K[j, P]$只需知道$A[j-k_1]$以及列j前k_1-1列的相关K值（见$OK(j, P)$的定义）, 而更新$K[j, P]$时就可以同时检查$A[j]$是否需要更新, 所以, 算法需要保存的值是: $A[j-k_1]$到$A[j]$共(k_1+1)个单元, 从列$j-k_1+1$到j需要保存k_1列的相关K值, 虽然实际程序中K比A可以少1个单元, 但是为了叙述简便, 算法中A和K的值都是采用长度为k_1+1的循环队列来存储, 以减小空间需求。算法中需

要调用的判断在两个列上的划分函数是否兼容的*Compatible*子过程如下（用类C语言编写）。

```
Boolean Compatible(column1, column2, P, P′)
//判断列 column1 上的划分函数 P 和列 column2 上的划分函数 P′是否兼容
{    i = rowset(column1)和 rowset(column2)共有的第一行（如果没有共有的
       行，则 i = –1）;
     same = –1;
     while (i != –1) do
     { Case 1. M_{i,Column1} != '–' AND M_{i,Column2} != '–' :
                 if (same == –1) then same = (M_{i,Column1} == M_{i,Column2});
                 else if (same != (M_{i,Column1} == M_{i,Column2})) then return FALSE;
       Case 2. M_{i,Column1} == '–' AND M_{i,Column2} != '–' :
                 Char1 = (P 的最低位 == 1 ? '1' : '0');   P >> 1;
                 if (same == –1) then same = (Char1 == M_{i,Column2});
                 else if (same != (Char1 == M_{i,Column2})) then return FALSE;
       Case 3. M_{i,Column1} != '–' AND M_{i,Column2} == '–' :
                 Char2 = (P′的最低位 == 1 ? '1' : '0');   P′ >> 1;
                 if (same == –1) then same = (M_{i,Column1} == Char2)
                 else if (same != (M_{i,Column1} == Char2)) then return FALSE;
       Case 4. M_{i,Column1} == '–' AND M_{i,Column2} == '–' :
                 Char1 = (P 的最低位 == 1? '1' : '0');   P >> 1;
                 Char2 = (P′的最低位 == 1 ? '1' : '0'); P′ >> 1;
                 if (same == –1) then same = ( Char1 == Char2 );
                 else if ( same != (Char1 == Char2) ) then return FALSE;
       i = rowset(column1)和 rowset(column2)共有的下一行(如果没有了，则
            i = –1);
     }
     return TRUE;
}
```

定理 4.3 对于一个$m\times n$满足(k_1, k_2; h)参数化条件的SNP矩阵**M**, PM_MSR算

法能正确地求出其PM_MSR问题的一个解;该算法的时间复杂度为$O(nk_1k_2 2^{2h}+k_1 2^h+nk_2+mk_1)$, 空间复杂度为$O(k_1 n 2^h+nk_2+mk_1)$。

证明: 首先分析其时间复杂度: Step 1为$O(mk_1)$; Step 2为$O((k_1+1)2^h)$; Step 3执行$n-1$次, Step 3.2可以通过扫描覆盖*rowset*(*curColum*)中的行在列*curColum*的值得到, 总的时间为$O((n-1)k_2)$, 由于$H[j]$的值不会超过h, 所以Step 3.3最多循环$(n-1)2^h$次, Step 3.3.2最多循环$(n-1)(k_1-1)2^h$次。因为*rowset*(*column*1)和*rowset*(*column*2)共有的行数不会超过k_2, 所以, 判断两划分函数是否兼容的子过程*Compatible*所需时间$O(k_2)$。这样, Step 3.3.2.2的时间为$O((n-1)(k_1-1)k_2 2^{2h})$; 因此, 加上预处理，PM_MSR算法的时间复杂度为$O(nk_1k_2 2^{2h}+k_1 2^h+nk_2+mk_1)$。

下面再进行空间复杂度分析: 数组A和H所需的空间为$O(k_1+1)$, K所需的空间为$O((k_1+1)2^h)$, 如果还保留对应的$Column[j, P]$, 则要$O((k_1+1)n2^h)$的空间, *rowset*所需的空间为nk_2, 加上SNP矩阵**M**所需的空间$O(mk_1)$, 算法总的空间复杂度为$O(k_1 n 2^h+nk_2+mk_1)$。

PM_MSR算法的正确性证明: PM_MSR算法的正确性由初始值公式(4-1)和递推公式(4-2)的正确性来保证。

显然, 初始值$K[1, P] = 1$是正确的, 下面利用数学归纳法来证明递推公式(4-2)的正确性。

假设对于任意$j'<j$和在列j'上的任意划分函数P', $K[j', P']$都是正确的, 现在来证明递推公式(4-2)能正确地求出$K[j, P]$。

对于列j前的某一列$j'>j-k_1$, 如果在该列存在与P兼容的划分P', 即$(j', P') \in OK(j, P)$, 则根据定义4.3中的条件(2), 对**M**(:, j')中所有的行存在着一个划分, 使得划分在同一子集中的行在$Column[j', P']$中的列上均不冲突, 且划分在同一子集中的行, 如果其覆盖列j', 则其P'函数值必相同, 再根据定义4.4, 这些行如果同时也覆盖了列j, 则其P函数值必相等, 这样根据定义4.2, 这些行在列j上也不会冲突;而那些没有覆盖列j的行, 它们自然不会在列j和任何其他行冲突。如果只保留$Column[j',P']$中的列和列j, 对于**M**(:, j)而言, 还需要考虑的只有**M**(:, j)中覆盖了列j但是没有覆盖列j'的行, 也就是说这些在$Column[j', P']$上的取值为空的行（注意: 列j'是$Column[j', P']$中的最后一列），这些行则可以按照其P函数值分配到具有相同P函数值的其他同时覆盖列j和j'的行所在的子集中, 如果没有同时覆盖列j和j'的行, 则按P函数值的不同分配到不同的子集中。至于其他行, 由于不覆盖$Column[j',P']$中的列, 也不覆盖列j, 它们在$Column[j',P']$中的列和列j上肯定不会与其他行冲突, 可以任意分配。这样就对**M**(:, j)中的所有行存在着一个划分,

使得划分在同一子集中的行在$Column[j', P']$中的列及列j上均不冲突, 从而满足定义4.3中的条件(2), 由于列j保留, 定义4.3中的条件(1)也满足, 所以有$Column[j, P]$至少比$Column[j', P']$多一列, 这一列就是列j, 因此有$K[j, P] \geq K[j', P'] + 1$。

如果列$j' \leq j - k_1$, 由于SNP矩阵$\mathbf{M}$满足$(k_1, k_2; h)$参数化条件, 所以没有一行覆盖的列数可以超过k_1, 也就是说, 没有一行可以同时覆盖列j和列j'或列j'前面的列,否则该行覆盖的列数将超过k_1。对列j'上任意分组函数P'而言, 因为列j'是$Column[j', P']$中的最后一列, 所以覆盖$Column[j', P']$中任意列的行都不会覆盖列j, 它们在列j上的值应为空, 按照定义4.3, 这些行总存在一个划分, 使得同一子集中的行在$Column[j', P']$中的列上不会冲突, 由于它们在列j上的值都为空, 所以它们也不会在列j上冲突; 和$\mathbf{M}(:, j')$比较, $\mathbf{M}(:, j)$中需要多考虑的行是那些覆盖列j的行, 这些行在$Column[j', P']$中所有列上的取值必定为空, 所以它们不会和其他行在$Column[j',P']$中的列上冲突, 按照P函数值对这些行进行划分, 值为'0'的加入到其中一个子集中, 值为'1'的加入到另一个子集中。根据P函数的定义, 加入在同一子集中的行在列j上是不会冲突的; 而其他既不覆盖$Column[j', P']$中的列, 也不覆盖列j的行, 它们在$Column[j',P']$中的列及列j上不会和任何其他行冲突, 对它们进行任意分配。这样, 在$\mathrm{M}(:, j)$的所有行上存在一个划分, 使得划分在同一子集中的行在$Column[j', P']$的列及列j上均不冲突, 满足定义4.3的条件(2), 由于列j保留, 定义4.3的条件(1)也满足, 所以对任意$j' \leq j - k_1$及其上的任意分组函数P', 有$K[j,P] \geq K[j',P'] + 1$, 即$K[j,P] \geq A[j - k_1] + 1$。

这样就说明$K[j, P]$至少要比$A[j - k_1]$和$K[j',P']$大1 (对所有的$(j', P') \in OK(i, P)$)。

反过来, 对于任意列j和列j上的划分函数P来说, 按照定义4.3, 对$\mathrm{M}(:, j)$中的所有行存在一个划分F, 使得划分在同一子集中的行在$Column[j, P]$中的所有列上均不冲突, 且划分在同一子集中的行, 如果其覆盖列j, 则其P函数值必相同。那么在同一子集中的行必定在$Column[j, P] - \{j\}$中的所有列上均不冲突, 即在$Column[j, P] - \{j\}$中的同一列上取'0'的行和取'1'值的行不会出现在同一个子集里 (否则它们就在该列冲突)。如果$Column[j, P]$中只有列j, 那么$K[j, P] = 1$, 根据公式(4-3), 任意$A[j - k_1]$都不比0小, 所以有$A[j - k_1] \geq K[j, P] - 1$; 如果$Column[j, P]$中除列$j$外还有其他列, 则令$Column[j, P]$中除列$j$外最后一列为$j'$, 构造列$j'$上的划分函数$P'$: 由于$\mathbf{M}$是一个精简的SNP矩阵, 所以在列$j'$上必有一些行取'0'值, 还有一些行取'1'值, 在划分F中, 这些在列j'上取'0'值的行必定在其中同一个子集中, 令该子集中覆盖列j'的所有行的P'值为0; 那些在列j'上取'1'值的行必定在另一个子集中, 令该子集中覆盖列j'的所有行的P'值为1。显然, 保留$Column[j, P]$

$-\{j\}$中的列能使M(:, j')可行, 并且存在划分F满足定义4.3的条件(2), 条件(1)显然满足, 这样就有$K[j', P'] \geq K[j, P]-1$, 如果$j'>j-k_1$, 根据P'的构造方法可知$(j', P') \in OK(j, P)$; 如果$j' \leq j-k_1$则$K[j',P'] \leq A[j-k_1]$, 就有$A[j-k_1] \geq K[j, P]-1$。

这样就说明, 不是$A[j-k_1] \geq K[j, P]-1$成立,就是存在一个$(j', P') \in OK(j, P)$使得$K[j', P'] \geq K[j, P]-1$成立。

递推公式(4-2)成立, 定理得证。　□

4.2.4　实验结果

我们对 PM_MSR 算法和 Bafna 等[62]提出的求解有空隙 MSR 的算法的性能进行对比测试。我们用 C++语言实现了 PM_MSR 算法，通过对 Fast hare 源程序[133]中的对应程序的修改实现了 Bafna 等[62]的对应算法，该算法记作 G_MSR。

作为精确算法，PM_MSR 和 G_MSR 在保留的列数上，结果是完全相等的，尽管在保留的具体列上可能存在随机性，算法的单体型重构精度不完全耦合，但总体上没有优劣之分。因此下面只进行算法运行时间对比测试。

测试平台与前一章的完全相同。测试用的输入数据的产生方法也与前一章基本相同，即：首先随机生成指定长度的单体型，根据指定的两个单体型的差异率来随机生成另一个单体型，本节采用差异率为 10%。

然后采用著名的 shotgun 测序模拟数据生成器 Celsim[145]来生成 m_1 条长度在 3 和 6 之间随机变化的片段和 m_2 条中间有 d 个连续的洞的长为 $6\times2+d$ 的 mate-pair 片段。最后在 Celsim 生成的片段数据的基础上，根据指定的测序误差 e 对片段的 SNP 值进行随机翻转，植入测序错误。根据生成洞的概率 p 对普通片段的 SNP 值置空。本节与文献[133]一样采用 e=5%，p=1%。

模拟数据生成的详细情况请参照本文第四章实验部分及文献[134]和[145]。

实验中的参数 d 和单体型长度即 SNP 位点数 n，在一个区间内变化，m_1、m_2 的大小则按照下面的公式由普通片段的覆盖度和 mate-pair 的覆盖度确定：片段数=单体型长度×覆盖度/片段平均长度。

下面的实验结果中，除图 4-7 为 20 次重复测试的平均值, 其他图的每一个点均为 100 次重复测试的平均值。

图 4-3 的结果显示, 当片段覆盖度为 10 (普通片段的覆盖度为 6、mate-pair 的覆盖度为 4)、mate-pair 中洞的个数 d 为 10 个 SNP 位点时, G_MSR 和 PM_MSR 的运行时间与测试的 SNP 位点数多少的关系。在 SNP 位点数为 200 时, G_MSR 和 PM_MSR 算法的运行时间分别是 1.6375 秒和 0.001 秒; 位点数增加到 600 时,

G_MSR 达到 71.397499 秒，PM_MSR 用了 0.0035 秒。这是因为当 SNP 位点数增加，覆盖度不变时，片段数也随着增加，在算法复杂度分析上，G_MSR 运行的时间的增长速度应该是位点数增长速度的 3 次方，而 PM_MSR 的时间则基本成线性增长。

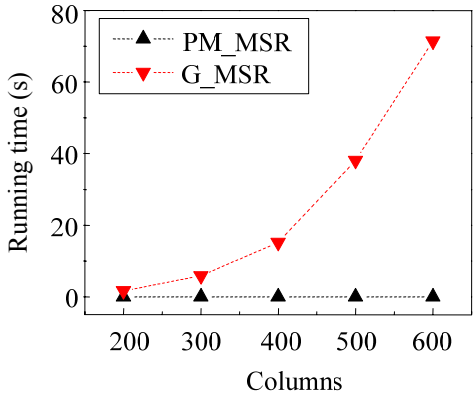


图 4-3　SNP 矩阵的列数变化时算法的运行时间对比

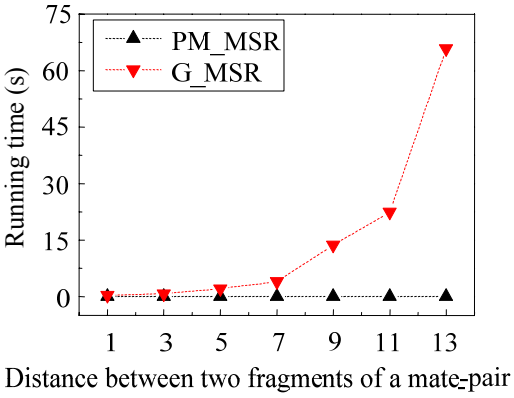


图 4-4　当 mate-pair 两片段的间距发生变化时算法的运行时间对比

图 4-4 则是当片段覆盖度为 10 (普通片段的覆盖度为 6、mate-pair 的覆盖度为 4)、SNP 位点数为 400、mate-pair 中洞的个数 d 从 1 个到 13 个 SNP 位点发生变化时，两算法的运行时间曲线。当 d 为 1 时，G_MSR 算法需 0.3285 秒，PM_MSR 需 0.0005 秒；当 d 增加到 13 时，G_MSR 和 PM_MSR 的运行时间分别是 65.776497 和 0.0035 秒。这次实验的结果说明，G_MSR 对 k 的大小非常敏感，当 d 增加时，k 随着增加,其运行时间急剧上升；但 PM_MSR 的时间变化却不大，这说明 PM_MSR 对 k 不敏感。

图 4-5 是当普通片段的覆盖度为 6、mate-pair 中洞的个数 d 为 10、SNP 位点数为 400、mate-pair 的覆盖度从 2 到 18 发生变化时 (即片段覆盖度从 10 增长到 24)，两算法的运行时间曲线图。当 mate-pair 的覆盖度为 2 时，G_MSR 和 PM_MSR 算法的运行时间分别是 5.391 5 和 0.001 5 秒;当 mate-pair 的覆盖度增大到 18 时，两算法的运行时间分别增大到 286.44 和 49.57 秒。这个实验说明了 PM_MSR 算法对 mate-pair 的覆盖度敏感，这是因为 mate-pair 的覆盖度上升时，覆盖同一 SNP 位点且在该位点取空值的片段数就会增加，导致 h 上升，PM_MSR 算法的时间呈指数上升。从实验结果看，即使在 mate-pair 的覆盖度上升到 18 时，PM_MSR 算法所需的时间比 G_MSR 还是要少得多。

当然，如果 mate-pair 的覆盖度继续增大，那么在覆盖某一位点的片段中，

在该位点取空值的片段也会增加，当 $h>k$ 时，从算法的复杂度分析可以看出，PM_MSR 的性能肯定不如 G_MSR，所幸的是，做 DNA 测序时，出于成本和时间的考虑，片段的覆盖度不会很大(人类基因组测序中，覆盖度约为 5[1] [138])。

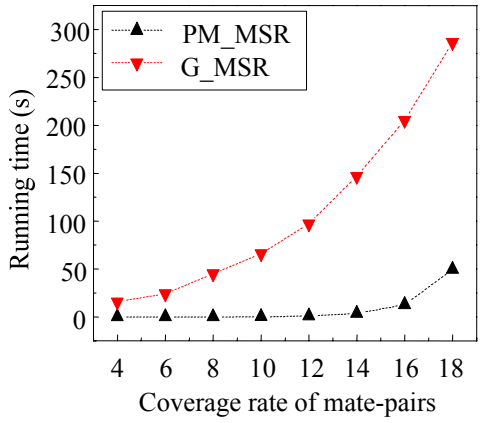


图 4-5 Mate-pair 的覆盖度发生变化时算法的运行时间对比

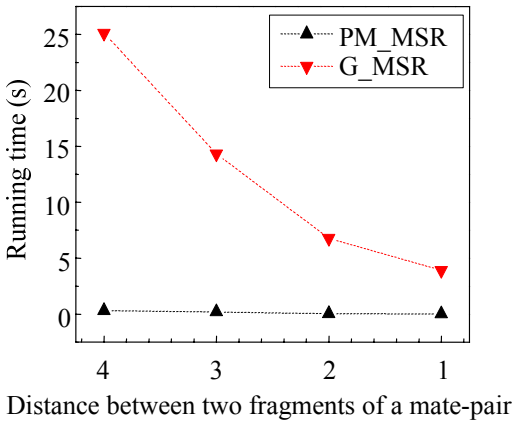


图 4-6 Mate-pair 的间距较小时算法的运行时间对比

图 4-6 是当 SNP 位点数为 600，片段覆盖度为 20 (普通片段的覆盖度为 6、mate-pair 的覆盖度为 14)、mate-pair 中两片段间的间距 d 从 4 减少到 1 时，两算法的对比。当 d 为 4 时，G_MSR 和 PM_MSR 算法的运行时间分别是 25.12 和 0.3 秒; 当 d 为 1 时，两算法的运行时间分别为 3.91 和 0.02 秒。从实验结果可以看出，当片段覆盖度保持不变时，d 减少时 G_MSR 算法和 PM_MSR 算法的时间都会急剧减少，这是因为 d 减少时，片段中的洞的最大个数 k 会随着减少，同时覆盖某位点的片段中在该位点取空值的概率也会着随着变小，即 h 的值也会跟着变小。极端的情况下，片段中没有洞，那么 k 为 0，h 也必定为 0，这时即使 $k_1=n$，$k_2=m$，根据算法复杂度分析，PM_MSR 算法也不会比 G_MSR 性能明显差。

当 mate-pair 的两片段的间距为 100 时，这时片段中的洞的最大数 k 可达到 100，G_MSR 已经无法运行了; 而 PM_MSR 算法则可以有效地运行。图 4-7 是当 SNP 位点数为 1 000、mate-pair 中两片段间的间距 d 为 100、普通片段的覆盖度为 6、mate-pair 的覆盖度从 6 增长到 14 时(即总的片段覆盖度从 12 增长到 20)，PM_MSR 算法的运行时间曲线。当 mate-pair 的覆盖度为 6 时， PM_MSR 算法的运行时间为 0.284 秒; 当 mate-pair 的覆盖度增大到 14 时，运行时间增大为 14 414.125 秒，约 4 小时。当 mate-pair 的覆盖度增加到 16 时，我们对 PM_MSR 进行了一次测试，时间为 231 913 秒，约 65 小时。从实验可以看出，当 mate-pair

的两片段的间距很大时 (k 也会很大) 而片段覆盖度不大时，PM_MSR 算法能够有效运行; 但 h 会随着 mate-pair 的覆盖度的增加而增加，从而导致 PM_MSR 算法的时间指数上升。幸运的是，为了减少测序的时间和金钱的代价，实验室进行测序时的片段覆盖度是不太大的。

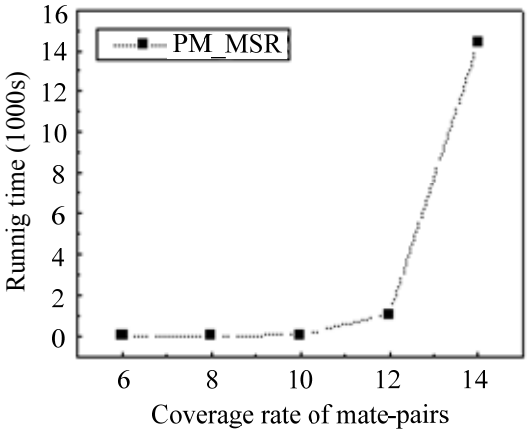


图 4-7 Mate-pair 两片段的间距为 100 个 SNP 位点时 PM_MSR 算法的运行时间

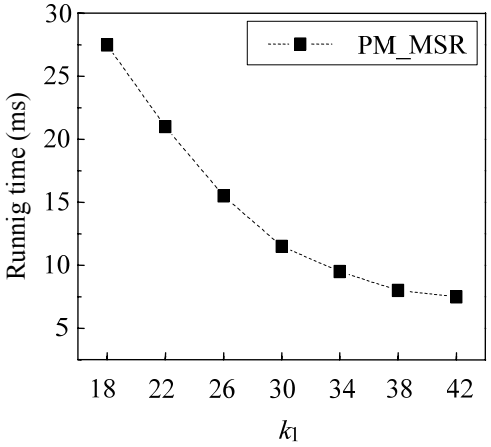


图 4-8 片段最大长度 k_1 变化时 PM_MSR 算法的运行时间

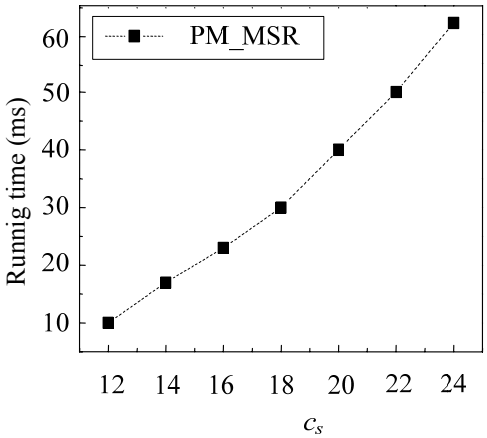


图 4-9 普通片段覆盖度 c_s 变化时 PM_MSR 算法的运行时间

图 4-8 是当 SNP 位点数为 600、片段覆盖度为 14 (普通片段的覆盖度为 8、mate-pair 的覆盖度为 6)、mate-pair 中两片段间的间距 d 为 10 时，通过控制普通片段的长度改变 k_1 的值 (为 mate-pair 覆盖的 SNP 位点数) 来测试 PM_MSR 算法的运行性能。当 k_1 为 18 时，PM_MSR 算法的运行时间为 0.0275 秒;随着 k_1 增加算法的时间逐渐减少，当 k_1 增大为 42 时，算法的运行时间减少到 0.007

5 秒。这是因为在 SNP 位点数和片段覆盖度保持不变的情况下，片段越长，片段数越少; 又因为构成 mate-pair 的两片段中洞的个数保持不变 (为 d)，这样，所有片段中洞所占的比例就会下降，覆盖同一 SNP 位点的片段中在该位点取空值的片段数的比例就会下降，进而导致 h 下降。基于以上两个因素，算法的运行时间将会减少。

图 4-9 是当 SNP 位点数为 600、mate-pair 中两片段间的间距 d 为 10、mate-pair 的覆盖度为 c_m=6，通过控制普通片段的覆盖度 c_s 来改变 k_2 的值来测试 PM_MSR 算法的运行性能。当 c_s 为 12 时，PM_MSR 算法的运行时间为 0.01 秒; 当 c_s 增大为 24 时，算法的运行时间增大为 0.062 秒。由于 c_s 增大，k_2 会随着增大，从图 4-9 可以看出，PM_MSR 算法运行时间基本上与 k_2 成线性增长关系，这与算法复杂度的理论分析结果是吻合的。

4.3 有长 Mate-Pair 时 MFR 参数化算法

4.3.1 有长 Mate-Pair 片段时 MFR 参数化模型

4.2.1 小节曾提到在有 mate-pair 片段数据的情况下，虽然 mate-pair 片段中洞的最大数可达 100 个，但是，因为测序的成本和所需的时间与覆盖度成正比，所以，实验室测序时片段的覆盖度(coverage)是不太大的。Huson 等[142]对 Celera 公司的人类基因组测序工程中的片段数据[138]进行了详细的分析，从其片段覆盖图上可看出绝大部分位点上的片段数为 5 左右，最大值没有超过 19。另外尽管 SNP 分布不均匀，从已有的数据[47, 140, 141]来看，长为 150kb 的 mate-pair，其覆盖的 SNP 位点数约为 150 个左右。这样即使片段数据中有较长的 mate-pair，片段覆盖的 SNP 最大位点数 k_1 不会很大，覆盖任意 SNP 位点的最大片段数 k_2 仍然较小。按照第三章提出的方法，我们仍然可以用(k_1, k_2)参数化条件对有长 mate-pair 片段数据的 MFR 问题进行参数化。

其中(k_1, k_2)参数化条件定义为片段覆盖的SNP位点数不超过k_1，覆盖任意SNP位点的片段数不超过k_2。图4-2所示的SNP矩阵满足(9, 6)参数化条件。

问题 4.1 PM_MFR (parameterized minimum fragment removal with mate-pair): 给定一个满足(k_1, k_2)参数化条件的SNP矩阵**M**（允许有空隙），PM_MFR问题是要求删除最少的行（即片段），即保留最多的行（即片段）使**M**可行。

本章下文中，一个SNP矩阵**M**的PM_MFR解指的是使**M**可行能保留下的最多的行数，用$MFR_{PM}(\mathbf{M})$来表示。

4.3.2　预处理

SNP 矩阵 **M** 仍然采用如下数据结构：对每一行 i，记下该行覆盖的最左边和最右边的列号，即 $left(i)$和 $right(i)$，记录该行从列 $left(i)$到列 $right(i)$的各列上的值。我们先对 SNP 矩阵 **M** 进行预处理，预处理的方法与 3.3.1 基本相同，即：

Step 1　对 SNP 中的行按其 *left* 值进行非降序排列，然后对每一列 j，计算出覆盖该列的行的序号的有序集，记作 $rowset(j)$。

Step 2　去掉没有出现‘0’或没有出现‘1’冗余列，修改受影响行的 *left* 和 *right* 函数值。

Step 3　去掉空的冗余行，修改受影响的列的 *rowset* 集。

预处理时间复杂度为 $O(mlogm+nk_2+mk_1)$。

根据一个定理 3.1 可知满足(k_1, k_2)参数化条件的 SNP 矩阵 **M** 经过以上预处理后仍然满足(k_1, k_2)参数化条件。

定理 4.4　令一个 SNP 矩阵 **M** 经过以上的预处理得到的 SNP 矩阵为 **M'**，其中去掉的行的集合为 **X**，则 **M** 的 PM_MFR 的解（这里指能保留的最大行数）等于 **M'**的对应解与 **X** 的行数之和。

用证明定理 4.2 的方法可以证明定理 4.4，在此不再赘述。

从定理 4.4 可知，求解一个 SNP 矩阵的 PM_MFR 的解可以归结为求其预处理后形成的矩阵的对应解。

为了叙述的简便，我们称一个经过上述预处理后的 SNP 矩阵为精简有序的 SNP 矩阵。本节下面讨论的 SNP 矩阵 **M**，除特别声明外，均指精简有序的 SNP 矩阵，该矩阵具有如下特点：行列数 m、$n > 0$；任何一个列总有一些行的值是‘0’，还有一些行的值为‘1’；对于行 $i < j$, 有 $left(i) \le left(j)$。

4.3.3　PM_MFR 参数化算法

对于一个 SNP 矩阵 **M**，如果保留 **M** 的一些行的集合 S 能使 **M** 可行，则 S 中的行又可以进一步分为两个子集 H_1 和 H_2，使得同一个子集中的行互相兼容。这样 **M** 中的覆盖某一列 j 的所有行可以分成三部分：删除的行属于第 1 部分，分在 H_1 中的行属于第 2 部分，分在 H_2 中的行属于第 3 部分。

定义 4.5 分组函数：分组函数 G 定义为从一个有序行集 R 到 $\{0, 1, 2\}$的映射，即 $G: R \to \{0, 1, 2\}$。

对于一个有h行的有序集合R， 分组函数G可以用一个h位的三进制数来编码，使得R中的第i行的分组函数值$G(i)$等于这个三进制数的第i位的值。为了使空集上的分组函数有编码，我们规定空集上的分组函数编码为−1。

为了叙述简洁，一个定义在$rowset(j)$ (覆盖列j的有序行集) 叫作在列j上的分组函数。对一个满足(k_1, k_2)参数化条件的SNP矩阵，在列j上最多有3^{k_2}不同的分组函数。

对于**M**中的列j，$rowset(j)$中的行可以按照其分组函数的值可分成三组H_0、H_1和H_2，其中H_0表示要删除的行集，H_1和H_2表示保留下来的行可以进一步分为两个子集。给定列j上的一个分组函数G，令$j' \le j$，如果存在两行 i_1和i_2，有$G(i_1) = G(i_2) \neq 0$且这两行在列j'上冲突，则称G在列j'上不可行，否则称G在列j'上可行。如果不存在列$j' \le j$使得G在列j'上不可行，则G是直到列j可行的。

对于任意列j，容易验证列j上至少存在着一个分组函数是直到列j可行的（考虑如下定义的分组函数G：对于任意i，$G(i) = 0$）。

令G是定义在行集R上的一个分组函数，R'是R的一个子集，G'是定义在行集R'上的一个分组函数，如果对于R'中的任意行的G'函数值均等于它的G函数值，那么称G'是G在R'上的投影，而G是G'在R上的一个扩展。

定义 4.6 $KR[j, G]$: 给定列j和$rowset(j)$上直到列j可行的分组函数G, $KR[j, G]$定义为满足下列条件的**M**的行的最大子集：

(1) 对于$rowset(j)$中的行i，$i \in KR[j, G]$当且仅当$G(i) \neq 0$;

(2) 对于属于$KR[j, G]$而不属于$rowset(j)$中的行i，行i至少要覆盖列j前的某一列；

(3) 对$KR[j, G]$中的所有行存在着一个划分, 使得划分在同一子集中的行在列1到列j的所有列上均不冲突, 且对于行$i_1, i_2 \in KR[j, G] \cap rowset(j)$，行$i_1, i_2$被划分到同一个子集中当且仅当$G(i_1) = G(i_2)$。

令$K[j, G] = |KR[j, G]|$，即$KR[j, G]$中的行数。

对于$rowset(j)$上的其他分组函数（即不是直到列j可行的）G，规定$KR[j, G] = \varnothing$，$K[j, G] = -1$。

对于一个$m \times n$ SNP矩阵**M**和列n上的分组函数G，如果G不是直到列n可行的，则显然$\mathrm{MFR}_{\mathrm{PM}}(\mathbf{M}) \ge K[n, G]$；否则保留$KR[j, G]$中的行必定能使**M**可行，也有$\mathrm{MFR}_{\mathrm{PM}}(\mathbf{M}) \ge K[n, G]$。

反过来，假设R_K是对应着**M**的PM_MFR问题的一个解，即R_k是使**M**可行能保留下来的行的最大集合。按照定义，必存在一个划分F把R_k中的行分成两个子集H_1和H_2，使得被划分在同一个子集中的行在列1到列j的所有列上均不冲突。这样一个定义在**M**的所有行上的分组函数G'可以通过如下方法构造出来： 对于任意行i，如果$i \notin R_K$，则$G'(i) = 0$；如果$i \in H_1$，则$G'(i) = 1$；如果$i \in H_2$，则$G'(i) = 2$。令G'在$rowset(n)$上的投影是G，易知G直到列n是可行的。同样容易验证R_K满足$KR[n, G]$的定义（即定义4.6）的3个条件：由分组函数的构造方法可知条件(1)满足；由于**M**经过了预处理，所有行肯定覆盖某一列，所以条件(2)也满足；同样划分F和G满足条件(3)。这样列n上必定存在一个直到列n可行的分组函数，使得$K[n, G] \geq \mathrm{MFR}_{\mathrm{PM}}(\mathbf{M})$。

由此下面的公式成立：

$$\mathrm{MFR}_{\mathrm{PM}}(\mathbf{M}) = \max\ (\ K[n,G] \mid G\text{是列}n\text{上的分组函数}\ ) \qquad (4\text{-}4)$$

下面考虑**M**的第1列上的分组函数G。如果G在该列上不可行，则：

$$KR[1,G] = \varnothing, \quad K[1,G] = -1. \qquad (4\text{-}5)$$

否则，$R_K = \{i \mid i \in rowset(1) \wedge G(i) \neq 0\}$是满足定义4.6中所有条件的最大集合：有条件(1)和(2)可知，属于$KR[1,G]$中的行必属于R_K，对R_K中的行按G函数的值进行划分满足条件(3)。因此，有下列公式成立：

$$KR[1,\ G] = \{\ i \mid i \in rowset(1) \wedge G(i) \neq 0\ \} \qquad (4\text{-}6)$$

$$K[1,\ G] = |\ KR[1,G]\ | \qquad (4\text{-}7)$$

为了算法的进一步阐述，我们需要把上述概念从一列扩展到两列。令既覆盖列j_1又覆盖列j_2的所有行的集合为$R_C(j_1, j_2)$。

定义 4.7 $BR[j, G']$：给定列j和$R_C(j, j+1)$上直到列j可行的分组函数G', $BR[j, G']$定义为满足下列条件的**M**的行的最大子集：

(1) 对于$R_C(j, j+1)$中的行i，$i \in KR[j, G]$当且仅当$G(i) \neq 0$;

(2) 对于属于$BR[j, G']$而不属于$R_C(j, j+1)$中的行i，行i至少要覆盖列j前的某一列；

(3) 对$BR[j, G']$中的所有行存在着一个划分，使得划分在同一子集中的行在列1到列j的所有列上均不冲突，且对于行$i_1, i_2 \in BR[j, G'] \cap R_C(j, j+1)$，行$i_1, i_2$被划分到同一个子集中当且仅当$G'(i_1) = G'(i_2)$。

令$B[j,G'] = |\ BR[j, G']\ |$，即$BR[j, G']$中的行数。

对于$R_C(j, j+1)$上的其他分组函数（即不是直到列j可行的）G'，规定$BR[j, G'] = \varnothing$，$B[j,G'] = -1$。

这样我们有了下面的定理。

定理 4.5 令G'是$R_C(j, j+1)$上的一个分组函数，下面的公式成立：

$$B[j, G'] = \max\{ K[j, G] \mid G是G'在rowset(j)上的一个扩展 \}；\quad (4\text{-}8)$$

$$BK[j, G'] = KR[j, G]，其中G是G'在rowset(j)上使K[j, G]最大的一个扩展。\quad (4\text{-}9)$$

证明：如果G'不是直到列j可行的，则它在$rowset(j)$上的任意扩展G都不是直到列j可行的，这时上述公式显然成立。

下面假设G'是直到列j可行的。注意$R_C(j, j+1)$是$rowset(j)$一个子集。

令G是G'在$rowset(j)$上的一个扩展，如果G不是直到列j可行的，则显然有$K[j, G] \le B[j, G']$；否则，$KR[j, G]$满足$BR[j, G']$定义（即定义4.7）的3个条件，这样也有$K[j, G] \le B[j, G']$。

另一方面，根据定义4.7，必存在一个划分F把$BR[j, G']$中的行分成两个子集H_1和H_2，使得被划分在同一个子集中的行在列1到列j的所有列上均不冲突，且对于行$i \in BR[j, G'] \cap R_C(j, j+1)$，行$i$被划分到子集$H_1$（或$H_2$）中当且仅当$G'(i) = 1$（或2）。

这样G'在$rowset(j)$上的一个扩展G可以通过如下方法构造出来：对于任意行$i \in rowset(j)$，如果$i \in R_C(j, j+1)$，则$G(i) = G'(i)$；如果$i \notin BR[j, G']$，则$G(i) = 0$；最后，如果$i \in BR[j, G'] - R_C(j, j+1)$，则$G(i) = q$ 当且仅当$i \in Hq$ (q=1,2)。容易验证$BR[j, G']$满足$KR[j, G]$定义（即定义4.6）的3个条件，所以有$K[j, G] \ge B[j, G']$。

定理得证。　□

定理 4.6 对于列$j > 1$，令G是$rowset(j)$上的在列j上可行的一个分组函数，G'是G在$R_C(j-1, j)$上的投影。如果$B[j-1, G'] = -1$，则

$$KR[j, G] = \varnothing,\quad K[j, G] = -1; \quad (4\text{-}10)$$

否则

$$KR[j, G] = BR[j-1, G'] \cup \{ i \mid i \in rowset(j) \wedge i \notin R_C(j-1, j) \wedge G(i) \neq 0 \} \quad (4\text{-}11)$$

$$K[j, G] = B[j-1, G'] + |\{ i \mid i \in rowset(j) \wedge i \notin R_C(j-1, j) \wedge G(i) \neq 0 \}| \quad (4\text{-}12)$$

证明：如果$B[j-1, G'] = -1$，则G'不是直到列$j-1$可行的，这就意味着G'的任意扩展也不是是直到列j可行的，因此，公式(4-10)成立。

当$B[j-1, G'] \neq -1$时，根据定义4.7，必存在一个划分F把$B[j-1, G']$中的行分成

两个子集H_1和H_2，使得被划分在同一个子集中的行在列1到列$j-1$的任意列上均不冲突，且对于行$i \in BR[j-1, G'] \cap R_C(j-1, j)$，行$i$被划分到子集$H_1$（或$H_2$）中当且仅当$G'(i) = 1$（或2）。$BR[j-1, G']$中的任意行$i$必须覆盖列$j$前的某一列，因此如果$i$不在$R_C(j-1, j)$中，则该行肯定不会覆盖列$j$，因而也不会和其他行在列$j$上冲突。根据投影的定义，对于两行$i_1, i_2 \in BR[j-1, G'] \cap R_C(j-1, j)$，如果行$i_1, i_2$被划分到同一个子集中，则必有$G(i_1) = G(i_2)$；$G$在列$j$上可行说明行$i_1, i_2$在列$j$上不冲突，这样行$i_1, i_2$在列1到列$j$的所有列上均不冲突。由此可知$BR[j-1, G']$中被划分到同一子集的行在列1到列$j$的所有列上均不冲突。

令$X = \{ i \mid i \in rowset(j) \wedge i \notin R_C(j-1, j) \wedge G(i) \neq 0 \}$。

X中的行覆盖列j但不覆盖列$j-1$，隐含着X中的行i不会覆盖列j前面的任何列，因此行i不会属于$BR[j-1, G']$，且行i不会和$BR[j-1, G']$中的行在列1到列$j-1$的任意列上冲突。由于G在列j上可行，对于任何行$i' \in BR[j-1, G'] \cap R_C(j-1, j)$，如果$G(i')=G(i)=1$（或2），则行$i$与行$i'$在列1到列$j$的任意列上冲突。因此，如果$G(i) = 1$（或2），则行$i$可加入到$H_1$（或$H_2$）中，这样行$i$就不会和同一个子集中的其他行在列1到列$j$的任意列上冲突。这样$G$是直到列$j$可行的，且$BR[j-1, G']$和$X$的并集满足$KR[j, G]$定义（即定义4.6）的条件(1)和(2)，同时对该并集的上述划分满足条件(3)。因此，有$K[j, G] \geq B[j-1, G'] + |X|$。

另一方面，假定已知$K[j, G]$和$KR[j, G]$，根据定义4.6，必存在一个划分F把$KR[j, G']$中的行分成两个子集H_1和H_2，使定义4.6的条件(3)得以满足。由定义4.6的条件(1)和(2)可知，$KR[j, G]$中所有不覆盖列j前任意列的行都在X中。令G'是G在$R_C(j-1, j)$上的投影。容易验证行集$KR[j, G]-X$满足$BR[j-1, G']$定义（即定义4.7）的条件(1)和(2)，同时F满足条件(3)。因此，有$B[j-1, G'] \geq K[j, G] - |X|$。

定理得证。　□

根据上面的讨论，我们得到了下面的PM_MFR算法。

PM_MFR 算法：

Input: 一个精简有序的 $m \times n$ SNP 矩阵 **M**，对所有的 j: $1 \leq j \leq n$ 的 $rowset(j)$

Output: M 的 MFR 解

Setp 1. 初始化

Step 1.1. **for** $rowset(1)$上的每一个分组函数G **do**

Step 1.1.1. 根据公式(4-5) ~ (4-7)计算 $K[1,G]$和 $KR[1,G]$;

Step 1.2. $j = 1$;

Step 2. **while** $j < n$ do

Step 2.1. 计算 $R_C(j, j+1)$;

Step 2.2. **for** $R_C(j, j+1)$上的每一个分组函数 G' **do**

Step 2.2.1. 根据公式(4-8)、(4-9)计算 $B[j, G']$和 $BR[j, G']$;

Step 2.3. $j = j+1$;

Step 2.4. **for** *rowset*(j)上的每一个分组函数 G **do**

Step 2.4.1. **if** G在列j上不可行 **then**

$KR[j, G]= \varnothing$, $K[j, G] = -1$; //公式(4-10)

Step 2.4.2. **else** 根据公式(4-11)和(4-12)计算$K[j,G]$和$KR[j,G]$;

Step 3. //根据公式(4-4)，最大的 $K[n, G]$就是 **M** 的 MFR 问题的解

Step 3.1. MAX=−1;

Step 3.2. **for** *rowset*(n)上的每一个分组函数 G **do**

Step 3.2.1. **if** MAX< $K[n, G]$ **then** MAX = $K[n, G]$;

Step 4. return MAX;

定理 4.7 对于一个(k_1, k_2)参数化条件的精简有序的 $m \times n$ SNP 矩阵 **M**，算法 PM_MFR 能正确地求出其 PM_MFR 问题的一个解; 加上预处理该算法的时间复杂度为 $O(nk_2 3^{k_2}+mlogm+nk_2+mk_1)$，空间复杂度为 $O(mk_1+nk_2+m3^{k_2})$。

证明：算法 PM_MFR 是基于公式(4-4) ~ (4-12)，而这些公式在引入的时候被得到证明。

下面讨论算法的时间复杂度。

在 Step 1.1.1, 对每一个分组函数 G 和每一行 $i \in$ *rowset*(1), 考虑以下两种情况：

(1) 行 i 满足条件 $\mathbf{M}_{i,1} \neq$ '−' $\wedge$ $G(i) = 1$: 如果行 i 是满足上述条件的第一行，那么用 c_1 表示 $\mathbf{M}_{i,1}$，用 r_1 表示行 i; 否则，检查 $\mathbf{M}_{i,1}$ 是否和 c_1 相等，如果 $\mathbf{M}_{i,1} \neq c_1$，那么行 i 和 r_1 冲突，G 在列 1 上不可行，这时采用公式(4-5)。

(2) 行 i 满足条件 $\mathbf{M}_{i,1} \neq$ '−' $\wedge$ $G(i) = 2$: 如果行 i 是满足上述条件的第一行，那么用 c_2 表示 $\mathbf{M}_{i,1}$，用 r_2 表示行 i; 否则，检查 $\mathbf{M}_{i,1}$ 是否和 c_2 相等，如果 $\mathbf{M}_{i,1} \neq c_2$，那么行 i 和 r_2 冲突，G 在列 1 上不可行，这时采用公式(4-5)。

在检查完 *rowset*(1)中的所有的行，如果公式(4-5)没有采用，则易知 G 在列 1 上可行，这时采用公式(4-6)和(4-7)。

因为 **M** 满足(k_1, k_2)参数化条件，所以 *rowset*(1)中的行数不回超过 k_2，*rowset*(1)上的分组函数个数不超过 3^{k_2}。这样 Step 1 需时间 $O(k_2 3^{k_2})$。

Step 2循环$n-1$次。同时扫描*rowset*(*j*)和*rowset*(*j*+1)可得到$R_C(j, j+1)$，这样 Step 2.1 takes time $O(nk_2)$。得到*G*'在*rowset*(*j*)上的一个扩展需时间$O(k_2)$，令$R_C(j, j+1)$中的行数为h_c，则*G*'在*rowset*(*j*)上的所有不同的扩展个数为$3^{k_2-h_c}$，因此Step 2.2需时间$O(nk_2 3^{h_c} 3^{k_2-h_c}) = O(nk_2 3^{k_2})$。

Step 2.3需时间$O(n)$，Step 2.4最多循环$n3^{k_2}$次。与Step 1.1.1相同，可以通过扫描*rowset*(*j*)，检查*rowset*(*j*)中的每一行的*G*函数值确定*G*在列*j*上是否可行，需时间$O(k_2)$。根据公式(4-10) ~ (4-12)进行计算需时间$O(k_2)$。因此，Step 2.4需时间$O(nk_2 3^{k_2})$。这样，Step 2需时间$O(nk_2 3^{k_2})$。Step 3需时间$O(3^{k_2})$。总之，包括预处理，整个算法的时间复杂度为$O(nk_2 3^{k_2}+m\log m+mk_1)$。

算法的空间复杂度：*rowset*和R_C所需的空间为$O(nk_2)$，*KR*和*BR*所需的空间为$O(m3^{k_2})$，*K*和*B*所需的空间为$O(3^{k_2})$，因此，包含SNP矩阵**M**所需的空间$O(mk_1)$，整个算法的空间复杂度为$O(mk_1+nk_2+m3^{k_2})$。定理得证。 □

4.3.4. 实验结果

我们对 PM_MFR 算法和 Bafna 等[62]提出的求解有空隙 MFR 的算法的性能进行对比测试。

我们用 C++语言实现了 PM_MFR 算法，通过对 Fast hare 源程序[134]中的对应程序的修改实现了 Bafna 等[62]的对应算法，该算法记作 G_MFR。作为精确算法，PM_MFR 和 G_MFR 在保留的行数上，结果是完全相等的，尽管在保留的具体行上可能存在随机性，算法的单体型重构精度不完全耦合，但总体上没有优劣之分。因此下面只进行算法运行时间对比测试。

测试平台与前一章的完全相同。测试用的输入数据的产生方法与参数与 4.2.4 节完全相同。生成参数如下：两单体型的差异率为 10%，测序误差 e=5%，普通片段植入空值得概率 p=1%，普通片段的长度在 3 和 6 SNP 位点之间随机变化，mate-pair 长为 $6\times2+d$。详细情况请参照本章 4.2.4 节及文献[134]和[145]。

同 4.2.4 节，参数 d 和单体型长度即 SNP 位点数 n，在一个区间内变化，普通片段数 m_1 和 mate-pair 片段数 m_2 的大小则按照下面的公式由普通片段的覆盖度和 mate-pair 的覆盖度确定：片段数=单体型长度×覆盖度/片段平均长度。

下面的实验结果中，图的每一个点均为 100 次重复测试的平均值。

当片段覆盖度为 10 (普通片段的覆盖度为 6、mate-pair 的覆盖度为 4)、

mate-pair 中洞的个数 d 为 4 个 SNP 位点时，图 4-10 显示 G_MFR 的运行时间随 SNP 矩阵列数 n 的增大而迅速上升。当 n=30, G_MFR 和 PM_MFR 算法的运行时间分别是 7.3798 秒和 0.0204 秒；当 n 增大到 90 时, G_MFR 和 PM_MFR 算法的运行时间分别上升到 513.433 秒和 0.328 秒。这是因为当 n 增加，覆盖度不变时，片段数 m 也随着增加，在算法复杂度分析上，G_MFR 运行的时间的增长速度应该是 n 增长速度的 3 次方，而 PM_MFR 的时间则基本成线性增长。

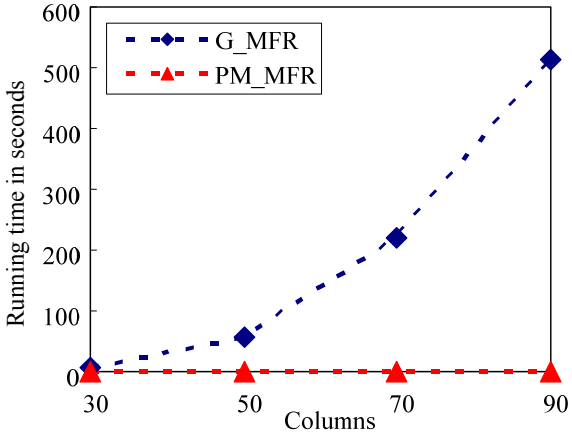


图 4-10　SNP 矩阵的列数变化时算法的运行时间对比

图 4-11 则是当片段覆盖度为 10 (普通片段的覆盖度为 6、mate-pair 的覆盖度为 4)、n = 50、mate-pair 中洞的个数 d 发生变化时，两算法的运行时间曲线。当 d 为 0 时, G_MFR 和 PM_MFR 算法的运行时间分别是 0.0956 和 0.0592 秒；当 d 增加到 5 时, G_MFR 和 PM_MFR 的运行时间分别上升到 299.954 和 0.083 秒。这次实验的结果说明, G_MFR 对 d 的大小非常敏感，当 d 增加时, k 随着增加,其运行时间急剧上升；但 PM_MFR 的时间变化却不大，对 k 不敏感。

当SNP位点数为50、mate-pair中洞的个数d为4，图4-12显示片段的覆盖度c从4增大到16（mate-pair的覆盖度c_p和普通片段的覆盖度c_s均为c/2）时，G_MFR和PM_MFR算法的运行时间。当c=2时，G_MFR和PM_MFR算法的运行时间分别是0.735和0.001秒;当c增大到16时，两算法的运行时间分别增大到184.03和35.44秒。这个实验说明了PM_MFR算法对片段的覆盖度敏感，这是因为覆盖度上升时，覆盖同一SNP位点的片段数就会增加，导致(k_1,k_2)的第2个参数k_2上升，PM_MFR算法的时间呈指数增长。

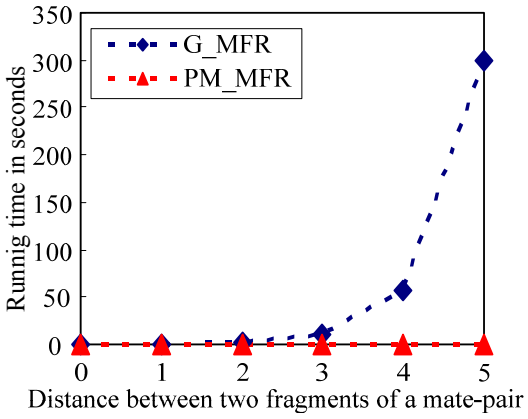


图 4-11　当 mate-pair 两片段的间距发生变化时算法的运行时间对比

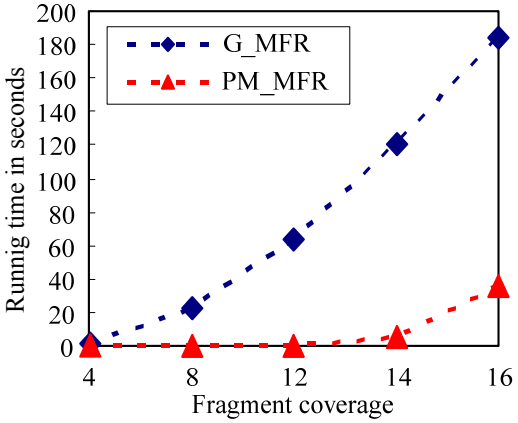


图 4-12　片段覆盖度发生变化时算法的运行时间对比

表 4-1　当$n=200$、$k_2=16$ ($c_p=8, c_s=8$) 时，G_MFR和PM_MFR算法的复杂度比较*

	k	4	6	8	10	20	100
Time Complexity	G_MFR $O(nm^2 2^{2k}+2^{3k}m^3)$	3.98E+11	2.32E+13	1.39E+15	8.44E+16	7.69E+25	1.03E+98
	PM_MFR $O(nk_2 3^{k_2}+mlogm+nk_2+mk_1)$	1.38E+11	1.38E+11	1.38E+11	1.38E+11	1.38E+11	1.38E+11
Space complexity	G_MFR $O(2^{2k}m^3)$	2.42E+10	3.60E+11	5.42E+12	8.24E+13	7.33E+19	8.13E+67
	PM_MFR $O(mk_1+nk_2+m3^{k_2})$	1.96E+10	1.91E+10	1.87E+10	1.84E+10	1.75E+10	1.59E+10

*片段的最大长度k_1等于mate-pair的长度$l_1=2\times6+k$, 普通片段的平均长度为$l_2=(3+6)/2$, 片段数$m=n\times c_p/l_1+n\times c_s/l_2$.

当片段中的最大洞数k大于6时，G_MFR因所需资源过大，无法在测试服务器上有效运行。基于G_MFR算法[62]和PM_MFR算法（参考本章定理 4.7）的复杂度理论分析，表4-1给出了当$n=200$、$k_2=16$ ($c_p=8$, $c_s=8$)时，G_MFR和PM_MFR算法复杂度如何随k而变化。

从表4-1可以看出，当片段数据中有mate-pair，片段中最大洞的个数不是很小时，PM_MFR比G_MFR在运算效率上有明显优势。

4.4　本章小结

为了提高DNA组装的精度，在全基因组测序中常常采用mate-pair。典型的

mate-pair长度有2kb、10kb、50kb和150kb[142]，按1kb DNA片段中有1个SNP位点[1, 8, 137]估计，典型的mate-pair片段中洞的个数是2、10、50和150个。这样，当测序数据中有长为50kb以上的mate-pair片段时，片段中洞的最大个数k可达到50以上，这时复杂度以片段中洞的最大个数k为2的指数的算法已不再实用。由于DNA测序成本和时间的限制，基因组的测序数据中的片段覆盖度不会很大，因而覆盖同一个SNP位点的片段数是有限的(约5左右，通常不大于19[1, 138, 142])，令其最大值为k_2，那么，其中覆盖某个SNP位点而在该位点上取空值的最大片段数h不会比k_2大，另外单一片段覆盖的最大SNP位点数k_1一般小于的单体型的SNP位点总数n。根据上述事实，针对长的mate-pair中洞的个数较多的情况，本章设计了求解MSR和MFR时间复杂度分别为$O(nk_1k_2 2^{2h}+k_1 2^h+nk_2+mk_1)$和$O(nk_2 3^{k_2}+m\log m+nk_2+mk_1)$、空间复杂度分别为$O(k_1 n2^h+nk_2+mk_1)$和$O(mk_1+nk_2+m3^{k_2})$的的参数化精确算法PM_MSR和PM_MFR， 其中为m为DNA片段数。理论分析和实验结果均表明PM_MSR和PM_MFR算法所需的时间与片段中洞的个数的最大值k没有直接的关系，在片段数据中存在长mate-pair的情况下仍然能有效计算, 具有很好的可扩展性和实用价值。

第五章 MEC 与 MEC/GI 参数化模型和算法

5.1 引言

最少错误更正（Minimum Error Correction, MEC）是单体型的组装问题中的一个重要计算模型[146]，该问题最先由 Lippert 等[81]提出了，指的是修改 DNA 片段上最少的 SNP 值, 使得修改后的 DNA 片段可以分成两个子集，使得每个子集的片段能决定一个单体型。MEC 模型也叫 MLF (Minimum Letter Flips)模型[82]。

对于一个 SNP 矩阵 **M**，MEC 模型的定义如下：

MEC (Minimum Error Correction)**模型**：给定一个 SNP 矩阵 **M**，把 **M** 中的某些元素的值从'0'翻转成'1'，或从'1'翻转成'0'，在翻转元素最少的条件下使 **M** 可行。

Lippert 等[81]在提出该模型时，同时证明了对于一般的 SNP 矩阵，MEC 模型是 NP-难。Cilibrasi 等[129]则进一步证明了，即使对于无空隙的 SNP 矩阵，MEC 模型也是 NP-难，并且证明了当片段中有空隙时，MEC 模型也是 APX-难的。

在 MEC 模型的基础上，附加其他生物信息，又出现了一些新的模型。

由于个体的基因型比较容易测定，为了提高模型的单体型重构精度，Wang 等[68]对MEC进行改进，加入个体的基因型信息，引入了MEC/GI (MEC with Genotype Information) 计算模型：

MEC/GI 模型：给定一个 SNP 矩阵 **M** 和基因型 G，翻转 **M** 中最少的元素（0 变成 1，或 1 变成 0），目标是使得翻转后的 SNP 矩阵能由一对构成 G 的单体型导出。

其中一个 SNP 矩阵 **M** 可由一对单体型导出指的是：**M** 中的任意行总是可以与其中的一个单体型兼容。在 2.3 节曾提到，一个 SNP 矩阵 **M** 是可行的当且仅当可以找到一对单体型，使得 **M** 可由这对单体型导出。

为求出上面这两个模型的精确解，Wang 等[68]曾设计了时间复杂度为 $O(2^m)$ 的分支限界算法，其中 m 为 SNP 矩阵的行数。由于这些算法是指数时间复杂度

的，只适合片段数不多的场合。对于片段数较多的情况，Wang 等[68]设计了遗传算法求其近似解。遗传算法的时空复杂度要优于分支限界算法，但其单体型重构精度要低于分支限界算法。

就我们所知，目前还没有其他实际有效的精确算法，本章根据基因测序片段数据的特点，对 MEC 和 MEC/GI 进行参数化，进而提出有效的参数化算法求解它们的精确解。

5.2 MEC 和 MEC/GI 参数化模型

对于 DNA 测序片段数据，第三章曾提到在通常情况下，与 DNA 片段总数(m)相比，覆盖一个 SNP 位点的片段数（通常不大于 19）是一个很小的数。即使在有 mate-pair 片段数据的情况下，一个片段覆盖的 SNP 位点数也不会很大。这样片段覆盖的 SNP 最大位点数 k_1 不会很大，覆盖任意 SNP 位点的最大片段数 k_2 仍然较小，因此我们仍然采用用第三章提出的(k_1, k_2)参数化条件对 MEC 和 MEC/GI 模型进行参数化。其中，(k_1, k_2)参数化条件定义为片段覆盖的 SNP 位点数不超过 k_1，覆盖任意 SNP 位点的片段数不超过 k_2。

问题 5.1 P_MEC (Parameterized Minimum Error Correction)：给定一个满足(k_1, k_2)参数化条件的 SNP 矩阵 **M**，P_MEC 问题是要求翻转（'0'变成'1'，或'1'变成'0'）**M** 中的最少元素使 **M** 可行。

问题 5.2 P_MEC/GI (Parameterized Minimum Error Correction with Genotype Information)：给定一个满足(k_1, k_2)参数化条件的 SNP 矩阵 **M** 和基因型 G，P_MEC/GI 问题是翻转 **M** 中最少的元素（0 变成 1，或 1 变成 0），使得翻转后的 SNP 矩阵能由一对构成 G 的单体型导出。

一个 SNP 矩阵 **M** 能由一对构成基因型 G 的单体型导出，我们就说 **M** 和 G 兼容。

本章中，一个 SNP 矩阵 **M** 的 P_MEC 解指的是使 **M** 可行需翻转的最少元素个数，用 P_MEC(**M**)来表示；一个 SNP 矩阵 **M** 的 P_MEC/GI 解指为使 **M** 能和基因型 G 兼容所需翻转的最少元素个数，用 P_MEC/GI (**M**,G)来表示。

5.3 MEC 和 MEC/GI 参数化算法

首先我们引入相关的一些定义。

定义 5.1 二分函数：二分函数 P 是定义在某个行集 R 上的一个映射：$R\rightarrow\{0, 1\}$, 即 P 把 R 中的行映射到 0 或 1。

显然利用二分函数 P，R 中的行可划分到两个子集中。

为了叙述的简便，令覆盖列 j 的行的有序集 $rowset(j)$上的二分函数被称为列 j 上的二分函数。

显然对于一个满足(k_1,k_2)参数化条件的 SNP 矩阵 M，覆盖列 j 的行数不会大于 k_2，那么在列 j 上的可能的二分函数最多不超过 2^{k_2} 个。

对于列 j 上的二分函数 P，覆盖列 j 且 P 值等于 q (q 的值为 0 或 1) 的行在列 j 上取'0'值的行数计作 $N_0(P, j, q)$， 取'1'值的行数计作 $N_1(P, j, q)$。

定义 5.2 投影和扩展：令 P 是定义在行集 R 上的一个二分函数，R'是 R 的子集，P'是定义在 R'上的一个二分函数，如果对于任意行 $i\in R'$, 有 $P(i)=P'(i)$，则 P'是 P 在 R'上的投影，而 P 是 P'在 R 上的一个扩展。

下面分别阐述 P_MEC 和 P_MEC/GI 参数化算法

5.3.1 P_MEC 参数化算法

首先考虑 SNP 矩阵的前面 j 列。

定义 5.3 $E[P, j]$, $ES[P, j]$: P 为列 j 上的二分函数，$E[P, j]$定义为在满足下述条件下 **M** 中必须翻转的元素个数, 对应的必须翻转的元素的集合记作 $ES[P, j]$：

(1) 对 **M** 中的在 $ES[P, j]$中的元素进行翻转后，存在着一个划分把 **M** 中的行划分成两个子集，使得同一个子集中的行在列 1~j 上不冲突;

(2) 对于覆盖列 j 的任意行 i，如果它被划分在子集 0 中当且仅当 $P(i)=0$；如果它被划分在子集 1 中当且仅当 $P(i)=1$。

由定义 5.3 容易验证如下事实：对于一个 $m\times n$ SNP 矩阵 **M**，**M** 的 P_MEC 解等于最小的 $E[P, n]$, P 为列 n 上的二分函数，即

$$\text{P\_MEC}(\mathbf{M})=\min_{\text{列}n\text{上所有的二分函数}P}(E[P,n]) \tag{5-1}$$

对于覆盖列 j 的行的集合 $rowset(j)$而言，按照列 j 上的二分函数 P 可以把 $rowset(j)$划分成 S_0 和子集 S_1。对于 $q=0$ 或 1，如果要使 S_q 中的行在列 j 上不冲突，且使翻转的元素最少，就必须维持该子集中大部分行在该列上都相同的的值，翻转该子集中少部分行在该列上都相同的的值，即：如果 $N_0(P,j,q)<N_1(P,j,q)$，则 S_q 中的行在列 j 上取‘0’值的元素应该翻转，否则 S_q 中的行在列 j 上取‘1’值的元素应该翻转。由上述方式得到的按照 P 划分到 S_q 中的行在列 j 上应该翻转的值记作 $Minor(P,j,k)$，即：如果 $N_0(P,j,q)<N_1(P,j,q)$，则 $Minor(P,j,k)=$‘0’；否则 $Minor(P,j,k)=$‘1’。

这样，我们得到了以下公式：

$$ES[P,1] = \{ (i,\ 1) \mid i \in rowset(1) \wedge \mathbf{M}_{i,1} = Minor(P,\ 1,\ P(i)) \} \tag{5-2}$$

$$E[P,1] = \mid \mathrm{ES}[P,1] \mid，即\ \mathrm{ES}[P,1]中要翻转的元素个数 \tag{5-3}$$

把上述概念从一列扩展到相邻的两列。考虑既覆盖列 j 又覆盖列 j+1 的所有行的集合为 $R_C(j,j+1)$。

定义 5.4　$C[P',j]$, $CS[P',j]$: P'为 $R_C(j,j+1)$上的一个二分函数，$C[P',j]$定义为在满足下述条件下 **M** 中必须翻转的最少的元素个数，$CS[P',j]$就是对应的翻转的元素的集合：

(1)　对 **M** 中的所有行，在对 $CS[P',j]$的元素进行翻转后，存在着一个划分把这些行划分成两个子集，使得同一个子集中的行在列 1~j 上不冲突；

(2) 对于 $R_C(j,j+1)$中任意行 i，如果它被划分在子集 0 中当且仅当 $P'(i)=0$；如果它被划分在子集 1 中当且仅当 $P'(i)=1$。

根据定义 5.3 和定义 5.4，为了求出 $C[P',j]$和 $CS[P',j]$，只要扫描一下 P'在 $rowset(j)$上的所有可能的扩展 P，$C[P',j]$就是最小的 $E[P,j]$，即下列等式成立:

$$C[P',\ j]= \min_{P是P'在rowset(j)上的扩展}(E[P,j]) \tag{5-4}$$

$$CS[P',j]=ES[P,j],\ 其中\ P\ 是\ P'在\ rowset(j)上使\ E[j,P]最大的一个扩展 \tag{5-5}$$

对于列 $j>1$，给定列 j 上的二分函数 P，由于 $R_C(j\text{-}1,j)$是 $rowset(j)$的一个子集，P 在 $R_C(j\text{-}1,j)$上的投影 P'是唯一的。根据定义 5.3 和定义 5.4，用得到公式(5-2)和(5-3)的方法，可得出以下公式：

$$ES[P,\ j] = CS[P',\ j\text{-}1] \cup \{ (i,\ j) \mid \mathbf{M}_{i,\ j} = Minor(P,\ j,\ P(i)) \} \tag{5-6}$$

$$E[P,\ j] = \mid ES[P,\ j] \mid \tag{5-7}$$

由上述公式可以得出下面的参数化算法。

P_MEC 算法

Input：$m\times n$ SNP 矩阵 **M**

Output：**M** 的 MEC 解

Step 1 初始化: 对 **M** 中的行按其第一个非空字符所在的列的序号进行非降序排列, 扫描 **M** 得到覆盖每一列的行号的有序集 *rowset*[*j*]及行数 *H*[*j*], *j* 从列 1 到 *n*

Step 2: // 根据公式(5-2)、(5-3)计算初始值

j=1;

for (*P*=0;*P*<$2^{H[j]}$;*P*++)　//二分函数用一个二进制数编码

{　// *E*[*P*]存贮 *E*[*P*,*j*], *ES*[*P*]存贮 *ES*[*P*,*j*]

E[*P*]=0; *ES*[*P*] =∅;

// 用 *CompFlips* 函数计算列 *j* 在 *P* 的划分下要翻转的最少元素，见图 5-1

CompFlips(*j*, *P*, *E*[*P*], *ES*[*P*]);　}

Step 3: while (*j* < *n*)　//根据公式(5-4)~(5-7)递推, MAXINT 为最大整数

{　//用 *Common* 函数得到列 *j* 和 *j*+1 共有的行集中的行数 CR 和

//表示列 *j* 中的行是否覆盖列 *j*+1 的向量 Bits，见图 5-2

Step 3.1:　*Common*(*j*, Bits, CR);

Step 3.2:　**for** (*P*'=0; *P*'<2^{CR}; *P*'++)　*C*[*P*']=MAXINT;

Step 3.3:　**for** (*P*=0; *P* < $2^{H[j]}$; *P*++)

{　//用 *Project* 函数得到 *P* 在 $R_C(j, j+1)$上的投影 *P*'，见图 5-3

Project(*P*, *P*', Bits);

if (*C*[*P*'] > *E*[*P*]) {　*C*[*P*'] = *E*[*P*];　*CS*[*P*'] = *ES*[*P*];　}

Step 3.4:　*j*++; //下一列

Step 3.5:　**for** (*P* = 0; *P* < $2^{H[j]}$; *P*++) { *E*[*P*] = MAXINT;　*ES*[*P*] =∅;}

Step 3.6:　**for** (*P*'=0; *P*'<2^{CR}; *P*'++)

Step 3.6.1:　**for**(*p*=0; *p* < $2^{H[j]-CR}$; *p*++)　//得到 *P*'在 *rowset*(*j*)上的扩展

{　*P* = *P*' | (*p* << CR); //由于 M 中的行有序,这样覆盖列 *j* 而

//不覆盖列 *j*-1 的行的序号比 $R_C(j-1, j)$中的行的序号大

deltEp=0; deltESp =∅;

CompFlips(*j*, *P*, deltEp, deltESp);

if (dletE + *C*[*P*'] < *E*[*P*])

{ $E[P]$ = dletE + $C[P']$;　$ES[P] = CS[P'] \cup$ deltESp; }

}　　}

Step 4: 输出最小的 $E[P]$　($P = 0, \ldots, 2^{H[n]}-1$)　// 根据公式(5-1)

```
CompFlips(j, P, Ep, ESp)   //初始值：Ep = 0，ESp[P] = ∅
{ // As[k], Bs[k]分别表示 N0(P, j, k), N1(P, j, k)，k=0, 1
      As[0] =Bs[0]= As[1] =Bs[1]=0;
      bit=0; shift=P;
      按序取 rowset(j)中的行号赋值给 i   do
       {    bit=shift&1;   // "&"位与，bit=P(i)
            shift=shift>>1; //右移 1 位
            if (M_{i,1}= 0)   As[bit] ++;
            if (M_{i,1}= 1)   Bs[bit] ++;    }
       if ( As[0] < Bs[0] ) Minor[0] = 1; else Minor[0] = 0;
       if ( As[1] < Bs[1] ) Minor[1] = 1; else Minor[1] = 0;
       shift=P;
       按序取 RowSet(j)中的行号赋值给 i   do
       {   bit = shift&1;   shift = shift >> 1;
           if (M_{i,j} = Minor[bit])
           {   Ep++; ESp = ESp ∪ (i, j);   }        }
}
```

图 5-1　计算列 j 在 P 的划分下要翻转的最少元素的函数 *CompFlips*

```
Common(j, Bits, CR)
{   for k = 0 .. H[j]−1 do Bits[k]=0;   // H(j)为覆盖列 j 的行数
    k = 0;
    i1 为 rowset(j)的第一行; i2 为 rowset(j+1)的第一行; //rowset 有序
    while   i1 > -1   and   i2  > -1 do
     {   if i1 > i2 then { i2 = rowset(j+1)中下一行(如果没有则为-1，下同);
         continue; }
         if i1 < i2 then { i1 = rowset(j)中下一行;   k = k+1;   continue; }
         else   // i1=i2
         {   Bits[k]=1;              // rowset(j)中的第 k 行覆盖列 j+1
             i1=rowset(j)中下一行;   i2=rowset(j+1)中下一行;
             k = k+1;  }    }
    CR = k;    }
```

图 5-2　计算列 j、j+1 共有的行数 CR 和表示覆盖列 j 的行是否覆盖列 j+1 的向量 Bits 的函数 *Common*

```
Project(P,P', Bits)
{    P' = 0;     d = 0;
     for k = 0 .. H[j]-1 do            // H(j)为覆盖列 j 的行数
     {    if Bits[k]= 1 then           // P'的第 d 位等于 P 的第 k 位
          {   //   "|", "<<"和">>"分别表示 2 进制位或,左移和右移
               P'=P' | (P 的最低有效位<<d);
               d = d+1;   }
          P = P >>1;   }   // P 右移 1 位
}
```

图 5-3 由 Bits 计算出列 j 上的划分函数 P 在 $R_C(j, j+1)$上的投影 P'的函数 *Project*

定理 5.1 如果 **M** 满足(k_1, k_2)参数化条件,则 P_MEC 算法的时间复杂度为 $O(nk_2 2^{k_2}+mlogm+mk_1)$，空间复杂度为 $O(mk_1 2^{k_2}+nk_2)$。

证明：M 满足(k_1, k_2)参数化条件，则意味着任意一行覆盖的 SNP 位点数不会超过 k_1，覆盖任意一个 SNP 位点的行数不超过 k_2。**M** 采用如下的存储结构：每一行存储其第一个和最后一个非空列的序号，再保存该行从第一个非空列到最后一个非空列的值，这样 **M** 所需的存储空间为 $O(mk_1)$，*rowset* 所需的空间为 $O(nk_2)$，H 所需的空间为 $O(n)$，E 和 C 为 $O(2^{k_2})$，ES 和 CS 为 $O(mk_1 2^{k_2})$，所以算法的空间复杂度为 $O(mk_1 2^{k_2}+nk_2)$。

下面分析其时间复杂度：Step1 需时间 $O(mlogm+mk_1)$；*CompFlips* 函数需时间 $O(k_2)$，所以 Step 2 需时间 $O(k_2 2^{k_2})$；Step 3.1 调用 *Common* 函数需时间 $O(k_2)$，Step3.2 需时间 $O(2^{k_2})$，Project 函数需时间 $O(k_2)$，故 Step3.3 需时间 $O(k_2 2^{k_2})$，Step3.5 需时间 $O(2^{k_2})$，Step3.6 需时间 $O(k_2 2^{k_2})$，所以 Step3 循环 n 次所需时间为 $O(nk_2 2^{k_2})$；Step4 所需时间为 $O(2^{k_2})$。整个算法的时间复杂度为 $O(nk_2 2^{k_2}+mlogm+mk_1)$。定理得证。 □

5.3.2 P_MEC/GI 参数化算法

由于MEC/GI模型中基因型G已经测定，下面根据基因型提供的信息对SNP矩阵进行预处理，以降低求其精确解的复杂性。

令 H_1 和 H_2 是构成基因型 G 的一对单体型。G、H_1 和 H_2 的第 j 个位点的值分别计作 $G[j]$、$H_1[j]$和 $H_2[j]$。

对SNP矩阵M的每一列j，做：

Case 1. $G[j] = 0$：则$H_1[j]$和$H_2[j]$必为0，这样对于SNP矩阵的第j列上的所有

值为1的单元必须翻转成0才能满足MEC/GI模型的要求。记下翻转的单元后，从**M**中删除该列，从G中删去第j个SNP位点的值。

Case 2. $G[j] = 1$：则$H_1[j]$和$H_2[j]$必为1，这样对于SNP矩阵的第j列上的所有值为0的单元必须翻转成1才能满足MEC/GI模型的要求。记下翻转的单元后，从M中删除该列, 从G中删去第j个SNP位点的值。

Case 3 $G[j] = 2$：则$H_1[j]$和$H_2[j]$必须不同，即如果$H_1[j] = 0$, 则$H_2[j] = 1$；如果$H_1[j] = 1$, 则$H_2[j] = 0$。对这种情况，不做任何处理。

经过上述去掉所有纯合位点的预处理后，得到的SNP矩阵记作**M'**,得到的基因型记作G', 显然G'中的值全为2，即构成该基因型的两单体型在所有保留的位点是杂合的。如果**M**满足(k_1, k_2)参数化条件，则**M'**一定满足(k_1, k_2)参数化条件, 且P_MEC/GI(**M**) = P_MEC/GI(**M'**) +预处理M中翻转的单元（元素）个数。

本节下文中的SNP矩阵**M**和基因型G都是指的经过了以上预处理的SNP矩阵和基因型。

为了简洁，对于一个全部是由2（杂合子）组成基因型G，一个SNP矩阵**M**的P_MEC/GI解P_MEC/GI (**M**,G)下面用P_MEC/GI (**M**)表示。

首先考虑 SNP 矩阵 **M** 的前面 j 列构成的 SNP 矩阵 **M**(:, j)。

定义 5.5 $E_G[P, j]$, $ES_G[P, j]$: P 为列 j 上的二分函数，$E_G[P, j]$定义为在满足下述条件下 **M**(:, j)中必须翻转的最少元素个数, $ES_G[P, j]$则是对应的必须翻转的元素集合，**M'**为翻转 **M**(:, j)中属于 $ES_G[P, j]$的元素后得到的 SNP 矩阵：

(1) $ES_G[P, j]$中的元素所在的列属于 1~j；

(2) 存在着一个划分把 **M'**的行划分成两个子集 S_0 和 S_1，使得在同一个子集中的行在列 1~j 上不冲突，且对于覆盖列 j 的任意行 i，如果它被划分在子集 S_0 中当且仅当 $P(i)$=0；如果它被划分在子集 S_1 中当且仅当 $P(i)$=1。

(3) 对于 **M'**中的任意两行 $i_1 \in S_0$ 和 $i_2 \in S_1$, 对属于 1~j 范围内的任意列 l，如果 $\mathbf{M'}_{i_1, l}$ 和 $\mathbf{M'}_{i_2, l}$ 都不是空值（即'–'），则必有 $\mathbf{M'}_{i_1, l} \neq \mathbf{M'}_{i_2, l}$。

当且仅当定义 5.5 的条件(3)满足，从导出矩阵 **M'**中划分在 S_0 中的所有行得出的单体型 H_1 和划分在 S_1 中的所有行得出的单体型 H_2 在所有的位点上才是杂合的。这样的 **M'**才可能与全部是杂合子的基因型兼容。

对于 **M** 和 G（G 中所有的值均为 2）， 根据定义 5.5，要使 SNP 矩阵 **M**(:, j)可行，在 **M**(:, j)中的片段按照二分函数 P 进行划分的情况下，并且导出这两个划分中的行的两个单体型能构成 G 的情况下，最少翻转的元素个数为 $E_G[P, j]$。

因此，下式成立：

$$\text{P\_MEC/GI}(\mathbf{M}) = \min_{P\text{是列}n\text{上的二分函数}}(E_G[P,n]) \tag{5-8}$$

对于覆盖列 j 的行的集合 $rowset(j)$而言，按照列 j 上的二分函数 P 可以把 $rowset(j)$划分成 S_0 和子集 S_1。为了使被划分在同一子集中的所有行均不冲突，必须把同一子集中的行在同一列上的值统一起来。给定 v_q（v_q 的值为 0 或 1，k ＝0, 1），按下述规则对 $\mathbf{M}$ 中第 j 列的元素进行翻转：行 i 如果其 P 值为 q, 则应划分到子集 S_q 中，如果还有 $\mathbf{M}_{i,j}= v_q$，则对 $\mathbf{M}$ 中的元素(i, j)进行翻转，由此可以得到在列 j 上的翻转元素集合，记作 $Flips$。显然为了满足定义 5.5 条件(3)，v_0 和 v_1 应该取不同的值。

为了确定使 $Flips$ 中的元素个数最少的 v_0 和 v_1 值，必须观察列 j 上的元素值。前面提到，对于列 j 上的二分函数 P，覆盖列 j 且 P 值等于 q (q 的值为 0 或 1) 的行在列 j 上取'0'值的行数为 $N_0(P, j, q)$， 取'1'值的行数为 $N_1(P, j, q)$。如果 $N_0(P, j, 0) + N_1(P, j, 1) < N_1(P, j, 0) + N_0(P, j, 1)$, 则令 $Minor_G(P, j, 0) = 0$，$Minor_G(P, j, 1) = 1$；否则 $Minor_G(P, j, 0) = 1$，$Minor_G(P, j, 1) = 0$。

容易验证，当 $v_0 = Minor_G(P, j, 0)$，$v_1 = Minor_G(P, j, 1)$时，在满足定义 5.5 条件(3)的前提下，$Flips$ 最小。

由此可知 $ES_G[P,1]$和 $E_G[P,1]$的值为：

$$ES_G[P,1] = \{ (i, 1) \mid i \in rowset(1) \wedge \mathbf{M}_{i,1} = Minor_G(P, 1, P(i))\} \tag{5-9}$$

$$E_G[P,1] = |ES_G[P,1]|, \quad \text{即 } ES_G[P,1]\text{中要翻转的元素个数} \tag{5-10}$$

考虑既覆盖列 j 又覆盖列 j+1 的所有行的集合为 $R_C(j, j+1)$。

定义 5.6 $C_G[P', j]$, $CS_G[P', j]$: P'为 $R_C(j, j+1)$上的一个二分函数，$C_G[P', j]$定义为在满足下述条件下 $\mathbf{M}(:, j)$中必须翻转的最少元素个数，$CS_G[P', j]$就是对应的翻转元素集合, $\mathbf{M}_C'$为翻转 $\mathbf{M}(:, j)$中属于 $CS_G[P', j]$的位点后得到的矩阵：

(1) $CS_G[P', j]$ 中的元素所在的列属于 1~j；

(2) 对 $\mathbf{M}_C'$中的所有行，存在着一个划分把这些行划分成两个子集 S_0 和 S_1，使得同一个子集中的行在列 1~j 上不冲突，且对于 $R_C(j, j+1)$中任意行 i，如果它被划分在子集 S_0 中当且仅当 $P'(i)=0$；如果它被划分在子集 S_1 中当且仅当 $P'(i)=1$；

(3) 对于 $\mathbf{M}_C'$中的任意两行 $i_1 \in S_0$ 和 $i_2 \in S_1$, 对属于 1~j 范围内的任意列 l，如果 $\mathbf{M}_C'_{i_1, l}$ 和 $\mathbf{M}_C'_{i_2, l}$ 都不是空值（即'–'）,则必有 $\mathbf{M}_C'_{i_1, l} \neq \mathbf{M}_C'_{i_2, l}$ 。

一旦对于 P'在 $rowset(j)$上的所有可能的扩展 P，$E_G[P, j]$都已求出，那么 $C_G[P', j]$就是其中的最小值，即下列等式成立:

$$C_G[P', j] = \min_{P是P'在rowset(j)上的扩展}(E_G[P, j]) \tag{5-11}$$

$CS_G[P', j] = ES_G[P, j]$，其中 P 是 P'在 $rowset(j)$上使 $E_G[j,P]$最大的一个扩展 (5-12)

反过来，对于列 j ($j > 1$)上的某一二分函数 P，其在 R_C (j-1, j)的投影 P'是唯一的，如果知道了 $C_G[P', j\text{-}1]$和 $CS_G[P', j\text{-}1]$，容易证明可由以下等式得出 $E_G[P, j]$和 $ES_G[P, j]$的值:

$$ES_G[P, j] = CS_G[P', j\text{-}1] \cup \{ (i, j) \mid \mathbf{M}_{i, j} = Minor_G(P, j, P(i)) \} \tag{5-13}$$

$$E_G[P, j] = |ES_G[P, j]| \tag{5-14}$$

根据等式(5-9),(5-10)可得出 E_G 和 ES_G 的初始值，然后由(5-11) ~ (5-14)进行递推最终可求出 **M** 的 MEC/GI 解。具体算法如下所示。

P_MEC/GI 算法

Input：$m \times n$ SNP 矩阵 **M**, 长为 n 的基因型 G

Output：**M** 的 MEC/GI 解

Step 1. 预处理: 见本节前几自然段，用集合 S 记下去掉的列上必须翻转的单元, 预处理后 **M** 的列数记作 n';

Step 2. 初始化: 对 **M** 中的行按其第一个非空字符所在的列的序号进行非降序排列, 扫描 **M** 得到覆盖列 j 的行号的有序集 $rowset(j)$及行数 $H[j]$, j 从列 1 到 n';

Step 3. $j = 1$;

```
        for (P = 0; P < 2^H[j]; P++) //二分函数用一个二进制数编码
        { // E_G[P]存贮 E_G[P,1], ES_G[P]存贮 ES_G[P, 1]
            E_G[P] = 0; ES_G[P] = ∅;
// 用 CompGflips 函数根据公式(5-9)、(5-11)计算 E_G[P,1]和 ES_G[P,1]，
// 即计算列 j 在 P 的划分下、满足定义 5.5 条件(3)的前提下要翻转的
// 最少元素，见图 5-4
            CompGflips(1, P, E_G[P], ES_G[P]); }
```

Step 4. **while**($j < n'$) //根据公式(5-11) ~ (5-14)递推, MAX 为最大整数

{ // 用 *Common* 函数得到列 j 和 j+1 共有的行集中的行数 CR 和

// 表示列 j 中的行是否覆盖列 j+1 的向量 Bits，见图 5-2

Step 4.1: *Common*(j, Bits, CR);

Step 4.2: **for** ($P' = 0$; $P' < 2^{CR}$; P'++) $C[P'] = MAX$;

Step 4.3: **for** ($P = 0$; $P < 2^{H[j]}$; P++)

{ // 用 *Project* 函数得到 P 在 $R_C(j, j+1)$上的投影 P'，见图 5-3

Project(P, P', Bits, $H[j]$);

// 公式(5-11)，(5-12)

if ($C[P'] > E_G[P]$) { $C[P'] = E_G[P]$; $CS_G[P'] = ES_G[P]$; }

Step 4.4: j++; //下一列

Step 4.5: **for** ($P = 0$; $P < 2^{H[j]}$; P++){ $E_G[P] = MAX$; $ES_G[P] = \varnothing$;}

Step 4.6: **for** ($P' = 0$; $P' < 2^{CR}$; P'++)

Step 4.6.1: **for**($p = 0$; $p < 2^{H[j]-CR}$; p++) //得到 P'在 *rowset*(j)上的扩展

{ $P = P' \mid (p << CR)$; //“|”位或，“<<”左移。M 中的行有序,

// 这样覆盖列 j 而不覆盖列 j-1 的行序比 $R_C(j\text{-}1,j)$中的行序大

deltEp=0; deltESp =∅;

CompGflips(j, P, deltEp, deltESp);

// 公式(5-13)和(5-14)

$E_G[P] = deltEp + C_G[P']$; $ES_G[P] = BS_G[P'] \cup deltESp$;

} }

Step 5: 找出最小的 $E_G[P]$对应的 P，记作 min$P$ ($P = 0, ..., 2^{H[n]} - 1$);

Step 6: **M** 的 MEC/GI 解为 $E_G[\min P] + |S|$, 对应的翻转的位点在 $ES_G[\min P]$ 和 S 中；

定理 5.2 如果 **M** 满足(k_1, k_2)参数化条件, P_MEC/GI 算法求出 $m \times n$ SNP 矩阵的 MEC/GI 解的其时间复杂度为 $O(nk_2 2^{k_2} + mlogm + mk_1)$，空间复杂度为 $O(mk_1 2^{k_2} + nk_2)$。

定理 5.2 的证明类似定理 5.1，在此不再赘述。

```
CompGflips (j, P, Ep, ESp)   //初始值：Ep = 0，ESp[P] = ∅
{ // As[k], Bs[k]分别表示 N0(P, j, k), N1(P, j, k)，k=0, 1
      As[0] =Bs[0]= As[1] =Bs[1]=0;
      bit=0; shift=P;
      按序取 rowset(j)中的行号赋值给 i   do
       {    bit=shift&1;   // "&"位与，bit=P(i)
            shift=shift>>1; //右移 1 位
            if (M_{i,1}= 0)   As[bit] ++;
            if (M_{i,1}= 1)   Bs[bit] ++;     }
       if (Bs[0] + As[1] < As[0] + Bs[1]) {   Minor_G[0] = 1,   Minor_G[1] = 0; }
       else   {   Minor_G[0] = 0,   Minor_G[1] = 1; }
       shift=P;
       按序取 RowSet(j)中的行号赋值给 i   do
       {   bit = shift&1;   shift = shift >> 1;
           if (M_{i,j} = Minor[bit])
           {   Ep++; ESp = ESp ∪ (i, j);   }          }
}
```

图 5-4　计算列 j 在 P 的划分下、满足定义 7.5 条件(3)的前提下要翻转的最少元素的函数 *CompGflips*

5.4　实验结果

我们对本章的参数化算法 P_MEC 及 P_MEC/GI 和 Wang 等[68]的对应的分支限界及遗传算法进行对比测试。P_MEC 及 P_MEC/G 算法用 C++实现，对应的分支限界及遗传算法的实现来自于 Wang[68]提供的源程序。

测试平台跟前几章相同，为一台 Linux 服务器 (4 个 Intel Xeon 3.6G CPU，4G RAM) ，评价指标仍运行时间(Running time)和重构率(Reconstruction rate)。

单体型和片段数据的生成跟第三章相同，其中实验中的单体型采用 2 种方式得到，第一种采用来自于国际人类基因组单体型图计划[13]2006 年 7 月发布的数据文件 genotypes_chr1_CEU_r21_nr_fwd_phased.gz①真实的单体型。第二种用计算机模拟生成，即首先随机生成指定长度的单体型，根据指定的两个单体型的差异率来随机生成另一个单体型，本节采用差异率与文献[134]一样，为 20%。

在单体型数据的基础上，我们仍然采用著名的 shotgun 测序模拟数据生成器 Celsim[145]。下面的实验生成片段的最小长度为 3，片段的最大长度和覆盖度

①从 http://www.hapmap.org/downloads/phasing/2006-07_phaseII/phased/ 下载而来

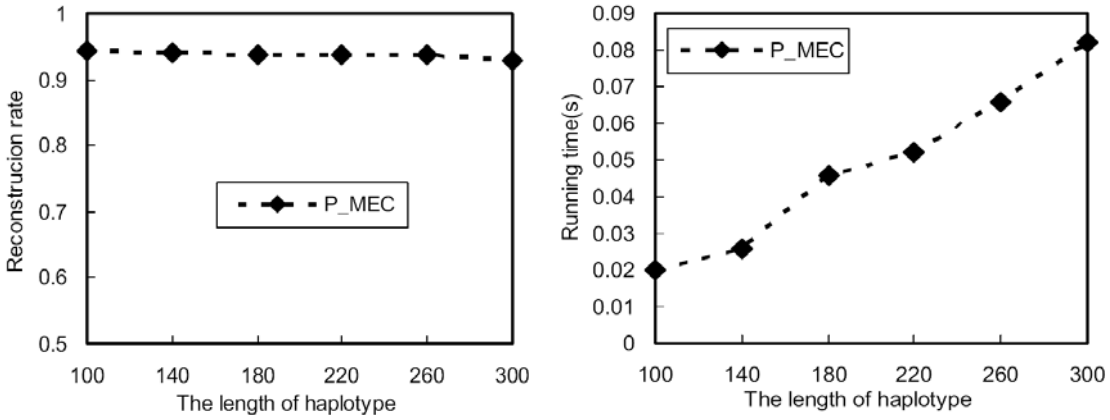
分别为 7 和 10，生成的片段数则按（单体型长度×片段覆盖度/片段平均长度）设置。最后在 Celsim 生成的片段数据的基础上，根据指定的测序误差 e 对片段的 SNP 值进行随机翻转，植入测序错误。根据生成洞的概率 p=2%对普通片段的 SNP 值置空。详细情况请参考 3.4 节及文献[134]和[145]

下面的实验数据均是 100 次重复测试的平均值。

表 5-1 在模拟单体型数据上 P_MEC 算法和 B_MEC 算法的性能比较

Parameters			T_{avg} (s)		T_{max} (s)		*RR*	
n	*m*	*e*	B_MEC	P_MEC	B_MEC	P_MEC	B_MEC	P_MEC
16	32	0.01	0.04	0.002	0.13	0.011	0.985	0.984
		0.03	0.40	0.002	2.34	0.011	0.978	0.980
		0.05	5.62	0.002	56.8	0.013	0.971	0.971
22	44	0.01	186.9	0.002	738.5	0.018	0.962	0.963
		0.03	200.1	0.003	1503	0.019	0.958	0.958
		0.05	532.3	0.003	3214	0.029	0.953	0.952
50	100	0.01	–	0.013	>96 hours	0.030	–	0.949
		0.03	–	0.018	>96 hours	0.037	–	0.945
		0.05	–	0.025	>96 hours	0.038	–	0.942

注：T_{avg}、T_{max} 和 *RR* 分别表示 100 次重复测试的平均时间、最大时间和重构率



(a) 单体型重建率 (b) 运行时间

图 5-5 在模拟的单体型数据上 P_MEC 的性能

表 5-1 在模拟的单体型数据上通过改变单体型长度、片段数 m 和测序误差 e 来比较 P_MEC 算法和 Wang 等[68]对应的分支限界算法 B_MEC 的性能。当 n 和 m 增大时，表 5-1 显示 B_MEC 运行时间急剧上升，当 n 增大到 50，m 增大到 100 时，B_MEC 运行 4 天仍不能得出解，但是这时 P_MEC 所需的时间还不到 1 秒。表 5-1 同时显示，当 e 较小时，作为精确算法，P_MEC 和 B_MEC 均

具有较高的重构率。

当 n 较大时，B_MEC 无法正常运行，图 5-5 显示了当 n 在 100 个 SNP 位点以上时，在模拟的单体型数据上 P_MEC 的性能，实验数据显示当 n=300 时，P_MEC 仍然具有较高的重构率和较短的运行时间。

下面的实验比较 P_MEC/GI 和 Wang 等[68]的对应的分支限界算法 B_MEC/GI 及遗传算法 G_MEC/GI。

表 5-2 MEC/GI 问题各算法性能比较

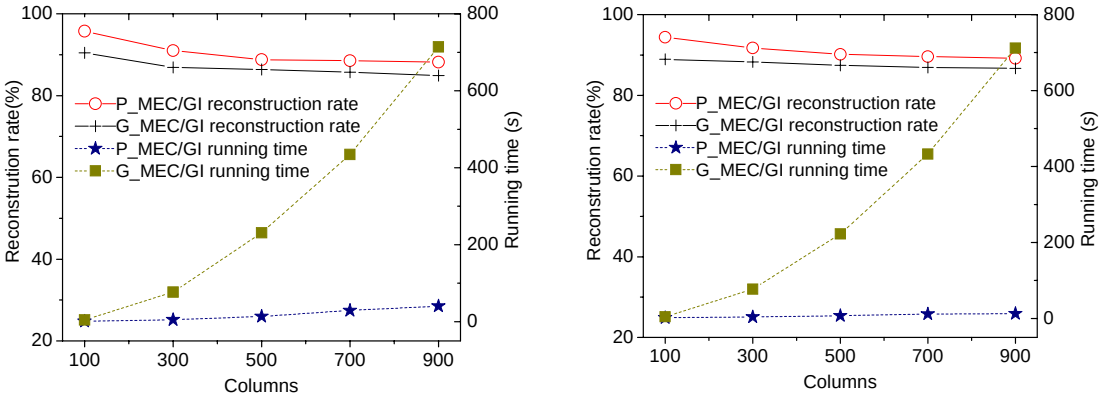
Parameters			*RR*			T_{avg}(s)		
n	*m*	*e*	P_MEC /GI	G_MEC/GI	B_MEC/GI	P_MEC /GI	G_MEC/GI	B_MEC/GI
10	20	0.01	99.25 (99.35)	99.25 (99.35)	99.25 (99.35)	0.11 (0.07)	0.18 (0.19)	0.0002 (0.0016)
		0.03	98.8 (98.9)	98.8 (98.9)	98.8 (98.9)	0.17 (0.21)	0.18 (0.19)	0.0019 (0.0028)
		0.05	98.75 (98.6)	98.75 (98.6)	98.75 (98.6)	0.19 (0.27)	0.19 (0.19)	0.0112 (0.0046)
20	40	0.01	99.25 (98.3)	96.5 (95.55)	98.88 (98.32)	0.49 (0.17)	0.45 (0.44)	5.12 (4.99)
		0.03	97.38 (97.5)	96.4 (95.35)	97.63 (97.5)	0.84 (0.20)	0.45 (0.44)	50.84 (51.96)
		0.05	96.94 (97.95)	96.1 (96.5)	96.94 (97.94)	0.17 (0.22)	0.47 (0.45)	83.03 (80.92)
50	100	0.01	96.98 (98.04)	94.04 (95.36)	-	0.47 (1.09)	1.48 (1.49)	>96 hours
		0.03	96.86 (97.58)	92.45 (91.04)	-	0.56 (1.72)	1.51 (1.48)	>96 hours
		0.05	95.98 (96.16)	91.26 (90.92)	-	1.27 (1.40)	1.58 (1.49)	>96 hours

注：T_{avg} 和 *RR* 分别表示 100 次重复测试的平均时间和重构率,其中没有用括号括起来的是在真实单体型数据上测试的结果，而括号里的则是在模拟的单体型数据上测试的结果。

表 5-2 比较了 MEC/GI 问题的上述三个算法在单体型长度 n，片段数 m 测序误差 e 发生变化时的性能。表中没有用括号括起来的是在真实单体型数据上测试的结果，而括号里的则是在模拟的单体型数据上测试的结果。表 5-2 显示在 m、n 和 e 均较小时，无论是真实单体型数据还是模拟单体型数据上的实验结果均表明上述三个算法均具有较高的单体型重构精度，并且精确算法要优于近似算法，这和文献[68]的结果是一致的。但是当 n 和 m 增加时，B_MEC/GI 的运行时间急剧上升，而遗传算法 G_MEC/GI 的重构率开始明显低于精确算法 P_MEC/GI。

当测序误差 e 固定为 1%不变，n 从 100 增大到 900, m 从 200 增大到 1800

时， 图 5-6 对 P_MEC/GI 和 G_MEC/GI 的性能进行了比较，其中图 5-6(a)是在真实的单体型数据上的测试结果，而图 5-6(b)则是在模拟的单体型数据上的测试结果。每个子图左边的 Y 轴表示算法的重构率，右边的 Y 轴表示算法的运行时间。



(a) 真实的单体型数据 (b) 模拟的单体型数据

图 5-6 P_MEC/GI 和 G_MEC/GI 的性能比较

图 5-6 的实验结果显示，真实的单体型数据上的实验结果与模拟的单体型数据上的实验结果相当接近。当单体型长度和片段数增加时，P_MEC/GI 和 G_MEC/GI 算法的运行时间增加，重构率下降。从实验结果可以看出，和 G_MEC/GI 比，P_MEC/GI 在运行时间和重构率均有优势。

5.5 本章小结

最少错误更正（Minimum Error Correction, MEC）是单体型的组装问题中的一个重要计算模型[146]，Wang 等[68]对 MEC 进行改进，加入个体的基因型信息，引入了 MEC/GI (MEC with Genotype Information) 计算模型。

即使对于无空隙的 SNP 矩阵，MEC 模型是 NP-难，当片段数据中至少有一个空隙时，MEC 模型也是 APX-难的。在一般情况下，MEC/GI 模型也是 NP-难的。为求出上面这两个模型的精确解，Wang 等[68]曾设计了时间复杂度为 $O(2^m)$ 的分支限界算法，其中 m 为 SNP 矩阵的行数。对于片段数较多的情况，Wang 等[68]设计了遗传算法求其近似解。

本章根据生物实验中能直接测序的 DNA 片段覆盖的最大 SNP 位点 k_1 和覆

盖一个 SNP 位点的最大片段数 k_2 均较小的事实，对 MEC 和 MEC/GI 进行参数化建模，在此基础上设计出求解这两个模型的精确算法 P_MEC 和 P_MEC/GI。这两个算法的时间复杂度均为 $O(nk_2 2^{k_2}+mlogm+mk_1)$，空间复杂度均为 $O(mk_1 2^{k_2}+nk_2)$。实验结果表明，P_MEC 和 P_MEC/GI 和 Wang 等[62]的对应的分支限界算法具有相同的重构精度，而在片段数 m 达到 100，Wang 等提出的分支限界算法已无法运行的情况下，P_MEC、P_MEC/GI 和 Wang 等提出的遗传算法一样，仍然能快速运行。作为精确算法，P_MEC 和 P_MEC/GI 在单体型重构精度上比 Wang 等[62]对应的遗传算法有明显优势。

第六章 WMEC/GS 模型和参数化算法

6.1 引言

前面已经提到单体型的组装问题目前已有很多计算模型[17, 63, 146]，主要包括MFR、MSR 和 MEC。MEC 模型也叫 MLF (Minimum Letter Flips)模型[82]。在该模型的基础上，附加其他生物信息，又出现了一些新的模型。

在DNA测序时，测序仪对测出的每一个碱基值赋予一个置信度以表示该值正确性的可靠程度[69]，这样$m\times n$ SNP矩阵$\mathbf{M}$中所有字符的置信度可以组织成一个$m\times n$的权值矩阵W, 其第i行第j列的值$W_{i,j}$表示对应的$\mathbf{M}_{i,j}$ 的置信度。若$\mathbf{M}_{i,j}$为'–'，则其置信度为0。在此基础上，Greenberg等[82]提出了WMLF (Weighted Minimum Letter Flips) 模型，Zhao等[69]进一步对其进行了形式化：

WMLF模型: 给定一个$m\times n$的SNP矩阵$\mathbf{M}$和一个$m\times n$的权值矩阵W，翻转$\mathbf{M}$中的元素值（0变成1，或1变成0）使得$\mathbf{M}$可行，且W中与翻转元素对应的权值之和最小。

由于个体的基因型比较容易测定，Wang等[68]则加入个体的基因型信息，引入了MEC/GI (MEC with Genotype Information) 计算模型：

MEC/GI 模型：给定一个 SNP 矩阵 $\mathbf{M}$ 和基因型 G，翻转 $\mathbf{M}$ 中最少的元素（0 变成 1，或 1 变成 0），目标是使得翻转后的 SNP 矩阵能由一对构成 G 的单体型导出。

WMLF 和 MEC/GI 比 MLF 或 MEC 模型在重构率上都有所改善，可是 WMLF 只考虑了测序的置信度，而 MEC/GI 只考虑基因型信息，并且 MEC/GI 模型隐含了输入的基因型信息是完全正确的假设，而事实上，实验室基因型分型测试是存在误差的。为了进一步提高单体型重构精度，本章引入一种基于加权 SNP 片段与有误差基因型的高精度单体型组装模型 WMEC/GS（Weighted Minimum Error Correction with GenoSpectrum）。

6.2 基因型谱加权最小错误纠正模型 WMEC/GS

基因型分型测试误差主要分为两大类[147]: 操作误差(operational errors)和基

因型记分误差(genotype scoring errors)。由于技术和实验条件的改善，对目前流行的大规模基因型分型技术来说操作误差已经被大大减小，然而对基因型记分软件中核心算法的研究还在不断深入之中[148-152]。受基因型记分软件的限制，基因型记分误差仍然不可避免，使得大部分基因型数据都含有测量误差[147, 153]，因此在单体型组装问题中加入基因型的置信度信息将有助于单体型重构。

带有置信度信息的基因型（Genootype）可以用如图 3 所示的一个 $3\times n$ 的基因型谱 (GenoSpectrum) F 来描述[147, 154]，其中 $f_{0,j}$、 $f_{1,j}$、 $f_{2,j}$ 分别表示该个体在第 j（$j = 1, \ldots, n$）个 SNP 位点的基因型是 0、1、2 的似然度（对于 $i= 0, \ldots, 2$，$f_{i,j} \geq 0$）。

	SNP_1	SNP_2	…	SNP_n
Genotype 0:	$f_{0,1}$	$f_{0,2}$	…	$f_{0,n}$
Genotype 1:	$f_{1,1}$	$f_{1,2}$	…	$f_{1,n}$
Genotype 2:	$f_{2,1}$	$f_{2,2}$	…	$f_{2,n}$

图 6-1 基因型谱（GenoSpectrum）

在引入 WMEC/GS 模型之前，下面首先给出一些记号和定义。

对于单体型 H 和基因型 G，它们的第 i 个字符分别记作 $H[i]$和 $G[i]$。

定义 6.1 给定一个 $m\times n$ SNP 矩阵 $\mathbf{M}$ 和一个 $3\times n$ 基因型谱 F，$\mathbf{M}$ 基于 F 的最大似然基因型 G 定义为：对于某列 j: $1 \leq j \leq n$，若不存在行 i 使 $\mathbf{M}_{i,j} \neq$‘−’，则 $G[j]$的值等于使 $f_{k,j}$ 最大的 k；否则必存在行 i 使得 $\mathbf{M}_{i,j} \neq$‘−’，在此情况下，如果还存在另外一行 l, 有 $\mathbf{M}_{l,j} \neq$‘−’，且 $\mathbf{M}_{i,j} \neq \mathbf{M}_{l,j}$，则 G 在第 j 个位点上为杂合子，即 $G[j]$=2，否则 G 在第 j 个位点上为纯合子，令 $G[j]=\mathbf{M}_{i,j}$。

$\mathbf{M}$ 基于 F 的最大似然基因型记作 $G_{\mathbf{M},F}$, 其第 j 列的值记作 $G_{\mathbf{M},F}[j]$。

定义 6.2 一个 $m\times n$ SNP 矩阵 $\mathbf{M}$ 和一个 $3\times n$ 基因型谱 F 的距离定义为:

$$d(\mathbf{M}, F) = \sum_{j=1}^{n} (1 - f_{G_{\mathbf{M},F}[j],\, j})$$

定义 6.3 给定一个基因型加权系数 g_w、一个 $m\times n$ SNP 矩阵 $\mathbf{M}$、一个相应的 $m\times n$ 权值矩阵 W、一个 $3\times n$ 基因型谱 F 和 $\mathbf{M}$ 的一些元素的集合 $S \subseteq \{\mathbf{M}_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$，令翻转 $\mathbf{M}$ 中属于 S 的所有元素后得到的矩阵为 $\mathbf{M}'$，定义在 $\mathbf{M}$ 中翻转属于 S 中的所有元素基于 g_w、W 和 F 的代价为：

$$Cost(S) = g_w \times d(\mathbf{M}', F) + \sum_{\mathbf{M}_{i,j} \in S} W_{i,j}$$

为了叙述简便， 在 **M** 中翻转属于 S 中的所有元素基于 g_w、W 和 F 的代价也称 S 基于 **M**、g_w、W 和 F 的翻转代价，在无歧义的情况下，简称为 S 的翻转代价。由定义 6.2 可以看出，S 的翻转代价也可以写成如下形式：

$$\begin{aligned} Cost(S) &= \sum_{j=1}^{n}(g_w \times (1 - f_{G_{\mathbf{M}',F}[j],j}) + \sum_{i:\ (i,j)\in S} W_{i,j}) \\ &= n \times g_w + \sum_{j=1}^{n}(\sum_{i:\ (i,j)\in S} W_{i,j} - g_w \times f_{G_{\mathbf{M}',F}[j],j}) \end{aligned} \tag{6-1}$$

基因型谱加权最小错误纠正(WMEC/GS)模型：给定一个基因型加权系数 g_w、一个 $m\times n$ SNP 矩阵 **M**、一个 $m\times n$ 权值矩阵 W 和一个 $3\times n$ 基因型谱 F，翻转 **M** 中的元素值(0 变成 1 ，或 1 变成 0)使得 **M** 可行，且使翻转元素的集合 S 的翻转代价最小。令该最小翻转代价为 WMEC/GS 模型的解，记作 WMEC/GS(g_w, **M**, W, F)。

容易看出当 g_w= 0 时，WMEC/GS 模型就变成了 WMLF 模型。

定理 6.1 即使 SNP 矩阵没有空隙，WMEC/GS 也是 NP-难的。

证明：与证明 WMLF 为 NP-难[69]的方法相同，可以把一个著名的 NP-难的加权 max-cut (WMC)问题在多项式时间内归结为 WMEC/GS。

WMC 问题定义为：给定一个无向图 $G = (V, E)$, 一个给图 G 的每条边附加一个正数的加权函数 w：$E\rightarrow R_+$，找到 V 的一个划分(V_1, V_2)，使得一个顶点在 V_1 中，另一个顶点在 V_2 中的割边的权值之和最大。

给定一个有 n 个顶点的无向图 G=(V,E)及加权函数 w，不失一般性，V 用顶点序号集合{1, .. , n}表示, 第 i 个顶点和第 j 个顶点间的边用(i, j)表示，边(i, j)的权值用 $w(i,j)$来表示（$w(i,j)$>0，且 $w(i,j)$=$w(j,i)$），按照下面规则构造一个 $n\times n$ 的无空隙 SNP 矩阵 **M** 及对应的加权矩阵 W：

$$\mathrm{M}_{i,j} = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \qquad W_{i,j} = \begin{cases} 0 & i \neq j \ \text{ and } \ (i,j) \notin E \\ w(i,j) & i \neq j \ \text{ and } \ (i,j) \in E \\ 1 + \sum_{(k,j)\in E} w(k,j) & i = j \end{cases}$$

其中 $i, j = 1, ..., n$。

然后构造一个 $3\times n$ 的基因型谱 F：对于 j=1 .. n，令 $f_{0,j}$= $f_{1,j}$= 0，$f_{2,j}$=1。令基因型加权系数 g_w 为任意正数。上述构造过程的时间复杂度为 $O(n^2)$。

令 **M** 的行 i 对应着图 G 的顶点 i，这样 V 的一个划分(V_1, V_2)就对应着 **M** 的

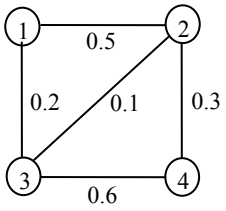
行的一个划分(S_1=V_1，S_2= V_2)。**M** 中只有对角线上的值为 0，即对于任意行 i，只有在列 j=i 上的值为 0，其它列上的值全为 1。从权值矩阵 W 可以看出：把对角线上的 0 翻转成 1 的权值很大（比把其他所有行在该列上的 1 翻转成 0 所对应的权值之和还大 1）。由此可以断定：在翻转元素对应的权值和最小的情况下，要使划分在同一个子集中的行相互兼容，对于 **M** 的任意行 i，如果 $i\in S_1$（或 S_2），则对于列 j=i，行 i 在该列上的值保持‘0’不变，S_1（或 S_2）中的其他行在该列上的值必须由‘1’翻转成‘0’，而 S_2（或 S_1）中的所有行在该列的值保持‘1’不变。这样属于 S_1（或 S_2）的行需要翻转的元素对应的权值之和为 $\sum_{i,j\in S_1,i\neq j} W_{i,j}$（或 $\sum_{i,j\in S_2,i\neq j} W_{i,j}$）。对 **M** 中的元素进行上述翻转后得到的 SNP 矩阵 **M**'显然可行。在 **M**'中，对于任意列 j，如果 S_1 中的行在该列的值为 0（或 1），则 S_2 中的行在该列的值为 1（或 0），这样 **M**'基于 F 的最大似然基因型在每个 SNP 位点上均为 2，这样 **M**'与 F 的距离 d(**M**', F)为 0。令上述翻转元素的集合为 S，则 S 的翻转代价

$$\begin{aligned} Cost(S) &= g_w \times d(\mathrm{M}', F) + \sum_{M_{i,j}\in S} W_{i,j} = 0 + \sum_{i,j\in S_1,i\neq j} W_{i,j} + \sum_{i,j\in S_2,i\neq j} W_{i,j} \\ &= \sum_{i,j\in V,(i,j)\in E} w(i,j) - \sum_{i\in V_1,j\in V_2,(i,j)\in E} w(i,j) - \sum_{i\in V_2,j\in V_1,(i,j)\in E} w(i,j) \end{aligned}$$

从上可以看出，图 G 中一个顶点在 V_1 中，另一个顶点在 V_2 中的割边的权值之和最大当且仅当上述翻转元素的集合 S 的翻转代价最小。定理得证。 □

为了使上述证明过程易于理解，图 6-2 给出了一个实例。对图 6-2(a)中的加权图 G 进行变换得到的 SNP 矩阵 **M** 及对应的权值矩阵 W 如图 6-2(b)所示（其中第 i 行第 j 列小括号外的是 $\mathbf{M}_{i,j}$ 的值，小括号内的是 $W_{i,j}$ 的值），得到的基因型谱 F 如图 6-2(c)所示。对于 G 的顶点集 V 的划分(V_1, V_2)及对应 **M** 的行的划分(S_1, S_2) 如果 V_1=S_1={2, 3}，V_2=S_2={1,4}，这样属于 S_1 的行在列 2 和 3 上的值应统一为 0，在列 1 和 4 上的值保持不变，都为 1，这样行 2 和 3 必须翻转的元素对应的权值之和为 $W_{2,3}$+ $W_{3,2}$=0.1+0.1=0.2；同样属于 S_2 的行在列 1 和 4 上的值应统一为 0，在列 2 和 3 上的值保持不变，为 1，这样行 1 和 4 必须翻转的元素对应的权值之和为 $W_{1,4}$+ $W_{4,1}$=0。这样所有翻转元素对应的权值之和为 0.2。通过上述翻转后，图 6-2(b)中的 SNP 矩阵 **M** 基于图 6-2(c)的基因型谱 F 的最大似然基因型为“2222”，与 F 的距离为 0。这样上述翻转元素的集合 S={$\mathbf{M}_{2,3}$, $\mathbf{M}_{3,2}$}

的翻转代价等于图 6-2(a)中边的权值和的两倍与一端在 V_1 中、一端在 V_2 的割边的权值和的两倍之差，即 0.2=2×(0.5+0.2+0.1+0.3+ 0.6) - 2×(0.5+0.2+0.3+ 0.6)。



(a)

$$\begin{bmatrix} 0\ (1.7) & 1\ (0.5) & 1\ (0.2) & 1\ (0) \\ 1\ (0.5) & 0\ (1.9) & 1\ (0.1) & 1\ (0.3) \\ 1\ (0.2) & 1\ (0.1) & 0\ (1.9) & 1\ (0.6) \\ 1\ (0) & 1\ (0.3) & 1\ (0.6) & 0\ (1.9) \end{bmatrix}$$

(b)

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

(c)

图 6-2 (a) 加权图； (b) 对应(a)中加权图的 SNP 矩阵和权值矩阵（括号里的是权值）；(c) 对应(a)中的加权图的基因谱

事实上，绝大部分单体型组装模型均是 NP-难的，求解这些模型的精确解通常是分支限界算法，其时间复杂度达到了 $O(2^m)$。当片段较多时，作为替代的遗传算法、动态聚类算法等都是启发式算法，无法保证其精确度。近年来在为 NP-难问题设计实用精确算法的领域里，小参数理论得到了广泛运用。跟前面几章相似，下一节将继续利用小参数理论来设计 WMEC/GS 模型的精确算法。

6.3 WMEC/GS 的参数化算法

我们仍然采用用第三章提出的(k_1, k_2)参数化条件。(k_1, k_2)参数化条件定义为片段覆盖的 SNP 位点数不超过 k_1，覆盖任意 SNP 位点的片段数不超过 k_2。在下面叙述中，总是假定 SNP 矩阵满足(k_1, k_2)参数化条件。下面详细阐述 WMEC/GS 模型的参数化算法 P_WMEC/GS。

对于 SNP 矩阵 **M** 的一个 WMEC/GS 解来说，翻转对应的元素后，所有的行应该可以划分成两个子集 S_0 和 S_1，使得在同一子集中的行相互兼容。这样，**M** 中的任意一行，它不是被划分在 S_0 中，就是被划分在 S_1 中。这样我们仍然可以采用第五章的下述定义: 二分函数, 投影和扩展。其中，二分函数 P 是定义在某个行集 R 上的一个映射：$R \rightarrow \{0,1\}$, 即 P 把 R 中的行映射到 0 或 1。而投影和扩展的定义如下：令 P 是定义在行集 R 上的一个二分函数，R'是 R 的子集，P'是定义在 R'上的一个二分函数，如果对于任意行 $i \in R'$, 有 $P(i)=P'(i)$，则 P'是 P 在 R'上的投影，而 P 是 P'在 R 上的一个扩展。

为了叙述简便，**M** 前 j 列所有非空值元素的集合记作 **M**(j)，**M**、W 和 F 的

前 j 列构成的 SNP 矩阵、权值矩阵和基因型谱分别记作 $\mathbf{M}(:,j)$、$W(:,j)$和 $F(:,j)$。

定义 6.4 可行翻转元素集：给定一个 $m\times n$ SNP 矩阵 $\mathbf{M}$ 及行集 R 上的二分函数 P，对 $\mathbf{M}(j)$的任一子集 S_e，如果下面的条件满足，则 S_e 叫做 $\mathbf{M}$ 基于 P 直到列 j 的一个可行翻转元素集：翻转 S_e 中的元素后，对 $\mathbf{M}$ 的行存在着一个划分 (S_0, S_1)，使得划分在同一个子集中的行在列 1~j 上不冲突，且对于属于 R 的任意行 i，如果它被划分在子集 S_k 中当且仅当 $P(i)=k$，$k = 0, 1$。

令覆盖列 j 的行的有序集为 *rowset*(j)，*rowset*(j)上的二分函数被简称为列 j 上的二分函数。显然对于一个满足(k_1,k_2)参数化条件的 SNP 矩阵，覆盖列 j 的行数不会大于 k_2，那么在列 j 上的可能的二分函数 P 的个数不超过 2^{k_2}。这样 P 就可以编码为一个不超过 k_2 位的二进制数，如果行 i 是 *rowset*(j)中的第 k 行，则 $P(i)$就由 P 的第 k 位来表示。

定义 6.5 WES[P, j], WE[P, j]: 给定基因型加权系数 g_w、$m\times n$ SNP 矩阵 $\mathbf{M}$、对应的 $m\times n$ 加权矩阵 W、$3\times n$ 基因型谱 F 及列 j 上的一个二分函数 P，在 $\mathbf{M}$ 基于 P 直到列 j 的所有可行翻转元素集中，取得基于 $\mathbf{M}(:,j)$、g_w、$W(:,j)$和 $F(:,j)$翻转代价最小值的集合定义为 WES[P, j]，对应的最小翻转代价定义为 WE[P, j]。

显然，下式成立：

$$\text{WMEC/GS}(g_w, \mathbf{M}, W, F) = \min_{P:P\text{是列}n\text{上的二分函数}}(\text{WE}[P,n]) \tag{6-2}$$

对于 $\mathbf{M}$ 基于 P 直到列 j 的任意一个可行翻转元素集 S_e 而言，由公式(6-1)可以看出，S_e 基于 $\mathbf{M}(:,j)$、g_w、$W(:,j)$和 $F(:,j)$的翻转代价为 $j\times g_w+\sum_{l=1}^{j} C_l(S_e)$，其中 $C_l(S_e)$表示 $\sum_{i:\,\mathbf{M}_{i,l}\in S_e} W_{i,l} - g_w \times f_{G_{\mathbf{M}',F(:,j)}[l],\,l}$，$\mathbf{M}'$是翻转 $\mathbf{M}(:,j)$中属于 S_e 的元素后得到的 SNP 矩阵，$G_{\mathbf{M}',F(:,j)}[l]$是 $\mathbf{M}'$基于 $F(:,j)$的最大似然基因型第 l 列的值，$f_{G_{\mathbf{M}',F(:,j)}[l],\,l}$是 $F(:,j)$在第 $G_{\mathbf{M}',F(:,j)}[l]$行、第 l 列上的元素值，即在第 j 个位点上基因型值为 $G_{\mathbf{M}',F(:,j)}[l]$的似然度。要使上式最小，只要对于每一列 $l = 1, ..., j$，$C_l(S_e)$的值最小就行。对于没有覆盖某列 l 的行 i 来说，由于其在列 l 上的值为空，元素(i, l)无需翻转，因此 $C_l(S_e)= \sum_{i:\,i\in rowset(j),\,\mathbf{M}_{i,l}\in S_e} W_{i,l} - g_w \times f_{G_{\mathbf{M}',F(:,j)}[l],\,l}$。

对于列 j 上的一个二分函数 P，为了满足定义 6.4 中的条件，覆盖列 j 的行

i 按照 P 值应划分在子集 $S_{P(i)}$中。为了使划分在同一子集中的行在列 j 上不冲突，必须把同一子集中的行在列 j 上的值统一起来。给定 v_k（v_k的值为 0 或 1，k=0, 1），按下述规则对 **M** 中第 j 列的元素进行翻转：覆盖列 j 的行 i 如果其 P 值为 k 且 $\mathbf{M}_{i,j}=v_k$，则对 **M** 中的元素(i, j)进行翻转，由此可以得到在列 j 上的翻转元素集合，记作 $Flips$，这样 $C_j(Flips) = \sum_{i:i\in rowset(j),P(i)=0,\mathbf{M}_{i,j}=v_0} W_{i,j} + \sum_{i:i\in rowset(j),P(i)=1,\mathbf{M}_{i,j}=v_1} W_{i,j} - g_w \times f_{G_{\mathbf{M}',F(:,j)}[j],j}$。翻转 $\mathbf{M}(:, j)$中属于 $Flips$ 中的元素后可得到 SNP 矩阵 **M'**，按照定义 6.1，$G_{\mathbf{M}',F(:,j)}[j]$就可以表示为 v_0 和 v_1 的一个函数 $g(v_0,v_1)$：当 v_0=0、v_1=0 时，$G_{\mathbf{M}',F(:,j)}[j]$ =1；当 v_0=1、v_1=1 时，$G_{\mathbf{M}',F(:,j)}[j]$ = 0，当 v_0=0、v_1=1 或 v_0=1、v_1=0 时，$G_{\mathbf{M}',F(:,j)}[j]$ = 2。

令 $w(P, j, k, v)=\sum_{i:i\in rowset(j),P(i)=k,\mathbf{M}_{i,j}=v} W_{i,j}$，其中 k, v = 0, 1。在 v_0 和 v_1 值的四种可能的组合中，令 $Minor(P, j, 0)$和 $Minor(P, j, 1)$分别表示使 $w(P, j, 0, v_0) + w(P, j, 1, v_1) - g_w \times f_{g(v_0,v_1),j}$最小的 v_0 和 v_1 值。

当 v_0 和 v_1 分别等于 $Minor(P, j, 0)$和 $Minor(P, j, 1)$时，列 j 上需要翻转元素的集合$\{\mathbf{M}_{i,j} \mid \mathbf{M}_{i,j} = Minor(P,j,P(i))\}$记作 $MFlips(P, j)$。对于在列 j 上的任意满足定义 6.4 中条件的翻转元素的集合 $Flips$，可以肯定，当 $Flips = MFlips(P, j)$时，$C_j(Flips)$最小。由此可知：

$$\mathrm{WES}[P,1] = MFlips(P, 1) \tag{6-3}$$

$$\mathrm{WE}[P,1] = g_w + C_1(\mathrm{WES}[P,1]) \tag{6-4}$$

求 $MFlips(P, j)$及 $C_j(MFlips(P, j))$的函数 $CompFlipsW$ 如图 6-3 所示。

对于一个满足(k_1, k_2)参数化条件的 SNP 矩阵 **M** 而言，$CompFlipsW$ 的时间复杂度为 $O(k_2)$。

为了推出其他列的 WES 和 ES 值，下面把上述概念从一列扩展到相邻的两列。考虑既覆盖列 j 又覆盖列 j+1 的所有行的集合为 $R_C(j, j+1)$。

定义 6.6 WBS[P', j], WB[P', j]: 给定基因型加权系数 g_w、$m\times n$ SNP 矩阵 **M**、对应的 $m\times n$ 加权矩阵 W、$3\times n$ 基因型谱 F 及 $R_C(j, j+1)$上的一个二分函数 P'，在 **M** 基于 P'直到列 j 的所有可行翻转元素集中，取得基于 $\mathbf{M}(:, j)$、g_w、$W(:, j)$和 $F(:, j)$翻转代价最小值的集合定义为 WBS[P', j]，对应的最小翻转代价定义为 WB[P', j]。

```
CompFlipsW(j, P, MFlips, C)          // MFlips表示MFlips(P, j)，C表示Cj(MFlips(P, j)
{   for k, v = 0, 1 do w(P, j, k, v)=0;      // w (P, j, k, v)的值初始化为0
    shift=P;     // 划分函数P用2进制编码，覆盖列j的第k行的划分函数值为P的第k位
    按序取rowset(j)中的行号给i do                    //   该循环得到w (P, j, k, v)的值
   {    shift的最低有效位给k；  shift右移1位；  v = M_{i,j};          // P(i)=k
        w(P, j, k, v)= w(P, j, k, v)+W_{i,j};                        // W为权值矩阵
   }
    v0. = v1 = 1;   g = 0 ;    //经过下面3条if语句后，v0 =Minor(P, j, 0), v1=Minor(P, j, 1)
    if (w (P, j, 0, 0) + w (P, j, 1, 0) − g_w× f_{1,j} < w (P, j, 0, v0) + w (P, j, 1, v1) − g_w × f_{g,j}) then
    {    v0 = 0;    v1 = 0;      g = 1;    }
    if (w (P, j, 0, 0) + w (P, j, 1, 1) − g_w× f_{2,j} < w (P, j, 0, v0) + w (P, j, 1, v1) − g_w × f_{g,j}) then
    {    v0 = 0;    v1 = 1;      g = 2;    }
    if (w (P, j, 0, 1) + w (P, j, 1, 0) − g_w× f_{2,j} < w (P, j, 0, v0) + w (P, j, 1, v1) − g_w × f_{g,j}) then
    {    v0 = 1;    v1 = 0;      g = 2;    }
    shift = P;    Flips = ∅;   C = 0;
    按序取RowSet(j)中的行号赋值给i   do
    {    取shift的最低有效位给k;                    // P(i)=k
         shift右移1位;
         if (M_{i,j} = vk) then                      //该位点必须翻转
         {     C = C + W_{i,j};      MFlips = MFlips ∪ M_{i,j};   }
    }
     // f_{g,j}为基因型谱的元素，表示第j个SNP位点的基因型是g的似然度
     C = C − g_w × f_{g,j};
}
```

图 6-3　求 *MFlips*(*P*, *j*)及 C_j(*MFlips*(*P*, *j*))的函数 *CompFlipsW*

显然 $R_C(j, j+1)$是 *rowset*(*j*)的子集，根据定义 6.5 和 6.6，一旦对于 *P*'在 *rowset*(*j*)上的所有可能的扩展 *P*，WE[*P*, *j*]都已求出，那么 WB[*P*', *j*]就是其中的最小值，即下列等式成立:

$$\mathrm{WB}[P', j] = \min_{P:P\text{是}P'\text{在}rowset(j)\text{上的扩展}} (\mathrm{WE}[P, j]) \tag{6-5}$$

$$\mathrm{WBS}[P', j] = \mathrm{WES}[P, j] \mid P = \arg\min_{P:P\text{是}P'\text{在}rowset(j)\text{上的扩展}} (\mathrm{WE}[P, j]) \tag{6-6}$$

反过来，对于列 *j* (*j*>1)上的某一二分函数 *P*，由于 $R_C(j\text{-}1, j)$也是 *rowset*(*j*)的子集，*P* 在 $R_C(j\text{-}1, j)$存在投影 *P*′，且 *P*′是唯一的，这样如果知道了 WB[*P*′, *j*-1]和 WBS[*P*′, *j*-1]，则可由以下等式得出 WE[*P*, *j*]和 WES[*P*, *j*]的值:

$$\mathrm{WES}[P, j] = \mathrm{WBS}[P', j\text{-}1] \cup MFlips(P, j) \tag{6-7}$$

$$\mathrm{WE}[P, j] = \mathrm{BS}[P', j\text{-}1] + g_w + C_j(MFlips(P, j)) \tag{6-8}$$

基于公式(6-1), 按照证明等式(6-3)和(6-4)的方法可以证明公式(6-7)和(6-8)的正确性。

这样根据公式(6-3)和(6-4)可得出 WE 和 WES 的初始值，然后由公式(6-5) ~ (6-8)进行递推最终可求出 **M** 的 WMEC/GS 解。具体算法如下：

P_WMEC/GS 算法

Input：基因型加权系数 g_w, $m\times n$ SNP 矩阵 **M**, 对应的权值矩阵 W 和基因型谱 F

Output：WMEC/GS(g_w, **M**, W, F)

Step 1. 初始化: 对 **M** 中的行按其第一个非空字符所在列的序号进行非降序排列, 扫描 **M** 得到覆盖列 j 的行的有序集 *rowset*(j)及 *rowset*(j)中的行数 $H(j)$, $j = 1 .. n$.

Step 2. $j = 1$;

for $P = 0 .. 2^{H(j)} - 1$ **do**　　//二分函数用一个二进制数编码

{ //　根据公式(6-3), (6-4)计算 WE[P,1]和 WES[P,1]

CompFlipsW (1, P, *MFlips*, C)；

//　WE[P]存贮 WE[P,j], WES[P]存贮 WES[P,j]

WE[P]= g_w+C;　WES[P] = *MFlips*;

}

Step 3. **while** ($j < n$)　//根据公式(6-5)~(6-8)递推, MAX 为最大浮点数

Step 3.1:　计算列j、j+1 共有的行数 CR 和表示覆盖列j 的行是否覆盖列j+1 的向量 Bits：Bits[k]=1 表示 *rowset*(j)的第 k 行覆盖列 j+1;

Step 3.2:　**for** $P' = 0 .. 2^{CR} - 1$ **do**　WB[P'] = MAX;

Step 3.3:　**for** $P = 0 .. 2^{H(j)} - 1$ **do**

{　由 Bits 计算出 P 在 $R_C(j, j+1)$上的投影 P';

if WB[P'] > WE[P] **then**　// 公式(6-5), (6-6)

{　WB[P'] = WE[P];　WBS[P']=WES[P];　}

Step 3.4:　$j = j+1$; //下一列

Step 3.5:　**for** $P = 0 .. 2^{H(j)} - 1$ **do** { WE[P]=MAX;　WES[P] =∅;}

Step 3.6:　**for** $P' = 0 .. 2^{CR} - 1$ **do**

Step 3.6.1:　**for**　P'在 *rowset*(j)上每一个扩展 P　**do**

{　*CompFlipsW* (j, P, *MFlips*, C);

WES[P] = WBS[P'] ∪ $MFlips$;　// 公式(6-7), (6-8)

WE[P] = WB[P'] + g_w + C ;　}

Step 4: 找出最小的 WE[P]对应的 P，记作 min$P$　(P = 0 .. $2^{H(n)}$−1).

Step 5: WMEC/GS(g_w, **M**, W, F) = WE[min$P$], 对应需要翻转的元素在 WES[min$P$]中.　// 公式(6-2)

定理 6.2 如果 **M** 满足(k_1,k_2)参数化条件,则 P_WMEC/GS 算法的时间复杂度为 $O(nk_2 2^{k_2}+mlogm+mk_1)$，空间复杂度为 $O(mk_1 2^{k_2}+nk_2)$。

证明：**M** 满足(k_1,k_2)参数化条件，则意味着任意一行覆盖的列数不会超过 k_1，覆盖任意一列的行数不超过 k_2, 因此 **M** 可以采用如下的存储结构：每一行存储其第一个和最后一个非空字符所在的列号，再保存该行从第一个非空列到最后一个非空列的值，这样 **M** 所需的存储空间为 $O(mk_1)$，同样 W 所需的存储空间为 $O(mk_1)$；*rowset* 所需的空间为 $O(nk_2)$；F 和 H 所需的空间为 $O(n)$；WE 和 WB 为 $O(2^{k_2})$，WES 和 WBS 为 $O(mk_1 2^{k_2})$，所以算法的空间复杂度为 $O(mk_1 2^{k_2}+nk_2)$。

下面分析其时间复杂度：Step 1 需时间 $O(mk_1+mlogm)$。*CompFlipsW* 函数需时间 $O(k_2)$，所以 Step 2 需时间 $O(k_2 2^{k_2})$。由于 *rowset* 中的行是有序的，只要按顺序扫描 *rowset*(j)和 *rowset*(j+1)中的行就可以完成 Step 3.1 的任务，时间复杂度为 $O(k_2)$，具体实现参见图 5-2，所以 Step 3.1 需时间 $O(k_2)$。Step 3.2 需时间 $O(2^{k_2})$。在 Step 3.3 中，为了计算 P 的投影 P'，只需把 P 中对应 *rowset*(j)− $R_C(j, j+1)$ 中的行的位点去掉即可，时间复杂度也是 $O(k_2)$，具体实现参见图 5-3，而 Step 3.3 最多循环 2^{k_2} 次，需时间 $O(k_2 2^{k_2})$。Step 3.5 需时间 $O(2^{k_2})$。在 Step 3.6.1 中，P' 是 $R_C(j\text{-}1,j)$上的二分函数，由于 $R_C(j\text{-}1, j)$中的行数为 CR，且 **M** 中的行有序，这样在 *rowset*(j) − $R_C(j\text{-}1, j)$中(即覆盖列 j 而不覆盖列 j−1)的行的行数为 $H(j)$−CR, 且这些行的行号比 $R_C(j\text{-}1,j)$中的任意行的行号都要大，由此可知 P'在 *rowset*(j)上的扩展 P 的个数为 $2^{H(j)\text{-CR}}$，并可以通过下列二进制数的位运算得到：$P = P' \mid (p$ 左移 CR 位)，其中 p = 0 .. $2^{H(j)\text{-CR}}$ −1, “|” 表示位或，这样 Step 3.6.1 循环 $2^{\text{CR}} \times 2^{H(j)\text{-CR}}$ 次需时间 $O(k_2 2^{k_2})$。Step 3 循环 n 次所需时间为 $O(nk_2 2^{k_2})$。Step 4 所需时间为 $O(2^{k_2})$。这样整个算法的时间复杂度为 $O(nk_2 2^{k_2}+mlogm+mk_1)$。　□

6.4 实验结果

WMEC/GS 模型采用由 C++语言实现的 P_WMEC/GS 算法。MEC/GI 模型采用 Wang[68]提供的遗传算法程序，本节称之为 GA_MEC/GI, WMLF 则采用 Zhao[69]的动态聚类算法，本节称之为 DC_WMLF，用 C++语言实现。由于 GA-MEC/GI 和 DC-WMLF 均不是精确算法，为了反映模型本身的优劣，在下面测试中，对于 MEC/GI 模型，我们也采用了第五章提出的精确算法 P_MEC/GI，对于 WMLF 模型，我们也设计了类似的参数化精确算法 P_WMLF。测试平台跟前几章相同，为一台 Linux 服务器 (4 个 Intel Xeon 3.6G CPU，4G RAM) ，评价指标为运行时间(Running time)和单体型正确重建率(haplotype correct reconstruction rate (CR))。单体型重建率指的是算法重建出的单体型中正确的 SNP 位点数与总的 SNP 位点数的比值：

$$HR=\max\left((r(h_1', h_1)+ r(h_2', h_2))/2n, (r(h_1', h_2)+ r(h_2', h_1))/2n\right),$$

其中，(h_1, h_2)表示个体真实的单体型，(h_1', h_2')表示通过算法重建出的单体型，它们都包含 n 个 SNP 位点，$r(h_j', h_k)$ 表示 h_j'和 h_k 具有相同值的 SNP 位点数。本文采用的单体型正确重建率与 Zhao[69]采用的正确率 (correct rate (RC)) 和 Wang [68]采用的重构率 (reconstruction rate (RR))不同，他们的评价指标没有考虑空值，而本节测试认为空值是没有正确重构的 SNP。

输入数据生成的方法如下：

单体型和片段数据的生成跟第三章相同，其中实验中的单体型采用 2 种方式得到，第一种采用来自于国际人类基因组单体型图计划[13] 2006 年 7 月发布的数据文件 genotypes_chr1_CEU_r21_nr_fwd_phased.gz①真实的单体型。第二种用计算机模拟生成，即首先随机生成指定长度的单体型，根据指定的两个单体型的差异率来随机生成另一个单体型，本节采用差异率与文献[134]一样，为 20%。

在单体型数据的基础上，我们仍然采用著名的 shotgun 测序模拟数据生成器 Celsim[145]。下面的实验生成片段的最小长度为 3，片段的最大长度和覆盖度分别为 7 和 10，生成的片段数则按（单体型长度×片段覆盖度/片段平均长度）设置。然后在 Celsim 生成的片段数据的基础上，根据指定的测序误差 e_s 对片段的 SNP 值进行随机翻转，植入测序错误。根据生成洞的概率 p=2%对普通片段的 SNP 值置空，详细情况请参考 3.4 节及文献[134]和[145]。

权值矩阵 W 按照文献[69]的方法生成，正确测序的 SNP 位点的权值服从均

① 从 http://www.hapmap.org/downloads/phasing/2006-07_phaseII/phased/ 下载而来

值 0.9、方差 0.05 的正态分布，而没有正确测序的 SNP 位点的权值服从均值 0.8、方差 0.05 的正态分布。

由正确的单体型可以得到正确的基因型，因为实验室采用的基因型分型技术通常有 1%~5%的误差[148]，本文在正确的基因型基础上随机引入错误得到带有分型误差的基因型 G，引入错误的方法如下[147, 153]：如果正确的基因型是 0 或 1，则该基因型变为 2，如果正确的基因型是 2，则该基因型变为 0 或 1 的概率各为 50%。对于基因型谱 F 中没有引入错误的列 j：如果该列的基因型为 0 (或 1)，则 $f_{0,j}$ (或 $f_{1,j}$) 服从方差为 0.05，均值为 0.9 的正态分布，$f_{2,j}$ 等于 max(0, 1-$f_{0,j}$) (或 max(0, 1-$f_{1,j}$))，$f_{1,j}$ (或 $f_{0,j}$)等于 0；如果该列的基因型为 2，则 $f_{2,j}$ 服从方差为 0.05，均值分别是 0.9 的正态分布，$f_{0,j}$ 和 $f_{1,j}$ 服从方差为 0.05，均值分别是 0.1 的正态分布。对于引入错误的列 j：如果正确的基因型为 i，引入错误后的基因型为 k，则 $f_{i,j}$ 和 $f_{k,j}$ 服从方差为 0.05，均值分别是 0.2 和 0.8 的正态分布，而基因型谱 F 中该列剩下元素的值则等于 max(0, 1−$f_{i,j}$−$f_{k,j}$)。

本节改变测序误差 e_s、单体型长度 n (即 SNP 位点数)、基因型分型误差率 e_g 来比较各模型相关算法的性能。

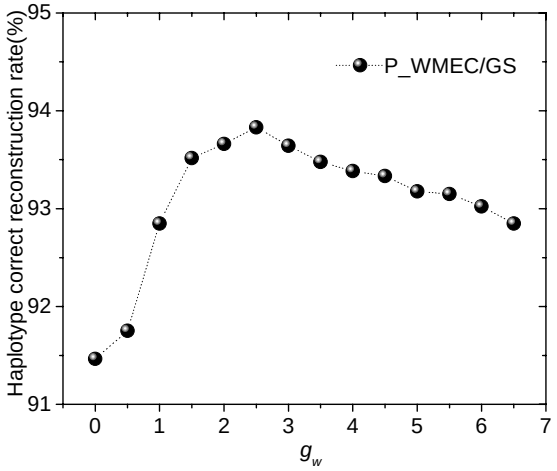


图 6-4 P_WMEC/GS 算法的单体型重建率与基因型加权系数的关系

为了确定一个合适的基因型加权系数 g_w，本节首先采用模拟方法生成长度为 80 的一对单体型对 P_WMEC/GS 算法的单体型正确重建率进行测试，测序误差 e_s 和基因型分型误差率 e_g 从 1%到 5%中随机选择，对 g_w 取 0.0, 0.5, ..., 6.5 中的每个情况进行 100 次测试，平均结果如图 6-4 所示。当 g_w 取 0.0 时，WMEC/GS 模型就退化成 WMLF 模型，P_WMEC/GS 算法求出的也是 WMLF

模型的精确解，从图 6-4 可以看出，g_w取 0.0 比 g_w取 2.5 时的单体型型重建率约低 2.5%。从这次实验的结果大致可以推测，在测序误差 e_s和基因型分型误差率 e_g相差不大时，g_w取值为片段覆盖度的 1/4 较为合适。下面各图或表的每个数据点均为 100 次重复测试的平均值，g_w均采用 2.5。

当测序误差 e_s和基因型分型误差率 e_g在 3%到 7%之间变化时, 我们在真实的单体型数据上对上面五个算法进行测试。测试的单体型长度 n = 100，片段数 m=200，测试结果如表 6-1 所示（表中 GS 列表示 P_WMEC/GS 算法）。表 6-1 的数据显示对于同一模型，精确算法的单体型正确重建率比对应的启发式算法要高，而这三个计算模型中，WMEC/GS 的单体型正确重建率最高。当 e_g＝7%时，e_s从 3%到 7%时，WMES/GS 的单体型正确重建率比 MEC/GI 要高 0.7%到 1.6%; 当 e_s＝7%时，e_g从 3%到 7%时，WMEC/GS 的单体型正确重建率比 WMLF 要高 3.6%到 4.2%。

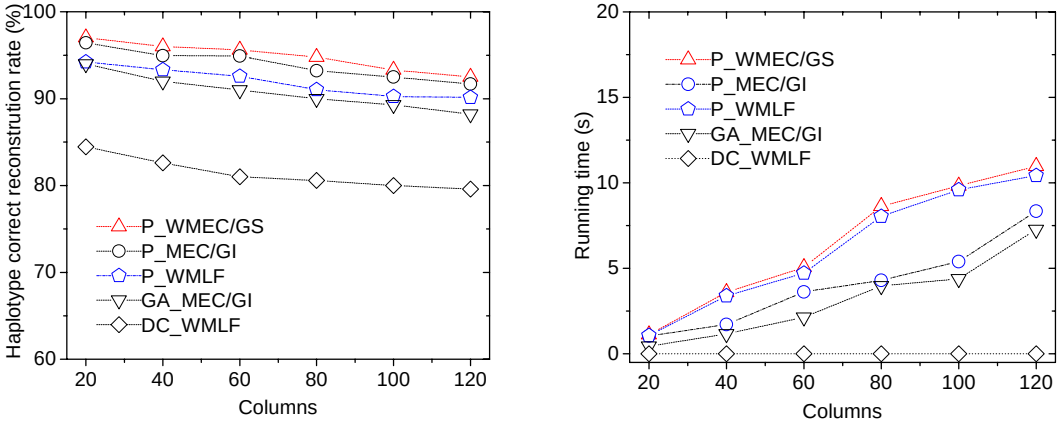
表 6-1 当e_s和e_g变化时各算法的单体型正确重建率比较*

e_s	单体型正确重建率 (%)								
	e_g = 3%			e_g = 5%			e_g = 7%		
	GS	MEC/GI	WMLF	GS	MEC/GI	WMLF	GS	MEC/GI	WMLF
3%	94.3	94.0 (90.2)	91.5 (81.1)	95.0	94.6 (89.6)	92.0 (80.0)	93.9	92.6 (87.5)	92.2 (79.8)
5%	92.1	92.0 (88.3)	89.8 (80.2)	93.5	93.2 (89.7)	89.9 (79.3)	92.9	92.2 (88.4)	89.9 (79.9)
7%	92.1	91.4 (88.6)	87.9 (79.0)	91.6	90.8 (87.5)	88.0 (80.0)	92.8	91.2 (87.4)	89.1 (80.1)

*表格中所有的数据均为在相同参数下真实单体型数据上重复 100 次测试的平均值，其中 GS 表示 P_WMEC/GS 算法，在 MEC/GI 列中，括号内外的数据分别是 GA_MEC/GI 和 P_MEC/GI 算法的单体型正确重建率，在 WMLF 列中，括号内外的数据分别是 DC_WMLF 和 P_WMLF 算法的单体型正确重建率。

在模拟的单体型数据上，当 e_s=e_g=5%、单体型长度 n 发生变化时，上述五个算法的性能比较如图 6-5 所示。从图 6-5 可以看出，随着单体型长度的增长，各算法的单体型正确重建率都有所下降。这五个算法中 P_WMEC/GS 的单体型正确重建率最高，但运行速度最慢，而 DC_WMLF 速度最快，但单体型正确重建率最低。在图 6-5 中，当 n=20 时，P_WMEC/GS、P_MEC/GI、GA_MEC/GI、P_WMLF 和 DC_WMLF 算法的单体型正确重建率分别是 97.03%、96.43%、94.00%、94.23%和 84.45%, 运行时间分别是 1.13、1.04、0.45、1.06 和 0.0004

秒；当长度增加到 120 时，他们的单体型正确重建率分别是 92.48%、91.67%、88.21%、90.13%和 79.58%, 运行时间分别是 10.97、8.35、7.27、10.42 和 0.0083 秒。从这些数据可以看出，虽然 P_WMEC/GS 的运行速度最慢，但是即使单体型长度增加至 120 时，P_WMEC/GS 所需的时间也在 10 秒左右。



(a)单体型正确重建率 (b) 运行时间

图 6-5 单体型长度变化时算法性能比较

从上述实验结果可以看出，由于考虑了基因型及基因型和片段数据可能的误差等多方面的信息，WMEC/GS 模型具有较高的重构率。尽管与 GA_MEC/GI 和 DC_WMLF 算法相比，P_WMEC/GS 需要较多的时间，但是根据时间复杂度理论分析和实验数据来看，在片段覆盖度保持不变的情况下，P_WMEC/GS 所需的时间与重构单体型的 SNP 位点数基本上成线性关系，所需的时间还是比较短的。

6.4 本章小结

个体单体型组装的计算模型目前主要有 MFR、MSR 和 MEC 及其变种，本文结合 DNA 测序仪可提供的碱基置信度及基因型的误差信息，提出了基于加权 SNP 片段与基因型谱的单体型组装模型 WMEC/GS，证明了其是 NP-难的，并且基于片段数据的特点提出了一个时间复杂度为 $O(nk_2 2^{k_2}+mlogm+mk_1)$的参数化精确算法，其中 m 为片段数，n 为单体型的 SNP 位点数，k_1 为一个片段覆盖的最大 SNP 位点数, k_2 为覆盖同一 SNP 位点的片段的最大数(通常不大于 19)。

基于真实单体型数据和模拟单体型数据上的对 MEC/GI、WMLF 和 WMEC/GS 三模型的大量实验表明 WMEC/GS 模型具有最高的单体型重构精度。对于生物实验中获得的 SNP 片段数据，即使 m 和 n 较大，本文提出的参数化算法也可以在不太长的时间得到 WMEC/GS 模型的精确解，具有较高的实用价值。

第七章 总 结

7.1 主要贡献和创新点

分析和识别单体型对复杂疾病的致病基因的精确定位、诊断和药物研究有重要作用。利用计算机技术来确定个体的单体型有极其重要的现实意义。

单体型检测计算问题可以分为单体型组装问题和单体型推断问题。单体型组装具有较高的单体型重构精度并开始尝试用于确定个体的全基因组单体型研究之中[3]。根据不同的优化准则和附加不同的信息，单体型组装问题有了很多计算模型。在通常的情况，这些模型都被证明是 NP-难的。

本文的主要贡献和创新点是通过对 DNA 测序实验数据特征的分析，发现了实验室 DNA 测序片段数据中的一些小的参数，在此基础上，对单体型组装问题的 MSR、MFR、MEC 和 MEC/GI 等模型进行参数化，进而提出对上述模型进行精确求解且在实际应用中快速有效的参数算法, 最后在已有模型的基础上，提出了基于加权 SNP 片段数据和有误差基因型的高精度计算模型 WMEC/GS，证明了其是 NP-难的，设计了求解 WMEC/GS 的参数算法。具体来说有以下几点：

(1) 通过对 DNA 测序片段数据特征的分析,发现了因为技术和测序成本的制约，DNA 测序实验中能直接测定的片段所覆盖的最大 SNP 位点数 k_1 和覆盖一个 SNP 位点的最大片段数 k_2 通常比较小的事实（k_1 通常小于 10，k_2 通常不大于 19）。基于以上事实，本文对 MSR 和 MFR 进行参数化建模。在此基础上，为求解无空隙的 MSR 和 MFR，本文设计了时间复杂度分别为 $O(nk_1k_2+mlogm+mk_1)$和 $O(mk_2^2+mk_1k_2+mlogm+nk_2)$的精确算法 P_MSR 和 P_MFR，其中 m 为片段数，n 为单体型的 SNP 位点数；为求解有空隙 MSR 和 MFR，本文设计了时间复杂度分别为 $O(2^knk_1k_2+mlogm+nk_2+mk_1)$和 $O(2^kmk_1k_2+2^{3k}mk_2^2+mlogm+nk_2+mk_1)$的精确算法 PG_MSR 和 PG_MFR，其中 k 为片段中最大洞数。大量实验结果表明，在 Bafna 等的对应算法基础上，上述算法的效率显著提高，适用于全基因组规模上的单体型组装。

(2) 针对长的 mate-pair 中洞的个数较多的情况，本文提出了求解 MSR 和

MFR 时间复杂度分别为 $O(nk_1k_2 2^{2h}+k_1 2^h+nk_2+mk_1)$和 $O(nk_2 3^{k_2}+m\log m+nk_2+mk_1)$的参数化精确算法 PM_MSR 和 PM_MFR，其中 h 为覆盖同一 SNP 位点且在该位点取空值的片段的最大数。在实际的 DNA 测序数据中，k_2 通常不大于 19，而 h 不大于 17，理论分析和实验结果均表明 PM_MSR 和 PM_MFR 算法所需的时间与片段中洞的个数的最大值 k 没有直接的关系，在片段数据中存在长的 mate-pair 的情况下仍然能有效计算。

(3) 根据实际 DNA 测序片段数据的特点，本文对 MEC 和 MEC/GI 进行参数化建模，进而设计出求解这两个模型时间复杂度均为 $O(nk_2 2^{k_2}+mlogm+mk_1)$ 的精确算法 P_MEC 和 P_MEC/GI。实验结果表明，P_MEC 和 P_MEC/GI 和 Wang 等对应的分支限界算法具有相同的重构精度，而在片段数 m 达到 100，Wang 等提出的分支限界算法已无法运行的情况下，P_MEC、P_MEC/GI 和 Wang 等提出的遗传算法一样，仍然能快速运行。作为精确算法，P_MEC 和 P_MEC/GI 在单体型重构精度上比 Wang 等对应的遗传算法有明显优势。

(4) 基于生物实验中误差是不可避免的，提出了基于加权的 SNP 片段数据和有误差基因型的计算模型 WMEC/GS。对此计算模型，证明了即使片段中无空隙其也是 NP-难的。进而根据片段数据的特点，提出了求解该模型的时间复杂度为 $O(nk_2 2^{k_2}+mlogm+mk_1)$的参数化精确算法 P_WMEC/GS。对 MEC/GI、WMLF 和 WMEC/GS 三模型的大量实验表明 WMEC/GS 模型具有最高的单体型重构精度。

7.2 展望

作为一门结合生物学、计算机科学、统计学等的新兴交叉学科，生物信息学（Bioinformatics）受到国外研究者的高度重视。随着人类基因组计划的提前完成，大量有关生物信息学的研究成果涌现在国际权威刊物和知名学术会议上。近年来，我国诸多学者也开始从事生物信息学的研究。参数计算理论是近年提出的一套计算复杂度理论，为求解实际应用中出现的难解计算问题提供了一条新的途径。本文尝试应用参数计算理论于生物信息学复杂的单体型计算问题之中，从整体来看，参数计算理论在单体型组装问题上得到了成功的应用，对于生物信息学中出现的大量 NP-难问题，本文采用的小参数方法为设计在实际应

用中可以有效运行的算法提供了范例，如果能够找到在实际应用中相对较小的一个或多个参数，设计出只以这些小参数为指数复杂度的算法，那么这些算法在实际应用中就可能有很高的实用价值。当然，本文的工作还是存在一些不足，在相关的研究领域还存在着大量悬而未决的问题，下面结合本文的内容，提出一些未来可以深入研究的方向：

(1) 现有的单体型组装问题的各计算模型模型都只要求一个最优解，而生物学家可能对多个解更感兴趣，因此枚举出多个最优解或最近似的几个解是未来可以研究的方向。具体来说，可有诸如下面的单体型组装问题：

***k*-MSR:** 给定一个体的联配的 SNP 片段数据，令为获得一对单体型 $H_i=(h_{i,1}, h_{i,2})$必须删除的 SNP 位点数为 S_i，求 k 对单体型，使 $\sum_{i=1}^{k} S_i$ 最小。

***k*-MFR:** 给定一个体的联配的 SNP 片段数据，令为获得一对单体型 $H_i=(h_{i,1}, h_{i,2})$必须删除的片段数为 F_i，求 k 对单体型，使 $\sum_{i=1}^{k} F_i$ 最小。

***k*-MEC:** 给定一个体的联配的 SNP 片段数据，令为获得一对单体型 $H_i=(h_{i,1}, h_{i,2})$必须翻转的片段字符数为 L_i，求 k 对单体型，使 $\sum_{i=1}^{k} L_i$ 最小。

(2) 对于单体型分型计算中的单体型推断问题，目前也有大量的计算模型：基于节俭原则的模型、基于进化史的模型、基于家族的模型等，在通常情况下，这些模型也是 NP-难的，目前也缺少实用的精确算法，如何根据问题和实际数据的特点及遗传领域的相关知识发现小的参数，继而设计有效的算法也具有重要的意义。

例如：单体型推断问题中有一类基于节俭原则（Parsimony）的模型，对于这类模型 Clark[12]最先提出了几条基本推断规则，应用这些规则在有些情况下需要随机判断从而使最终的结果很有可能不是最节俭的。后来 Gusfield 等证明了纯节俭模型（Pure Parsimony）是 NP-难的[18, 52, 83]，至今人们仍在努力寻找相关的实用算法。

对于单体型推断问题的基于家族的最小重组单体型配置（Minimum-Recombinant Haplotype Configuration, MRHC)模型， Li 等[64-66]曾证明其是 NP-难的，一个家族中的重组现象发生的次数是否是一个小参数？如果是，该问题是否存在一个实用的参数算法？这些问题值得深入研究。

(3) 复杂疾病全基因组 SNP 连锁分析高复杂度计算问题：随着大规模的 SNP 芯片技术的发展，Affymetrix 公司提供的一次能测 500K SNP 位点的芯片已用于实验室进行复杂疾病的全基因组的 SNP 连锁分析研究。目前绝大部分用于连锁分析的软件都是基于 Lander-Green 或 Elston-Stewart 算法。Elston-Stewart 算法的时间复杂度是 SNP 位点数的指数级，不适用于较多 SNP 位点的连锁分析。Lander-Green 算法的运行时间随家系的位数（bits）增大而指数级增大，其中家系的位数等于家系中的非建立者（non-founder）数量的 2 倍减去建立者（founder）数量。目前被普遍使用的基于 Lander-Green 算法的软件 Genehunter 和 Merlin 最多只能处理不超过 30-bits 家系[155, 156]，对大的家系利用 SNP 芯片进行多 SNP 位点的连锁分析，即使机器能提供足够大的内存，这些软件也可能需要运行几个月的时间，因此寻找较低时间复杂度的全基因组 SNP 连锁分析的算法具有重要的现实意义。下面是一些可能在这方面有所突破的方法：寻找家族 SNP 数据中小参数的特性进而设计参数化算法；选择标签 SNP 位点，利用标签 SNP 位点进行连锁分析；利用家系的 SNP 数据得到个体的单体型数据，利用单体型进行连锁分析；采用分而治之的策略。

参考文献

[1] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. Nature, 2001, 409(6822):860-921.

[2] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. Nature, 2004, 431:931 - 945.

[3] S. Levy, G. Sutton, P. C. Ng, et al. The Diploid Genome Sequence of an Individual Human. PLoS Biology, 2007, 5(10):e254.

[4] P. Taillon-Miller, Z. Gu, Q. Li, et al. Overlapping Genomic Sequences: A Treasure Trove of Single-Nucleotide Polymorphisms. Genome research, 1998, 8(7): 748-754.

[5] D. G. Wang, J. B. Fan, C. J. Siao, et al. Large-scale identification, mapping, and genotyping of single-nucleotide polymorphisms in the human genome. Science, 1998, 280(5366):1077-82.

[6] M. Cargill, D. Altshuler, J. Ireland, et al. Characterization of single-nucleotide polymorphisms in coding regions of human genes. nature genetics 1999, 22 (3):231-238.

[7] The International SNP Map Working Group. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. Nature, 2001, 409(6822):928-933

[8] G. A. Thorisson, L. D. Stein. The SNP Consortium website: past, present and future. Nucleic Acids Research, 2003, 31(1):124-127.

[9] J. C. Stephens, J. A. Schneider, D. A. Tanguay, et al. Haplotype variation and linkage disequilibrium in 313 human genes. Science, 2001, 293(5529):489-493.

[10] Y. Horikawa, N. Oda, N. J. Cox, et al. Genetic variation in the gene encoding calpain-10 is associated with type 2 diabetes mellitus. Nature Genetics, 2000, 26 (2):163-175.

[11] The International HapMap Consortium. A haplotype map of the human genome. Nature, 2005, 437(7063):1299-1320.

[12] A. G. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. Mol. Biol. Evol., 1990, 7(2):111-122.

[13] The International HapMap Consortium. The international HapMap project. Nature, 2003, 426(6968):789-796.

[14] The International HapMap Consortium. Integrating Ethics and Science in the International HapMap Project. Nature Reviews Genetics, 2004, 5:467 -475.

[15] G. A. Thorisson, A. V. Smith, L. Krishnan, et al. The International HapMap Project Web site. Genome Research, 2005, 15:1591-1593.

[16] The International HapMap Consortium. A second generation human haplotype map of over 3.1 million SNPs. Nature, 2007, 449:851-861.

[17] G. Lancia, V. Bafna, S. Istrail, et al. SNPs problems, complexity, and algorithms. In: F. M. Heide (ed.). Proc. Ann. European Symp. on Algorithms (ESA). Berlin -Heidelberg: Springer-Verlag, 2001. Lecture Notes in Computer Science, 2161: 182-193.

[18] D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. J. Computational Biology, 2001, 8(3):305-324.

[19] 刘万清, 贺林. SNP-为人类基因组描绘新的蓝图. 遗传, 1998, 20(6):38-40.

[20] 高秀丽, 景奉香, 杨剑波, et al. 单核苷酸多态性检测分析技术. 遗传, 2005, 27(1):110-122.

[21] 李婧, 潘玉春, 李亦学, et al. 人类基因组单核苷酸多态性和单体型的分析及应用. 遗传学报, 2005, 32(8):879-889.

[22] J. Akey, L. Jin, M. Xiong. Haplotypes vs single marker linkage disequilibrium tests: what do we gain? Eur J Hum Genet, 2001, 9(4):291-300.

[23] A. G. Clark. The role of haplotypes in candidate gene studies. Genet Epidemiol, 2004, 27(4):321-33.

[24] I. Tachmazidou, C. J. Verzilli, M. D. Iorio. Genetic association mapping via evolution-based clustering of haplotypes. PLoS Genet, 2007, 3(7):e111.

[25] B. L. Browning, S. R. Browning. Efficient multilocus association testing for whole genome association studies using localized haplotype clustering. Genet Epidemiol, 2007, 31(5):365-75.

[26] R. W. Morris, N. L. Kaplan. On the advantage of haplotype analysis in the presence of multiple disease susceptibility alleles. Genet Epidemiol, 2002, 23(3): 221-33.

[27] M. P. Epstein, G. A. Satten. Inference on haplotype effects in case-control stud-

ies using unphased genotype data. Am J Hum Genet, 2003, 73(6):1316-29.

[28] M. Zoledziewska, C. Perra, V. Orru, et al. Further evidence of a primary, causal association of the PTPN22 620W variant with type 1 diabetes. Diabetes, 2008, 57(1):229-34.

[29] G. Zhu, K. Carlsen, K. H. Carlsen, et al. Polymorphisms in the endothelin-1 (EDN1) are associated with asthma in two populations. Genes Immun, 2008, 9(1):23-9.

[30] H. Zhou, L. J. Wei, X. Xu, et al. Combining association tests across multiple genetic markers in case-control studies. Hum Hered, 2008, 65(3):166-74.

[31] G. Q. Shen, L. Li, S. Rao, et al. Four SNPs on chromosome 9p21 in a South Korean population implicate a genetic locus that confers high cross-race risk for development of coronary artery disease. Arterioscler Thromb Vasc Biol, 2008, 28(2):360-5.

[32] U. Sauermann, R. Siddiqui, Y. S. Suh, et al. Mhc class I haplotypes associated with survival time in simian immunodeficiency virus (SIV)-infected rhesus macaques. Genes Immun, 2008, 9(1):69-80.

[33] K. L. Chien, M. F. Chen, H. C. Hsu, et al. Genetic association study of APOA1/C3/A4/A5 gene cluster and haplotypes on triglyceride and HDL cholesterol in a community-based population. Clin Chim Acta, 2008, 388(1-2): 78-83.

[34] J. Arcaroli, J. Sankoff, N. Liu, et al. Association between urokinase haplotypes and outcome from infection-associated acute lung injury. Intensive Care Med, 2008, 34(2):300-7.

[35] S. Adamovic, S. S. Amundsen, B. A. Lie, et al. Fine mapping study in Scandinavian families suggests association between coeliac disease and haplotypes in chromosome region 5q32. Tissue Antigens, 2008, 71(1):27-34.

[36] J. Tan, L. Zhang. The consecutive ones submatrix problem for sparse matrices. Algorithmica, 2007, 48(3):287-299.

[37] Y. Yoshikawa, T. Nakayama, K. Saito, et al. Haplotype-based case-control study of the association between the guanylate cyclase activator 2B (GUCA2B, Uroguanylin) gene and essential hypertension. Hypertens Res, 2007, 30(9): 789-96.

[38] Y. Xiao, M. R. Segal, Y. H. Yang, et al. A multi-array multi-SNP genotyping algorithm for Affymetrix SNP microarrays. Bioinformatics, 2007, 23(12):1459-1467.

[39] J. Y. Tzeng, D. Zhang. Haplotype-based association analysis via variance-components score test. Am J Hum Genet, 2007, 81(5):927-38.

[40] M. van Oijen, E. Y. Cheung, C. E. Geluk, et al. Haplotypes of the fibrinogen gene and cerebral small vessel disease. The Rotterdam scan study. J Neurol Neurosurg Psychiatry, 2007.

[41] 彭子文, 陈晓岗, 唐劲松, 等. 生物钟基因隐花色素-1 与精神分裂症核心家系的单体型相对风险分析. 中国神经精神疾病杂志, 2007, 33(8):476-477.

[42] 刘敏, 凌四海, 李文标, 等. BDNF、GRIN1 基因与双相情感障碍的关联研究. 遗传, 2007, 29(1):41-46.

[43] 金明娟, 陈坤, 张爽爽, 等. XRCC1 单核苷酸多态及单体型分布与乳腺癌的相关研究. 浙江大学学报：医学版, 2006, 35(4):370-376.

[44] 陈汉奎, 冯炳健, 梁慧, 等. 单体型分析将家族性鼻咽癌易感基因定位于 4p11～p14 区域. 科学通报, 2003, 48(16):1776-1779.

[45] D. Botstein, N. Risch. Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease. Nat Genet, 2003, 33 Suppl:228-37.

[46] B. S. Weir, L. R. Cardon, A. D. Anderson, et al. Measures of human population structure show heterogeneity among genomic regions. Genome Res, 2005, 15(11):1468-76.

[47] D. A. Hinds, L. L. Stuve, G. B. Nilsen, et al. Whole-genome patterns of common DNA variation in three human populations. Science, 2005, 307(5712): 1072-1079.

[48] D. G. Clayton, N. M. Walker, D. J. Smyth, et al. Population structure, differential bias and genomic control in a large-scale, case-control association study. Nat Genet, 2005, 37(11):1243-6.

[49] P. C. Sabeti, P. Varilly, B. Fry, et al. Genome-wide detection and characterization of positive selection in human populations. Nature, 2007, 449(7164): 913-918.

[50] 杨银峰, 朱月春, 李丹怡, 等. 中国西南阿昌族 G6PD 缺陷的特征及新单体

型 487G＞A/IVS5-612（G＞C）的鉴定. 中国科学：C 辑, 2007, 37(4):460-465.

[51] 黄玮俊, 李彩霞, 拉布, et al. 藏族人群 15 号染色体中心粒区域基因的高精度连锁不平衡和单体型图谱及其与汉族人群的比较. 科学通报, 2006, 51(3):283-291.

[52] D. Gusfield. A practical algorithm for optimal inference of haplotypes from diploid populations. In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB 2000), 2000. 183-189.

[53] D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In: Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB). Washington, DC, United States: Association for Computing Machinery, 2002. 166-175.

[54] D. Gusfield, S. Eddhu, C. Langley. Efficient reconstruction of phylogenetic networks with constrained recombination. In: Proceedings of the 2003 IEEE CSB Bioinformatics Conference, 2003. 363-374.

[55] D. Gusfield. Combinatorial approaches to haplotype inference. Lecture Notes in Computer Science. 2004. 2983: 136-136.

[56] D. Gusfield, S. Eddhu, C. Langley. Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. Journal of Bioinformatics and Computational Biology, 2004, 2(1):173-213

[57] D. Gusfield. An overview of combinatorial methods for Haplotype Inference. In: S. Istrail, et. al. (Eds.). Proceedings of Computational Methods for SNPs and Haplotype Inference. Berlin Heidelberg: Springer-Verlag, 2004. LNBI, 2983: 9-25.

[58] G. Lancia. Integer programming models for computational biology problems. Journal of Computer Science and Technology, 2004, 19(1):60-77.

[59] G. Lancia, R. Rizzi. A polynomial case of the parsimony haplotyping problem. Operations Research Letters, 2006, 34(3):289-295.

[60] V. Bafna, B. V. Halldorsson, R. Schwartz, et al. Haplotypes and informative SNP selection algorithms: don't block out information. In: Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB). Washington, DC, United States: Association for Computing Machinery, 2003.19-27.

[61] V. Bafna, D. Gusfield, G. Lancia, et al. Haplotyping as Perfect Phylogeny: A direct approach. Journal of Computational Biology, 2003, 10(3-4):323-340.

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

[63] R. Rizzi, V. Bafna, S. Istrail, et al. Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. Lecture Notes in Computer Science, 2002. 2452: 29-43.

[64] J. Li, T. Jiang. Efficient rule-based haplotyping algorithms for pedigree data. In: Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB), Washington, DC, United States: Association for Computing Machinery, 2003. 197-206.

[65] J. Li, T. Jiang. Efficient inference of haplotypes from genotypes on a pedigree. Journal of Bioinformatics and Computational Biology (JBCB), 2003, 1(1):41-69.

[66] J. Li, T. Jiang. An exact solution for finding minimum recombinant haplotype configurations on pedigrees with missing data by integer linear programming. In: Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB). Washington, DC, United States: Association for Computing Machinery, 2004. 20-29.

[67] L. Wang, Y. Xu. Haplotype inference by maximum parsimony. Bioinformatics, 2003, 19(14):1773-80.

[68] R. Wang, L. Wu, Z. Li, et al. Haplotype reconstruction from SNP fragments by minimum error correction. Bioinformatics, 2005, 21(10):2456-62.

[69] Y. Zhao, L. Wu, J. Zhang, et al. Haplotype assembly from aligned weighted SNP fragments. Computational Biology and Chemistry, 2005, 29(4):281-287.

[70] X. Zhang, R. Wang, L. Wu, et al. Models and Algorithms for Haplotyping Problem. Current Bioinformatics, 2006, 1(1):105-114.

[71] Z. Li, W. Zhou, X. Zhang, et al. A parsimonious tree-grow method for haplotype inference. Bioinformatics, 2005, 21(17):3475-81.

[72] X. Zhang, R. Wang, L. Wu, et al. Minimum conflict individual haplotyping from SNP fragments and related genotype. Evolutionary Bioinformatics, 2006,

2:271-280.

[73] 王瑞省, 吴凌云, 张继红, 等. 单体型装配问题及其算法. 高校应用数学学报: A 辑 2004, 19(B12):515-528.

[74] 李珍萍, 王勇, 赵玉英, 等. 单体型推断问题与配对图. 高校应用数学学报: A 辑, 2004, 19(B12):567-576.

[75] F. Y. L. Chin, Q. Zhang, H. Shen. k-recombination haplotype inference in pedigrees. Lecture Notes in Computer Science, 2005, 3515: 985-993.

[76] Q. Zhang, F. Chin, H. Shen. Minimum Parent-Offspring Recombination Haplotype Inference in Pedigrees. Lecture Notes in Computer Science, 2005, 3680:100-112.

[77] Q. Zhang, Y. Xu, G. Chen, et al. Maximum-Likelihood Estimation of Haplotype Frequencies in trio pedigrees. In: Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06), 2006. 35-39.

[78] 张强锋, 车皓阳, 陈国良, 等. 最大节约原则下单倍型推导问题的实用算法(英文). 软件学报, 2005, 16(10):1699-1707.

[79] 张强锋, 徐云, 陈国良, 等. 三元家庭基因数据的单体分型和单体型频率估计(英文). 软件学报, 2007, 18(09):2090-2099.

[80] 张强锋, 陈国良, 孙广中. 最大节约原则下单体型推导问题的复杂性. 中国科学技术大学学报, 2006, 36(2):213-218.

[81] R. Lippert, R. Schwartz, G. Lancia, et al. Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. Briefings in Bioinformatics, 2002, 3(1):1-9.

[82] H. Greenberg, W. Hart, G. Lancia. Opportunities for combinatorial optimization in computational biology. INFORMS Journal on Computing, 2004, 16(3): 211-231.

[83] D. Gusfield. Haplotype inference by pure parsimony. Lecture Notes in Computer Science. 2003, 2676: 144-155.

[84] E. Eskin, E. Halperin, R. M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. Journal of Bioinformatics and Computational Biology (JBCB), 2003, 1(1): 1-20.

[85] E. Halperin, E. Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. Bioinformatics, 2004, 20(12):1842-9.

[86] K. Doi, J. Li, T. Jiang. Minimum recombinant haplotype configuration on tree Pedigrees. In: Proceedings of the 3th Annual International Workshop on Algorithms in Bioinformatics (WABI). Berlin/Heidelberg: Springer-Verlag, 2003. 339–353.

[87] Z. S. Qin, T. Niu,J. S. Liu. Partition-ligation-expectation-maximization algorithm for haplotype inference with single-nucleotide polymorphisms. Am J Hum Genet, 2002, 71(5):1242–1247.

[88] L. Excoffier, M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. Molecular Biology and Evolution, 1995, 12(5):921-927.

[89] T. Niu, Z. S. Qin, X. Xu, et al. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. Am. J. Hum. Genet, 2002, 70:157–169.

[90] L. Eronen, F. Geerts,H. Toivonen. A markov chain approach to reconstruction of long haplotypes. In: Proceedings of 9th Pacific Symposium on Biocomputing. World Scientific, 2004. 104–115.

[91] M. R. Garey, D. S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. New York: W. H. Freeman and Company; 1979.

[92] G. Ausiello, P. Crescenzi, G. Gambosi, et al. Complexity and approximation. Springer-Verlag, 1999.

[93] D. S. Hochbaum. Approximation algorithms for NP-hard problem. Boston, MA: PWS Publishing Company, 1997.

[94] R. Motwani, P. Raghavan. Randomized algorithm. Cambridge University Press, 1995.

[95] Z. Michalewicz, D. B. Fogel. How to solve it: modern heuristics. Springer-Verlag, 2000.

[96] R. Niedermeier. Invitation to fixed-parameter algorithms. Oxford University Press, 2006.

[97] M. R. Fellows. Parameterized complexity: the main ideas and some research rrontiers. Lecture Notes in Computer Science, 2001, 2223: 291–307.

[98] R. G. Downey, M. R. Fellows. Parameterized Complexity: Springer-Verlag;

1998.

[99] R. Downey, M. Fellows. Parameterized complexity after almost ten years: review and open questions. In: Proc. of DMTCS'99 and CATS'99, Berlin-Heidelberg: Springer-Verlag, 1999. 1–33.

[100] J. Alber, J. Gramm, R. Niedermeier. Faster exact algorithms for hard problems: a parameterized point of view. Discrete Mathematics, 2001, 229: 3–27.

[101] M. Grohe. The parameterized complexity of database queries. In: Proc. 20th ACM Symp on Principles of Database Systems (PODS'01), 2001. 82-92.

[102] K. Weihe. On the differences between "practical" and "applied". Lecture Notes in Computer Science, 2001, 1982: 1-10.

[103] S. Khot, V. Raman. Parameterized complexity of finding subgraphs with hereditary properties. Lecture Notes in Computer Science, 2000, 1858: 137-147.

[104] J. Chen, X. Huang, I. Kanj, et al. Polynomial time approximation schemes and parameterized complexity. Lecture Notes in Computer Science, 2004, 3153: 500-512.

[105] F. Hüffner, R. Niedermeier, S. Wernicke. Techniques for practical fixed-parameter algorithms. The Computer Journal, 2008, 51(1):7-25.

[106] M. A. Langston, A. D. Perkins, A. M. Saxton, et al. Innovative computational methods for transcriptomic data analysis: a case study in the use of FPT for practical algorithm design and implementation. The Computer Journal, 2008, 51(1):26-38.

[107] J. Chen, J. Meng. On parameterized intractability: hardness and completeness. The Computer Journal, 2008, 51(1):39-59.

[108] D. Marx. Parameterized complexity and approximation algorithms. The Computer Journal, 2008, 51(1):60-78.

[109] J. Gramm, A. Nickelsen, T. Tantau. Fixed-parameter algorithms in phylogenetics. The Computer Journal, 2008, 51(1):79-101.

[110] L. Cai. Parameterized complexity of cardinality constrained optimization problems. The Computer Journal, 2008, 51(1):102-121.

[111] C. Sloper, J. A. Telle. An overview of techniques for designing parameterized algorithms. The Computer Journal, 2008, 51(1):122-136.

[112] J. Guo, J. Gramm, F. Huffner, et al. Compression-based fixed-parameter algo-

rithms for feedback vertex set and edge bipartization. Journal of Computer and System Sciences, 2006, 72(8): 1386-1396.

[113] J. Flum, M. Grohe. Parameterized complexity theory. Springer. 2006.

[114] J. Chen. Parameterized computation and complexity: a new approach dealing with NP-hardness. Journal of Computer Science and Technology, 2005, 20(1):18-37.

[115] J. Chen, I. A. Kanj, W. Jia. Vertex cover: further observations and further improvements. Journal of Algorithms, 2001, 41(2):280-301.

[116] G. H. Gonnet, M. T. Hallett, C. Korostensky, et al. Darwin v. 2.0: an interpreted computer language for the biosciences. Bioinformatics, 2000, 16(2):101-3.

[117] C. Roth-Korostensky. Algorithms for building multiple sequence alignments and evolutionary trees [Ph. D.]. ETH Zűrich, 2000.

[118] U. Stege. Resolving conflicts in problems in computational biochemistry [Ph. D.]. ETH Zűrich, 2000.

[119] J. Cheetham, F. Dehne, A. Rau-Chaplin, et al. Solving large FPT problems on coarse-grained parallel machines. Journal of Computer and System Sciences, 2003, 67(4):691-701.

[120] W. Charatonik. Directional type checking: beyond discriminative types. Lecture Notes in Computer Science, 2000, 1782: 72-87.

[121] S. Y. Kuo, W. K. Fuchs. Efficient spare allocation in reconfigurable arrays. IEEE Design and Test of Computers, 1987, 4(1):24-31.

[122] J. Chen, I. AKanj. Constrained minimum vertex cover in bipartite graphs: complexity and parameterized algorithms. Journal of Computer and System Sciences, 2003, 67(4):833-847.

[123] J. Chen, I. Kanj. On constrained minimum vertex covers of bipartite graphs: improved algorithms. Lecture Notes in Computer Science, 2001, 2204: 55-65.

[124] H. Bodlaender, M. Fellows, M. Hallett, et al. Parameterized complexity analysis in computational biology. Computer Applications in the Biosciences, 1995, 11(1):49-57.

[125] H. L. Bodlaender, R. G. Downey, M. R. Fellows, et al. The parameterized complexity of sequence alignment and consensus. Theoretical Computer Science 1995, 147:31-54.

[126] G. Gottlob, F. Scarcello, M. Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. Artificial Intelligence, 2002, 138(1-2):55-86.

[127] S. Wernicke. On the algorithmic tractability of single mucleotide polymorphism (SNP) analysis and related problems [Ph. D.]. Universität Tűbingen, 2003.

[128] J. A. Douglas, M. Boehnke, E. Gillanders, et al. Experimentally-derived haplotypes substantially increase the efficiency of linkage disequilibrium studies. Nat Genet, 2001, 28(4):361-4.

[129] R. Cilibrasi, L. Iersel, S. Kelk, et al. On the complexity of several haplotyping problems. Lecture Notes in Computer Science, 2005, 3692: 128-139.

[130] R. Cilibrasi, L. van Iersel, S. Kelk, et al. The complexity of the single individual SNP haplotyping problem. Algorithmica, 2007, 49(1):13-36.

[131] Y. Wang, E. Feng, R. Wang, et al. The haplotype assembly model with genotype information and iterative local-exhaustive search algorithm. Computational Biology and Chemistry, 2007, 31(4):288-293.

[132] W. L. Hsu. A simple test for the consecutive ones property. Journal of Algorithms, 2002, 43(1):1-16.

[133] K. S. Booth, G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. Journal of Computer and System Sciences, 1976, 13(3):335-379.

[134] A. Panconesi, M. Sozio. Fast hare: a fast heuristic for single individual SNP haplotype reconstruction. In: I. Jonassen, J. Kim (eds.). Proc. of the 4th Int'l Workshop on Algorithms in Bioinformatics (WABI 2004). Heidelberg: Springer, 2004. LNCS, 3240: 266-277.

[135] W. Qian, Y. Yang, N. Yang, et al. Particle swarm optimization for SNP haplotype reconstruction problem. Applied Mathematics and Computation, 2008, 196(1):266-272.

[136] L. Genovese, F. Geraci, M. Pellegrini. A fast and accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate and low coverage. In: R. Giancarlo,S. Hannenhalli (eds.). Proc. WABI 2007. Berlin Heidelberg: Springer-Verlag, 2007. LNCS, 4645: 49-60.

[137] L. Li, J. H. Kim, M. S. Waterman. Haplotype reconstruction from SNP align-

ment. In: Proc. the Annual International Conference on Computational Molecular Biology (RECOMB 2006). Washington, DC, United States: Association for Computing Machinery, 2003. 207-216.

[138] J. C. Venter, M. D. Adams, E. W. Myers, et al. The sequence of the human genome. Science, 2001, 291(5507):1304-1351.

[139] I. C. Gray, D. A. Campbell, N. K. Spurr. Single nucleotide polymorphisms as tools in human genetics. Human Molecular Genetics, 2000, 9(16):2403-2408.

[140] M. J. Daly, J. D. Rioux, S. F. Schaffner, et al. High-resolution haplotype structure in the human genome. Nature Genetics, 2001, 29(2):229-232.

[141] S. B. Gabriel, S. F. Schaffner, H. Nguyen, et al. The structure of haplotype blocks in the human genome. Science, 2002, 296(5576):2225-2229.

[142] D. H. Huson, A. L. Halpern, Z. Lai, et al. Comparing assemblies using fragments and mate-pairs. Lecture Notes in Computer Science, 2001, 2149: 294-306.

[143] F. Hüffner. Algorithm engineering for optimal graph bipartization. Lecture Notes in Computer Science, 2005, 3503:240-252.

[144] W. Ansorge, H. Voss, U. Wirkner, et al. Automated Sanger DNA sequencing with one label in less than four lanes on gel. Journal of Biochemical and Biophysical Methods, 1989, 20(1):47-52.

[145] G. Myers. A dataset generator for whole genome shotgun sequencing. In: T. Lengauer, R. Schneider, P. Bork, et al. (Eds.). Proc. 7th Int'l Conf. Intelligent Systems for Molecular Biology (ISMB 99). California: AAAI Press, 1999. 202-210.

[146] P. Bonizzoni, G. Della Vedova, R. Dondi, et al. The haplotyping problem: an overview of computational models and solutions. Journal of Computer Science and Technology, 2003, 18(6):675-688.

[147] H. Kang, Z. S. Qin, T. Niu, et al. Incorporating genotyping uncertainty in haplotype inference for single-nucleotide polymorphisms. American journal of human genetics, 2004, 74(3):495-510.

[148] Y. Xiao, M. R. Segal, Y. H. Yang, et al. A multi-array multi-SNP genotyping algorithm for Affymetrix SNP microarrays. Bioinformatics, 2007, 23(12):1459 -1467.

[149] B. Carvalho, H. Bengtsson, T. P. Speed, et al. Exploration, normalization, and genotype calls of high-density oligonucleotide SNP array data. Biostatistics, 2007, 8(2):485-499

[150] G. Hu, H. Y. Wang, D. M. Greenawalt, et al. AccuTyping: new algorithms for automated analysis of data from high-throughput genotyping with oligonucleotide microarrays. Nucleic Acids Res, 2006, 34(17):e116.

[151] W. M. Liu, X. Di, G. Yang, et al. Algorithms for large-scale genotyping microarrays. Bioinformatics, 2003, 19(18):2397-403.

[152] D. C. Walley, B. W. Tripp, Y. C. Song, et al. MACGT: multi-dimensional automated clustering genotyping tool for analysis of microarray-based mini-sequencing data. Bioinformatics, 2006, 22(9):1147 - 1149. .

[153] 朱文圣. 基因型带有误差时单倍型分析的统计方法[博士学位论文]. 长春: 东北师范大学, 2006.

[154] 朱文圣, 郭建华. 病例-对照研究中基因型不确定时单倍型关联分析的似然方法. 中国科学：A 辑, 2006, 36(4):403-417.

[155] L. Kruglyak, M. J. Daly, M. P. Reeve-Daly, et al. Parametric and nonparametric linkage analysis: a unified multipoint approach. Am J Hum Genet, 1996, 58(6):1347-63.

[156] G. R. Abecasis, S. S. Cherny, W. O. Cookson, et al. Merlin--rapid analysis of dense genetic maps using sparse gene flow trees. Nat Genet, 2002, 30(1): 97-101.

致　　谢

在经过百年难遇的漫天冰雪之后，在 2008 年大地回暖、春光初现之时，在满目苍翠的岳麓山下，在简陋但温馨的屋子里，本论文终于得以完成。回顾过去几年攻读博士学位的学习和科研生涯，许多关心帮助我的人和大量感人的事不断浮现在我的脑海，令我心潮澎拜。在此谨向指导、关心和支持我的老师、同学和亲朋好友们致以我最真挚的感谢!

首先，衷心感谢我的指导导师陈建二教授和王建新教授! 两位导师以其渊博的知识、敏锐的学术洞察力、勤奋刻苦的科研精神、平易近人的工作作风和一丝不苟、精益求精的科研态度对我言传身教，使我受益终生。我的每篇成功的学术论文都凝结着他们的心血。在博士课题研究期间，他们在选题、论文的整体构思和撰写论文的各个阶段都给予我悉心的指导。感谢我的两位指导老师，是他们引导我自己去发现问题，解决问题，教会了我科研的方法，大大提升了我的科研创新能力。

感谢陈松乔教授对我的关心和鼓励，感谢张祖平、王伟平、谭长庚、王斌等老师对我的帮助！感谢杨路明教授、胡志刚教授、王国军教授、黄东军教授等对论文初稿提出的宝贵意见。

感谢生物信息课题组李敏、王静、吴璟莉，彭佳扬、李绍华、黄海滨、汪洁、任峻、王明宇、黄新、黄元南、莫秋菊、黄敏、杨德、刘运龙、蔡钊、陈钢、戴义坚、李秀林等所有成员，这几年与他们的讨论和交流使我一步一步地跨入生物信息学的崭新领域，激发着创新的火花，使我学到许多书本上学不到的东西。感谢研究生周伟替我完成了一些实验测试。

感谢在同一个实验室的盛羽、吴艳辉、叶进、黄家玮、周扬、刘耀、郑莹、刘绪崇、刘辉宇、冯启龙、廖卓凡、颜国风、梁俊斌、骆伟忠、刘志雄、黄俊杰、马行坡等师兄弟妹们。回想在实验室里共同学习、讨论和交流的日子，至今倍感亲切。

感谢中南大学信息科学和工程学院的领导和老师给我提供了良好的求学和科研环境。感谢湖南师范大学物理和信息科学学院的领导和老师对我工作和科研的支持，感谢伍祥生教授、王玲教授、藏晓峰老师、张连明老师等通信系的领导和老师对我的关心和帮助。

感谢我年近古稀但仍坚持劳作的双亲，是他们含辛茹苦将我养大，是他们培养了我吃苦耐劳、百折不挠的精神，他们对子女只问付出，而不问回报的无私爱心每时每刻都在激励着我奋发的斗志。

感谢我的儿子谢灵尧。多少次为吸引我的注意，他拿着书本故意向我请教，都被我推向他的母亲。每每想起这些情景，我总是心酸不已。希望今后我能多出一点时间陪他一起嬉戏和学习。

感谢我的妻子刘新求女士，她不但要不断地通过读研、读博来提升自己的教学科研能力，而且还要任劳任怨地肩负起家庭的重任。她在工作、学习之余，还要准备一日三餐，照顾小孩的生活，辅导小孩的学习。她的安慰和鼓励总能使我很快走出情绪的低谷。感谢她对我的温柔和体贴，感谢她陪我度过的这段艰辛岁月。

最后，衷心感谢各位专家和学者的评议和指导!

谢民主

2008 年 2 月 26 日

攻读博士学位期间主要的研究成果

攻读博士学位期间参与的主要科研项目有：

[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)，2005.1~2008.12，主要参与。

[2] 湖南省教育厅资助科研项目—“生物信息学中单体型检测计算机优化算法研究”(06C526)，2006.12~2008.12，主持。

攻读博士学位期间与本课题直接相关的主要论文：

[1] 谢民主, 陈建二, 王建新. 有 Mate-Pairs 的个体单体型 MSR 问题的参数化算法. 软件学报, 2007, 18(9): 2070-2082.

[2] Minzhu Xie, Jianer Chen, Jianxin Wang. Research on Parameterized Algorithms of the Individual Haplotyping Problem. Journal of Bioinformatics and Computational Biology, 2007, 5(3): 795-816.

[3] Minzhu Xie, Jianxin Wang. An Improved (and Practical) Parameterized Algorithm of the Individual Haplotyping Problem MFR with Mate-pairs. Algorithmica (SCI, published online): http://www. springerlink.com/content/p2202u8wrnr65117/

[4] 谢民主, 王建新, 陈建二. 单体型组装问题 MEC/GI 模型的参数化算法. 高技术通讯, 2008, 18(4): 422-428.

[5] Minzhu Xie, Jianxin Wang, Jianer Chen. A Practical Parameterized Algorithm for

the Individual Haplotyping Problem MLF. TAMC 2008, LNCS 4978, pp. 439–450.

[6] 谢民主, 陈建二, 王建新. 个体单体型问题参数化算法研究. 计算机学报(已录用).

[7] Minzhu Xie, Jing Wang. Parameterized Algorithms of the Individual Haplotyping Problem with Gaps. International Journal of Bioinformatics Research and Applications (IJBRA) (Accepted).

[8] Minzhu Xie, Jianxin Wang, Jianer Chen. A Practical Exact Algorithm for the Individual Haplotyping Problem MEC. BMEI 2008 (Accepted).

[9] Minzhu Xie, Jianxin Wang, Jianer Chen. A Practical Exact Algorithm for the Individual Haplotyping Problem MEC/GI. COCOON 2008 (Accepted).

[10] Minzhu Xie, Jianxin Wang, Jianer Chen. A High Accurate Model of the Individual Haplotyping Problem Based on Weighted SNP Fragments and Genotype with Errors. ISMB2008 (Accepted).

[11] Minzhu Xie, Jianxin Wang, Wei Zhou, Jianer Chen. A Practical Parameterized Algorithm for Weighted Minimum Letter Flips Model of the Individual Haplotyping Problem. FAW 2008 (Accepted).