肖存威,石海鹤,王岚,等.基于混合策略的 de novo 序列拼接算法构造[J].江西师范大学学报(自然科学版),2022,46(3):300-307.
XIAO Cunwei,SHI Haihe,WANG Lan,et al.The construction of de novo sequence assembly algorithm based on hybrid strategy[J].Journal of Jiangxi Normal University(Natural Science),2022,46(3):300-307.

文章编号:1000-5862(2022)03-0300-08

基于混合策略的 de novo 序列拼接算法构造

肖存威,石海鹤*,王 岚,程柏良
(江西师范大学计算机信息工程学院,江西 南昌 330022)

摘要: 在对 3 种 de novo(从头)序列拼接的基本策略进行分析的基础上,该文研究了混合策略序列拼接算法的构造过程,从而整合多个单一策略优点;再利用形式化方法和形式化平台方面的优势,结合领域分析建模和产生式编程的方法,构造了 2 个基于 OLC 策略的算法(OLC_assembly_1,OLC_assembly_2)及 1 个基于 DBG 策略的算法(DBG_assembly),进一步组装出在(OLC+DBG)→OLC 混合模式下的算法(简称 ODO 算法);最后,从 GenBank 中选取了 3 个实验样本,从 N50、Contigs number、Coverage 等角度,比较了在 3 个单一策略下的算法和 ODO 构造算法的拼接结果,分析了 coverage depth 和 k 值的变化对拼接结果的影响.实验结果表明:该文实现的 ODO 算法比单一策略在序列拼接时所产生的结果在 N50 和 Coverage 等参数上均有一定的优势.

关键词: de novo 序列拼接;混合策略;领域特征建模;产生式编程;形式化方法
中图分类号: Q 344;TP 312 **文献标志码:** A **DOI:** 10.16357/j.cnki.issn1000-5862.2022.03.13

0 引言

以人类基因组计划(human genome project,HGP)的实施为界,可以将生物信息学(Bioinformatics)的发展大致分为 2 个阶段,分别为前基因组时代和后基因组时代.HGP 的主要贡献是提供了一项新的技术——基因测序[1].测序技术是进行生物研究最基本的技术之一,近几年来,随着测序技术的高速发展,这既降低了测序成本,又推动了生物信息学的发展,对精准医疗和基因检测等研究产生了促进作用.

在测序技术的发展过程中,一共产生了 3 代基因测序技术.第 1 代基因测序技术在分子生物学研究中发挥了重要作用,比较知名的基因测序方法是双脱氧法[2].第 2 代基因测序技术是在 HGP 的研究过程中被逐步开发出来,主要有 Roche 公司的 454[3]、Illumina 公司的 Solexa[4] 和 ABI 公司的 Solid[5] 等技术方法.第 2 代基因测序技术是通过对 DNA 片段复制扩增,把碱基的信号强度放大,加入 dNTP,在每个接头上聚合新的碱基,就可以得到在整个接头上的 DNA 序列信息[6].目前出现了以单分子测序为特点的第 3 代基因测序技术,使得在测序成本和测序通量上有了较大的改善.第 3 代基因测序技术主要以生物科学公司的 HeliScope[7]、牛津纳米孔技术公司的 Nanopore[8] 以及太平洋生物科学公司的 PacBio[9] 等技术方法为主流[10].与第 2 代基因测序技术不同的是,第 3 代基因测序技术既无需打断整条 DNA 链,又无需对 PCR 扩增,就可实现对 DNA 分子的单独测序[11].测序读取的最大长度可达到 12~15 kbp,从而减少了拼接次数,降低了成本.不进行 PCR 扩增,避免了在扩增过程中出现错误,然而相比于测序的准确率来说,第 3 代基因测序技术的平均错误率约为 15%,这远远高于第 2 代基因测序技术的平均错误率(1%).

因为当前测序读取的最大长度的序列也无法将整个基因组的遗传信息表达准确,所以需要将经测序得到的短基因序列通过相关的技术拼接起来,得

收稿日期:2022-01-12
基金项目:国家自然科学基金(62062039,61662035)和江西省自然科学基金(20202BAB202024,20212BAB202017)资助项目.
通信作者:石海鹤(1979—),女,江西乐平人,教授,博士,主要从事生物信息学、形式化验证研究.E-mail:haiheshi@jxnu.edu.cn

到一条完整的基因序列,这就是序列拼接(sequences assembly) ,其流程如图 1 所示.

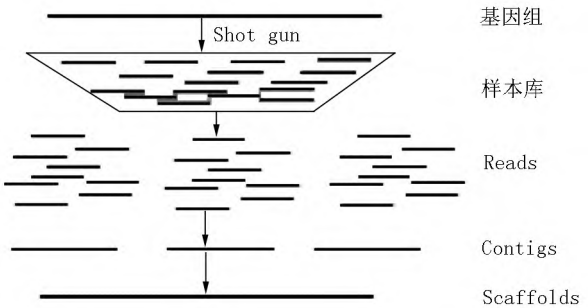

图 1　序列拼接流程图

按照是否有参考序列参与的标准,序列拼接可分为 2 类: 第 1 类是 de novo(从头) 拼接,不需要参考序列,仅依靠短序列之间的重叠关系拼接成长序列; 第 2 类是参考序列拼接,需要本物种或相似物种的基因组序列作为参考序列,以降低拼接过程的复杂度和成本.在实际应用中对一个新物种进行的测序往往没有合适的参考序列,在这种情况下只能采取 de novo 方式进行序列拼接.

随着物种的范围越来越大,序列拼接的准确度和长度要求越来越高,这给序列拼接工作带来挑战.在拼接序列检测方面,目前拼接得到的大多数序列是基于 de novo 拼接的,没有参考基因组,无法从基因组层面对结果进行比对检测,即使有参考基因组参与比对验证,在拼接过程中碱基也可能会发生突变,从而降低拼接的准确度.在测序方面,目前的测序技术无法同时满足低错误率和读长较长的要求,如第 2 代基因测序准确度较高,但是读长较短、测序时间长、费用高,而第 3 代基因测序读长较长、时间短、测序成本低,但准确度较低.在序列拼接算法方面,大多算法是基于单一策略开发的,具有一定的缺陷,如贪婪策略算法是利用启发式搜索方式[12] ,选择与种子 reads 重叠最多碱基的 reads 进行拼接,最后得到局部最优解或次优解,却无法保证得到全局最优解.基于 OLC(overlap-layout-consensus) 策略和 DBG(De Bruijngraph,De Bruijn 图) 策略开发的算法,在单独使用时存在缺陷.对于复杂度较高的基因组,若单独采用 OLC 策略,则杂合 reads 之间的同源性降低,容易导致重叠; 若单独采用 DBG 策略,则所形成的有向图会存在较多的分叉,导致 contigs 长度减小,无法进一步拼接.因此,可以尝试研究在混合策略下的序列拼接问题,期望得到更好的序列拼接结果.

本文在分析同一拼接策略算法实现原理的基础上,进一步研究在相同策略下的不同算法在具体实现过程中的差异性,得到算法在领域内的共同特征和差异特征.利用形式化方法及形式化平台方面的优势,结合领域分析建模和产生式编程的方法,形式化开发算法在领域内的序列拼接算法构件.对于共同特征可以生成单种策略下的必须执行构件,差异特征可以生成在单种策略下的可选执行构件[13] .根据用户对可选构件的选择和输入文件的特征自动组装序列拼接算法.根据(OLC+DBG) →OLC 混合模式将不同策略算法在领域内生成的拼接算法进行组装,构造得到基于混合策略的 ODO 拼接算法.ODO 算法在综合不同策略优势、提高拼接效率和准确度的同时也保证了结果的可靠性.

1　方法和技术

1.1　单一策略拼接

基于贪婪策略开发的序列拼接算法是在整个序列拼接研究过程中最早提出的一类算法,也被称为贪心算法,是从问题的某一个初始值出发,逐步从两边逼近目标值,以尽快地求得一个较好的解,当算法运行到某一步不能继续前进时,算法停止.运行结束后会得到一个局部最优解或次优解,在运行过程中这一类算法所做的每个选择都是在当前状态下最好的选择(贪心选择) ,贪心策略拼接原理示意图如图2(a) 所示.常见的基于贪婪策略开发的序列拼接算法有 SSAKE[14] 、VCAKE[15] 、SHARCGS[16] 、NPSCARF[17] 等.

OLC 策略主要基于 reads 之间的重叠关系进行拼接,是最能直观体现序列拼接原理思想的策略.OLC 策略在序列拼接时将所有待拼接的 reads 构造成一个有向图,图中的每个节点都代表特定的 reads; 若 2 个节点之间存在边,则说明这 2 个 reads 的重叠部分的碱基数量大于设置的阈值.对图进行简化和分割后,可以在一系列子图上寻找经过每个节点一次且仅一次的路径,最后得到所需的序列.由此把序列拼接问题转化为 Hamilton 路径问题,OLC 策略拼接原理示意图如图 2(b) 所示.基于 OLC 策略开发的序列拼接算法有 CABOG[18] 、Phrap[19] 、Celera Assembly[20] 、Edena[21] 等.

DBG 策略的特点是不需要进行序列比对,可大幅度减少系统内存的消耗,De Bruijn 图的大小只与基因组大小和算法复杂度有关.在 DBG 算法中,把 reads 按照长度 k 分割成 k-mer,并将这些 k-mer 存入 Hash 表中,不同 k-mer 在 Hash 表中只存储 1 次,然后将这些 k-mer 作为点,构建 De Bruijn 图.对 De Bruijn 图进行简化和分割,并在图中找到 1 条欧拉

路径构成 contig,再通过 Pair-end 文库可以确定 contig 之间的位置关系,指导 contig 形成长度更长的 scaffold,DBG 策略拼接原理示意图如图 2(c)所示.基于 DBG 策略开发的序列拼接算法有 SOAPdenovo[22]、Velvet[23]、ABySS[24]、EULER[25]、ALLPATHS[26]等.

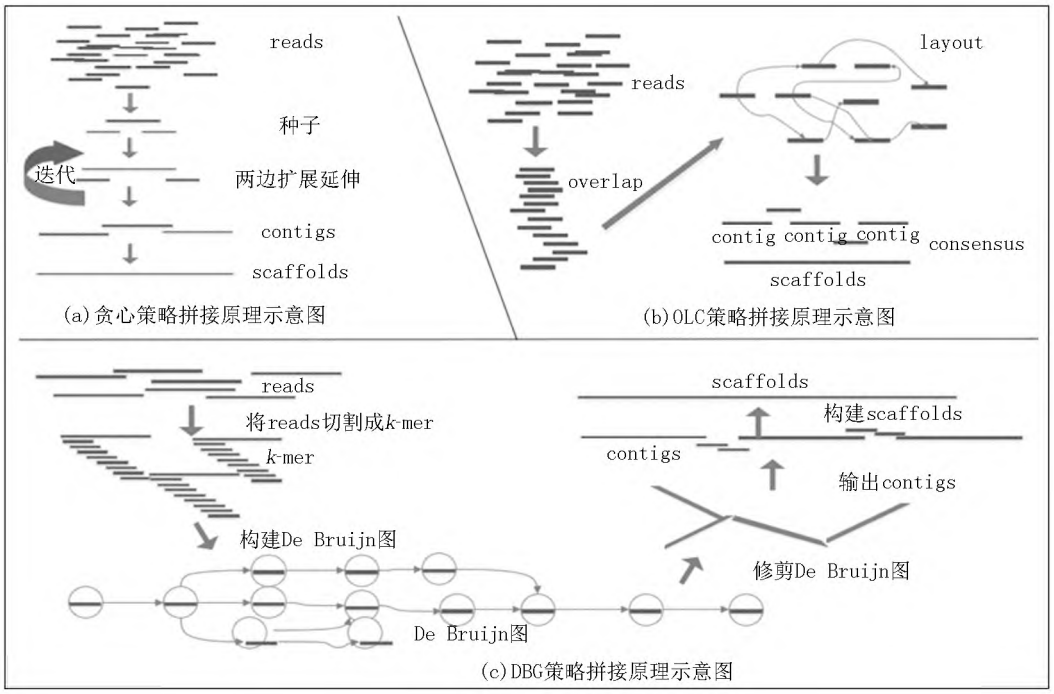

图 2 单一策略拼接原理示意图

1.2 多策略混合拼接

贪婪策略在拼接序列时总是做出在当前情况下最好的选择,不从整体加以考虑,且并非对所有问题都适用.OLC 策略在使用时因 PCR 扩增而会存在大量的重复数据,并且对长 reads 拼接有偏向性.DBG 策略在使用时将短 reads 重新分割成长度更短的 k-mer 进行序列拼接,增加了计算时间和拼接的复杂度.由此可见这些策略在单独使用时均存在不足.

为了解决单一策略在序列拼接时易出现的问题,综合各个策略的优点,这可以使用混合模式来实现策略的混合.如(贪婪策略+DBG)→OLC、贪婪策略+OLC、贪婪策略+DBG、DBG+OLC、(OLC+DBG)→DBG、(OLC+DBG)→OLC、(OLC+OLC)→DBG 以及(DBG+DBG)→OLC 等模式.A.K. Kusuma 等[27]对(OLC+DBG)→OLC 进行了研究,在实验过程中,首先使用 Velvet 算法拼接 reads,然后使用 Edena 算法对相同的 reads 再次拼接,最后使用 Minimus assembler 对拼接产生的 contigs 再次拼接可得到最终结果.使用上述方法,A.K. Kusuma 等[27]在 3 个真实数据集和 4 个模拟数据集上进行了实验.该方法既避免了 DBG 策略拼接导致的错误率过高的现象发生,又解决了 OLC 策略拼接带来的重复序列和占用内存过大的问题.

本文充分利用在软件开发方法上及其在平台方面具有的优势,对(OLC+DBG)→OLC 混合模式的算法进行了研究.使用形式化、产生式编程等方法和技术,分别构造了基于 OLC 策略和基于 DBG 策略的算法域构件库,借助在 PAR 平台上的 Apla→Java 程序生成系统生成了基于 OLC 的拼接算法 OLC_assembly_1、OLC_assembly_2 和基于 DBG 的拼接算法 DBG_assembly,进一步拼装出混合模式的 ODO 算法,从而将尽可能多的创造性劳动转化为非创造性劳动,提高了算法开发过程的可理解性、算法的可靠性以及算法构件库在相关领域中的可重用性,ODO 算法流程图如图 3 所示.

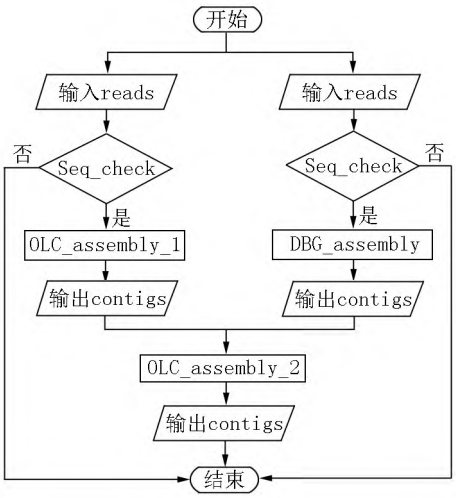

图 3 ODO 算法流程图

2　算法构造

实现相同策略功能的算法可以是不同的,这是因为可按照使用需求生成同种单一策略下的不同算法.软件形式化是实现软件自动化的关键,PAR 是一种统一的软件形式化方法,包含了自定义泛型算法设计语言 Radl、泛型抽象程序设计语言 Apla、统一的算法程序设计方法以及 PAR 平台(Radl 到 Apla 程序生成系统,Apla 到 Java、C++等系列程序生成系统)和其他关键技术[28].根据序列拼接算法的需求设计出相应的程序规约,并在此基础上进行程序设计,得到最终的算法和程序.

首先在领域层次上建立领域模型,然后利用产生式编程对算法构件进行交互设计,再形式化实现算法的构件库,最后在不同策略的构件库中自动生成不同单一策略下的拼接算法.利用生成的序列拼接算法按照(OLC+DBG)→OLC 模式组装后可得 ODO 算法,同时利用形式化方法保证了算法构造的高效性和可靠性.下面叙述了基于 DBG 策略拼接算法的构造过程,阐述了基于 OLC 策略拼接算法的构造过程.

2.1　DBG 策略拼接算法生成

基于 DBG 策略开发的算法有 SOAPdenovo、Velvet、ABySS、EULER 等,将这些序列拼接算法联合起来,构成了基于 DBG 策略的算法拼接领域.通过对在领域内算法的分析,找出算法的共同特征和可变特征并进行分类,选择和定义需要分析和解决的领域,收集相关的领域信息,并形成领域模型[29].根据领域分析的结果,抽取算法共性,用参数表示算法的差异性,设计出 Apla 语言描述的拼接算法构件,为实现完整的算法构件库,进一步分析不同构件之间的交互模型和构件之间的约束关系.

2.2　OLC 策略拼接算法生成

2.2.1　**领域特征模型分析**　将基于 OLC 策略的序列拼接算法联合起来,构成 OLC 算法领域,在领域内的序列拼接算法在进行序列拼接时的过程大致可以分为 3 步: 首先,通过序列比对找到 reads 之间重叠长度超过设置阈值的重叠关系; 其次,通过这些重叠信息将 reads 建立一种新的组合关系,可得到重叠群 contigs,再将 contigs 进一步排列,生成较长的 scaffold 重叠群; 最后,在经过简化和分割后的有向图中找到一条最优的路径(Hamilton 路径)所对应的序列,这即是需要的拼接序列.接下来对在 OLC 算法领域中的算法 CABOG、Celera Assembly 和 Edena 进行分析,找出这 3 个算法的共同特性和可变特性.如在实现序列核查、序列比对、构建有向图等功能中,这 3 种算法使用的方法和步骤是类似的.但在纠正测序错误这个步骤中,Edena 使用快速检测虚假 reads 来实现对测序错误的纠正,CABOG 使用 rocks 和 stones 技术来实现对测序错误的纠正,Celera Assembly 使用 PacBioToCA 自纠算法来实现对测序错误的纠正.在对这 3 个算法进行领域分析后,找到了 3 个算法的共同特性和不同特性,归纳得到了 OLC 领域算法流程图和领域特征模型(见图 4 和图 5).

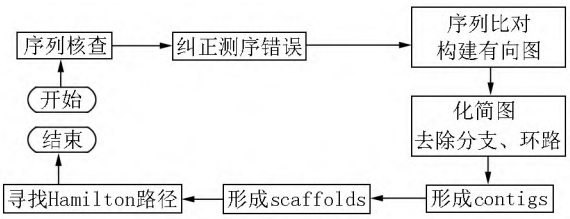

图 4　OLC 领域算法流程图

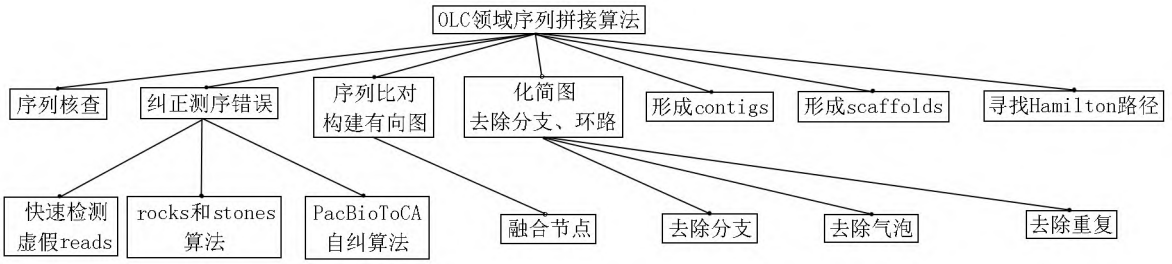

图 5　OLC 领域算法特征模型

2.2.2　**领域构件设计**　根据对 OLC 领域的分析和建模,可抽取出其中的特征,用参数表示其中的差异性,设计出可使用 Apla 语言描述的算法构件[29].在构建得到 ODO 算法后,测序所得的序列作为输入,算法将会对输入的序列做核查,避免由硬件问题而出现字符错误的现象发生,在纠正测序错误后,再进行序列比对,构建有向图,去除错误分支,生成 contigs 和 scaffolds.因此,这种 OLC 算法的主要构件是序列核查构件(Seq_check)、纠错构件(Error_correction)、序列比对构件(Sequence_alignment)、有向图生成构件(Graph_structure)、图化简去分支构件(Graph_simplification)、Contigs 拼接构件(Contigs_

assembly)、Scaffolds 拼接构件(Scaffolds_assembly)和最长路径构件(Longest_path).算法构件关系图如图 6 所示,这里使用 Radl 语言为 Seq_check 构件的功能做形式化描述.

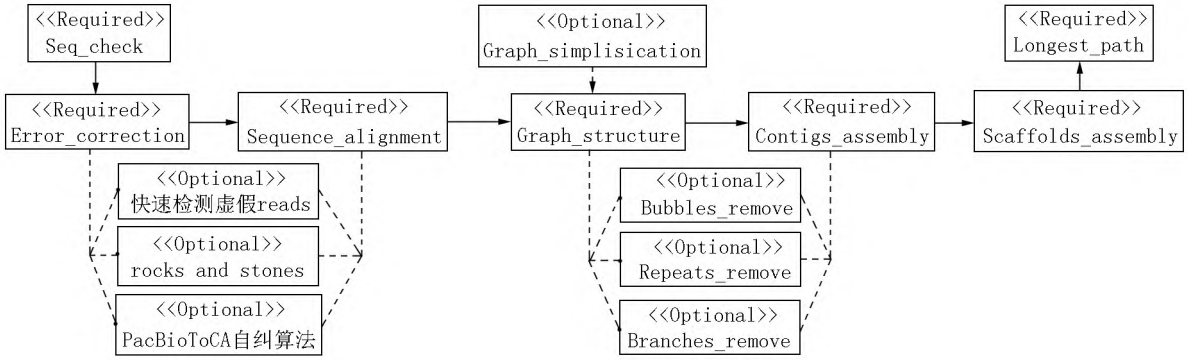

图 6 算法构件交互模型

Seq_check 构件:此构件对输入的基因序列进行核查,如在 DNA 基因序列中仅含有 4 种碱基(A,G,C,T),若序列中含有其他碱基,则会停止运行并提示错误.

| [in A_seq: list of char; B_seq: list of char; n: integer; out_flag: boolean] |;

AQ: $n=2 \wedge |A\_seq|>0 \wedge |B\_seq|=4 \wedge perm(B\_seq,\{A,G,C,T\})$;

(perm 表示标准核查序列 B_seq 和序列{A,G,C,T}互为置换);

AR: flag=($\forall i: 0\leqslant i\leqslant |A\_seq|: (\exists j: 0\leqslant j\leqslant 3: A\_seq[i]=B\_seq[j])$).

在上述的形式化规约中,输入(in)和输出(out)是在 PAR 平台中定义的 2 个关键字标识符,序列类型(list)、整型(integer)和布尔型(boolean)是在 PAR 平台中预定义的数据类型,AQ 和 AR 分别是构件的前置条件和后置条件.

2.2.3 Apla 形式化实现 抽象程序设计语言 Apla 可以直接使用抽象数据类型和抽象过程编写程序,可抽象描述算法,并进行正确性验证,这就保证了生成算法的可靠性,并且易于生成各种可执行程序设计语言程序 Java、C++等.在 Apla 中的标识符、关键字、符号表达方式和在 Radl 语言中涉及的一致.

在使用形式化方法开发图 6 中算法构件的 Apla 实现后,利用 PAR 平台 Apla→Java 程序生成系统得到对应的 Java 代码.

下面仅列出 Seq_check 构件的 Apla 实现.

```
function Seq_check(A_seq: list(char); B_seq: list(char,4)): boolean;
var i,j,sum: integer;
begin
i,j,sum: =0,0,0;
do (0<i<|A_seq|) →
    do (j<3) →if(A_seq[i]==B_seq[j]) →
        sum: =sum+1;
        fi;
        j: =j+1;
    od;
    i: =i+1;
od;
if(sum==|A_seq|) →Seq_check: =true
    [] →Seq_check: =false;
fi;
end.
```

再得到相应的 Java 程序如下:

```
public boolean Seq_check(String A_seq,String B_seq){
    int i=0;
    int j=0;
    int sum=0;
    for(i=0;i<A_seq.length();i++){
    for(j=0;j<3;j++){
    if(A_seq[i]==B_seq[j]){
    sum++;
            }
      }
    }
    if(sum == A_seq.length()){
    return true;
    }else{
    return false;
    }
}.
```

2.2.4 OLC 策略拼接算法组装 在构件设计完成后,根据使用需求和输入文件的特征可选择相对应的算法构件组装 ODO 算法.本文选择 Seq_check、Er-

ror _correction(rocks and stones) 、Sequence _alignment、Contigs_assembly 等构件构造出了 OLC_assembly_1 算法.按照相同的方法,从 OLC 和 DBG 领域算法构件库中选择符合使用需求的算法构件,生成 OLC_assembly_2 和 DBG_ assembly 算法,进一步组装得到 ODO 算法,由此将尽可能多的创造性劳动转化为非创造性劳动.

3　材料和实验

本次实验选择了 3 种单一策略(贪婪策略、OLC 策略和 DBG 策略) 的算法(SSAKE、Edena 和 Velvet) 与 ODO 构造算法进行对比实验.

3.1　实验材料

本文从 GenBank 中下载了 3 种基因组较小的样本来进行实验,实验样本信息如表 1 所示,基因名称分别是 *Clostridium perfringens*(产气荚膜梭菌) 、*Acetobacteraceti*(醋酸杆菌) 、*Escherichia coli*(大肠杆菌) ,该实验在如表 2 所示的实验平台上运行.

表 1　实验样本信息

基因名称	基因大小/M	碱基数/Mbp
Clostridium perfringens	6.1	4.0
Acetobacteraceti	3.0	2.9
Escherichia coli	2.8	2.7

表 2　实验运行平台

系统	Ubuntu 20.04.2 LTS 64bit
处理器	Intel ® Core™ i5-9500 CPU@ 3.00GHz
内存	4 G
图形处理器	llvmpipe (LLVM 11.0.0,256 bits)

3.2　coverage depth 和 k 值对序列拼接的影响

coverage depth 增加表示基因组复制的次数增加,经过鸟枪法(shot gun) 剪切,可得到更多不同的 reads,建立更完善的 Hash 表和更完整的有向图.但若无限的增加 coverage depth 的值,则过多的 reads 会导致文库过大,消耗大量的内存和计算时间,同时还会增加在有向图中的分支和错误节点,降低拼接准确度.本次实验将 coverage depth 的值设为 50.

Velvet 算法和 ODO 算法在构建 De Bruijn 图时会将文库中的 reads 切割成长度为 k 的 k-mer.k 值越大越容易得到唯一的序列,但在拼接过程中越容易产生 gaps; 随着 k 值减小,可以增加 De Bruijn 图的连通性,但会提高算法的复杂度,增加处理时间.同时,为了避免正反链混淆产生回文序列的现象发生,k 值应选奇数.本次实验所使用的 k 值取为 31.

3.3　实验结果与分析

构建了 OLC 和 DBG 领域构件库,生成了基于 OLC 的拼接算法 OLC_assembly_1、OLC_assembly_2 和基于 DBG 的拼接算法 DBG_ assembly,进一步组装出(OLC+DBG) →OLC 混合模式算法,简称 ODO 算法.为评估实验效果,在表 2 的实验平台上用 SSAKE(贪婪策略) 、Edena(OLC 策略) 、Velvet(DBG 策略) 和构造生成的 ODO 算法分别对在表 1 中的 3 种基因样本(*Clostridium perfringens*、*Acetobacteraceti*、*Escherichia coli*) 进行序列拼接实验.

由实验结果(见表 3) 可知: 构造的 ODO 算法无论是在 N50 大小上还是在基因组覆盖度(coverage depth) 上都优于其他的单一策略.如对于 *Clostridium perfringens* 这一样本基因,ODO 算法拼接出来的结果 N50 为 1 625 bp,基因覆盖度为 93.9%,优于单一策略进行序列拼接的结果.

表 3　4 种策略算法在不同样本上的拼接结果

基因名称	评估指标	SSAKE	Edena	Velvet	ODO
Clostridium perfringens	N50/bp	693	639	1 179	1 625
	Contigs number	165	132	286	106
	Longest of contigs/bp	730	775	4 200	4 979
	Coverage/%	87.3	91.2	92.7	93.9
Acetobacteraceti	N50/bp	432	568	854	1 135
	Contigs number	106	87	54	67
	Longest ofcontigs/bp	446	593	1 169	1 644
	Coverage/%	93.4	92.3	94.8	95.1
Escherichia coli	N50/bp	328	545	673	894
	Contigs number	634	384	268	254
	Longest of contigs/bp	354	693	746	1 065
	Coverage/%	89.6	94.2	94.6	97.1

在评估指标中 N50 表示在拼接得到的 Contigs 长度从大到小排列后最中间 Contigs 长度的值,N50 的值越大说明拼接得到的 Contigs 长度越长,即拼接效果越好.在对 *Escherichia coli* 实验样本进行拼接时,SSAKE、Edena、Velvet 和 ODO 得到的 N50 值分别为 328 bp、545 bp、673 bp 和 894 bp.由实验结果可知:相比于其他的单一策略下的算法,ODO 算法取得了较优的结果.产生这一结果的主要原因有 2 个方面:(i)先对输入的 reads 文库进行 2 次拼接,然后将 2 次拼接的结果混合再次作为第 3 阶段拼接算法的输入,得到最终的 Contigs 输出;(ii)在 ODO 算法的构造过程中使用形式化方法开发算法领域构件,可根据用户的需求和输入文件的特征选择不用的构件组装得到策略拼接算法,在提高算法准确度和效率的同时,保证了结果的可靠性.

在评估指标中 Contigs number 表示拼接结果的 Contigs 数量,在基因组碱基数量大致相同的情况下,Contigs 数量越少表示每条 Contigs 的长度越长,即拼接结果越好.在对 *Acetobacteraceti* 实验样本进行拼接时,SSAKE、Edena、Velvet 和 ODO 得到的 Contigs number 值分别为 106、87、54 和 67.由评估指标可知:ODO 算法的结果比 SSAKE 和 Edena 算法的结果更优,但是离 Velvet 算法的结果还有一定的差距.产生这一结果的可能原因是:在序列中还有较多的 gaps,当碱基比对遇到 gap 时算法会默认当前匹配不成功,并停止匹配,输出结果,这将导致 Contigs number 反向增大.

4 结论

本文利用形式化方法及形式化平台方面的优势,结合领域分析建模和产生式编程的方法,形式化开发了序列拼接算法构件,并构造了(OLC+DBG)→OLC 混合模式算法,简称 ODO 算法.实验结果显示:在对 3 种基因样本进行序列拼接对比实验时,构造生成的 ODO 算法比单一策略算法所取得的结果在 N50 和 Coverage 等参数上有一定的优势.ODO 算法在 *Escherichia coli* 样本的序列拼接实验中基因组覆盖度达到 97.1%,N50 的值达到 894 bp.研究结果表明:利用产生式编程和形式化方法开发的 ODO 算法在综合不同策略优势、提高拼接效率和准确度的同时也保证了结果的可靠性,为深入研究算法构造在序列拼接上的影响提供了参考.

5 参考文献

[1] YANG Huanming.*MEOMIC*(medicine in omics) and the HGP(human genome project)[J].Medicine in Omics,2021,1:100004.

[2] SANGER F,NICKLEN S,COULSON A R.DNA sequencing with chain-terminating inhibitors[J].Proceedings of the National Academy of Sciences of the United States of America,1977,74(12):5463-5467.

[3] MAKHLUF H,BUCK M D,KING K,et al.Tracking the evolution of dengue virus strains D2S10 and D2S20 by 454 pyrosequencing[J].PLoS One,2013,8(1):e54220.

[4] FEDURCO M,ROMIEU A,WILLIAMS S,et al.BTA,a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies[J].Nucleic Acids Research,2006,34(3):e22.

[5] ONDOV B D,VARADARAJAN A,PASSALACQUA K D,et al.Efficient mapping of applied biosystems SOLiD sequence data to a reference genome for functional genomic applications[J].Bioinformatics,2008,24(23):2776-2777.

[6] 张丁予,章婷曦,王国祥.第二代测序技术的发展及应用[J].环境科学与技术,2016,39(9):96-102.

[7] THOMPSON J F,STEINMANN K E.Single molecule sequencing with a Heli Scope genetic analysis system[J].Current Protocols in Molecular Biology,2010,92(1):10-36.

[8] CLARKE J,WU Haichen,JAYASINGHE L,et al.Continuous base identification for single-molecule nanopore DNA sequencing[J].Nature Nanotechnology,2009,4(4):265-270.

[9] MUNROE D J,HARRIS T J R.Third-generation sequencing fireworks at Marco Island[J].Nature Biotechnology,2010,28(5):426-428.

[10] 范佳雯.高通量测序数据识别拼接错误方法[J].电子技术与软件工程,2019(23):146-147.

[11] 俞晓玲,姜文倩,郑玲,等.单分子测序技术及应用研究进展[J].生物化学与生物物理进展,2020,47(1):5-16.

[12] 蒋帅,周永坤,王涛.启发式搜索思想在路径规划中的应用[J].指挥信息系统与技术,2021,12(4):57-63.

[13] 张旭初.多序列比对算法族的构件组装研究[D].南昌:江西师范大学,2020.

[14] WARREN R L,SUTTON G G,JONES S J M,et al.Assembly millions of short DNA sequences using SSAKE[J].Bioinformatics,2007,23(4):500-501.

[15] JECK W R,REINHARDT J A,BALTRUS D A,et al.Extending assembly of short DNA sequences to handle error[J].Bioinformatics,2007,23(21):2942-2944.

[16] DOHM J C,LOTTAZ C,BORODINA T,et al.SHARCGS,a fast and highly accurate short-read assembly algorithm for

de novo genomic sequencing [J]. Genome Research, 2007,17(11): 1697-1706.

[17] CAO M D,NGUYEN S H,GANESAMOORTHY D,et al. Scaffolding and completing genome assemblies in real-time with nanopore sequencing [J]. Nature Communications, 2017,8(1): 14515.

[18] ZIMMERMANN T. Description of a genome assembler: CABOG [EB/OL]. [2021-11-02]. https://www.theozimmermann.net/pdf/presentation-CABOG.pdf.

[19] GORDON D. Viewing and editing assembled sequences using Consed [EB/OL]. [2021-11-02]. https://currentprotocols.onlinelibrary.wiley.com/doi/10.1002/0471250953.bi1102s02.

[20] MYERS E W,SUTTON G G,DELCHER A L,et al. A whole-genome assembly of *Drosophila* [J]. Science,2000, 287(5461): 2196-2204.

[21] HERNANDEZ D,FRANÇOIS P,FARINELLI L,et al. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer [J]. Genome Research,2008,18(5): 802-809.

[22] LI Ruiqiang,ZHU Hongmei,RUAN Jue,et al. De novo assembly of human genomes with massively parallel short read sequencing [J]. Genome Research,2010,20(2): 265-272.

[23] ZERBINO D R,BIRNEY E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs [J]. Genome Research,2008,18(5): 821-829.

[24] SIMPSON J T,WONG K,JACKMAN S D,et al. ABySS: a parallel assembler for short read sequence data [J]. Genome Research,2009,19(6): 1117-1123.

[25] PEVZNER P A,TANG Haixu,WATERMAN M S. An Eulerian path approach to DNA fragment assembly [J]. Proceedings of the National Academy of Sciences of the United States of America,2001,98(17): 9748-9753.

[26] BUTLER J,MACCALLUM I,KLEBER M,et al. ALLPATHS: de novo assembly of whole-genome shotgun microreads [J]. Genome Research,2008,18(5): 810-820.

[27] KUSUMA W A,ISHIDA T,AKIYAMA Y. A combined approach for de novo DNA sequence assembly of very short reads [J]. IPSJ Transactions on Bioinformatics,2011, 4(10): 21-23.

[28] 石海鹤. 形式化框架下置换和查找类算法的组装生成 [M]. 北京: 科学出版社,2017: 25-27.

[29] SHI Haihe,WU Gang. Gene sequence assembly algorithm model based on the DBG strategy and its application [J]. Journal of Healthcare Engineering,2021,2021: 6676194.

The Construction of de novo Sequence Assembly Algorithm Based on Hybrid Strategy

XIAO Cunwei,SHI Haihe*,WANG Lan,CHENG Bailiang

(School of Computer and Information Engineering,Jiangxi Normal University,Nanchang Jiangxi 330022,China)

Abstract: Based on the analysis of three basic strategies of de novo sequence assembly,namely greedy strategy,OLC (Overlap-Layout-Consensus) strategy and DBG(De Bruijn Graph) strategy,the construction process of hybrid strategy sequence assembly algorithm is studied,so as to integrate the advantages of multiple single strategies.Taking advantage of the team's advantages in formal methods and platforms,combined with the methods of domain analysis modeling and generative programming,two algorithms based on OLC strategies (OLC_assembly_1,OLC_assembly_2) and an algorithm based on DBG (DBG-assembly) strategies are constructed,and the algorithms in the (OLC+DBG)→OLC hybrid mode (referred to as ODO algorithms) are further assembled.Finally,three experimental samples are selected from GenBank,and the stitching results of the algorithm and ODO construction algorithm under three single strategies are compared from the perspectives of N50,Contigs number,Coverage,etc.,and the effect of cover depth and k value change on the stitching result is analyzed.Experimental results show that the ODO algorithm implemented in this paper has certain advantages over the results of sequence assembly in terms of parameters such as N50 and Coverage.

Key words: de novo sequences assembly; mixed strategy; domain feature modeling; generative programming; formal method

(责任编辑: 冉小晓)