基因序列相似程度的 LCS 算法研究

王映龙[1,2],杨炳儒[1],宋泽锋[1],陈 卓[1],唐建军[2]
WANG Ying-long[1,2],YANG Bing-ru[1],SONG Ze-feng[1],CHEN Zhuo[1],TANG Jian-jun[2]

1.北京科技大学 信息工程学院,北京 100083
2.江西农业大学 计算机与信息工程学院,南昌 330045
1.School of Information Engineering,University of Science and Technology Beijing,Beijing 100083,China
2.School of Computer and Information Engineering,Jiangxi Agriculture University,Nanchang 330045,China
E-mail:wangyinglongdl@sohu.com

WANG Ying-long,YANG Bing-ru,SONG Ze-feng,et al.LCS algorithm research for gene sequence similar degree. Computer Engineering and Applications,2007,43(31):45-47.

Abstract: This paper first re-examines the difficulty of using the exhaustion method to solve the LCS question,as well as corresponding advantages,and then aims at merit of the exhaustion method to present two types of optimizations,finally,produces the algorithm achievement and algorithm conclusions.The experiment proves that the algorithm efficiency than conventional dynamic programming LCS algorithm has been greatly improved.

Key words: Longest Common Subsequence(LCS);exhaustion method;gene sequence analyzing

摘 要: 首先重新审视了采用穷举法求解 LCS 问题的困难,以及对应的优点;随后针对穷举法的优点进行了两类优化;最后给出了算法实现的图示以及算法的结论。通过实验证明,算法的效率较传统的动态规划的 LCS 算法有了很大的提升。

关键词: 最长公共子序列;穷举法;基因序列排比
文章编号: 1002-8331(2007)31-0045-03 **文献标识码:** A **中图分类号:** TP391;TP309

1 前言

LCS 是 Longest Common Subsequence 的缩写。LCS 算法是一种非常基础的算法。其主要作用是找出两个序列中最长的公共子序列。目前 LCS 算法有着非常广泛的应用,在图形相似处理[1,2]、媒体流的相似比较[1]、计算生物学(computational biology)[2,3]等等研究方向上都有应用。

目前与 LCS 算法相关的论文已经从很多方面对求解两个序列的 LCS 进行了优化与算法分析。但基本上都是对两个序列在组成序列的元素无法穷举的条件下进行研究。也就是说,不考虑待比较的序列本身的特点。例如在解决基因的序列排比等等方面的问题时,组成序列的元素是固定的(由 ACTG 4 个元素组成),而并没有针对特性进行仔细的分析与算法的优化。所以目前研究出的 LCS 算法, 优点是适合于所有序列进行比较,但缺点是针对元素可穷举的特殊的序列组合并不一定是效率与方法最高。还有,在很多具体的研究中,都要求快速地从很多的序列中找出最相似的两个序列,也就需要有一些快速算法先将要计算的队列减少。将"非常不相似"的序列经过预处理后先去除。如果采用传统的 LCS 算法来进行这样的预处理过程,将是非常困难的。

2 背景知识

2.1 LCS 算法的定义

令 $S_1=S_1(1)S_1(2)\cdots S_1(n)$和 $S_2=S_2(1)S_2(2)\cdots S_2(m)$为两个序列片段;$S_K(i,j)$表示 $S_K(i)S_K(i+1)\cdots S_K(j)$的子字符串(其中 K=1 或 2)。则此两序列的 LCS 定义如下:

定义 1 S_1和 S_2的 LCS 是指共同存在于 S_1及 S_2中的最长子序列,且此子序列不一定要连续地出现在 S_1或 S_2,只要出现的顺序和在 S_1和 S_2内出现的顺序相同即可。令 $LLCS(S_1,S_2)$表示 S_1和 S_2的 LCS 长度[5,6]。

例如:令 S_1=AAGTACC,S_2=ATTACCT,则 $LCS(S_1,S_2)$为 ATACC,且 $LLCS(S_1,S_2)$为 5。

传统上,LCS 的解法是采用动态规划(Dynamic Programming)算法进行解决。其解法往往可以用一个递归函数(Recursive Function)来表示,如下

$$M(i,j)=\begin{cases}\max\begin{cases}M(i-1,j)\\M(i,j-1)\end{cases} & \text{if } S_1(i)\neq S_2(j)\\ M(i-1,j-1)+1 & \text{if } S_1(i)=S_2(j)\end{cases}$$

基金项目: 国家自然科学基金(the National Natural Science Foundation of China under Grant No.60675030);教育部科技重点项目(No.教技司[2000]175)。

作者简介: 王映龙,男,副教授,博士研究生,研究方向为数据挖掘;杨炳儒,男,教授,博士生导师,研究方向为知识发现与智能系统,柔性建模与集成技术;宋泽锋,男,博士研究生,研究方向为数据挖掘;陈卓,男,博士研究生,研究方向为数据挖掘;唐建军,男,博士研究生,副教授,研究方向为控制理论。

其中,$M(i,j)$表示 $S_1(1,i)$和 $S_2(1,j)$之间的 LCS 长度。所以,$M(n,m)=LLCS(S_1,S_2)$。以 S_1=AAGTACC 和 S_2=ATTACCT 为例,其矩阵 $\boldsymbol{M}$ 的计算结果如图 1 所示。从这个矩阵可以得知 $LCS(S_1,S_2)$的长度为 $M(4,4)=3$。

		0	1	2	3	4
	S_2	S_1				
			G	A	G	T
0		0	0	0	0	0
1	A	0	0	1	1	1
	G	0	1	1	2	2
2	A	0	1	2	2	2
3	T	0	1	2	2	3

图 1 LCS 矩阵 $\boldsymbol{M}$

若要计算的两序列 S_1和 S_2的长度都为 n,则计算 $LCS(S_1,S_2)$所需的时间为 $O(n^2)$。当 n 的值越来越大如趋近百万时,$O(n^2)$的时间复杂度在实际上将形成一个庞大的负担。那么采用动态规划的方法来解决这个问题就不合适。所以在下一节,先重新回到采用穷举法解决 LCS 问题老路上,分析穷举法的困难与优势。然后再针对元素可穷举的特点对穷举法进行优化。

2.2 针对元素可穷举序列的 LCS 穷举法

针对元素可穷举序列的 LCS 穷举法的思路从以下定理开始:

定理 1 令 $S_1=S_1(1)S_1(2)\cdots S_1(n)$和 $S_2=S_2(1)S_2(2)\cdots S_2(m)$为两个序列片段,属于 S_1的元素组成了 $Q(S_1)$,属于 S_2的元素组成了 $Q(S_2)$,集合 $Q=Q(S_1)\cap Q(S_2)$。则 $LCS(S_1,S_2)$的元素一定属于集合 Q。

定理 2 令 $S_1=S_1(1)S_1(2)\cdots S_1(n)$和 $S_2=S_2(1)S_2(2)\cdots S_2(m)$为两个序列片段。若某一个 $LCS(S_1,S_2)$的最后一个元素为 z,n'为 S_1中从 n 开始找到的第一个 z 的位置减 1,如果没有找到 z 则 $n'=0$,$S_1'=S_1(1)S_1(2)\cdots S_1(n')$。$m'$为 S_2中从 m 开始找到的第一个 z 的位置减 1,如果没有找到 z 则 $m'=0$,$S_2'=S_2(1)S_2(2)\cdots S_2(m')$。则此 $LCS(S_1',S_2')+z=LCS(S_1,S_2)$。

根据以上两点:$LCS(S_1,S_2)$中元素位置不会在(S_1'',S_2'')、$LCS(S_1,S_2)$中最后一个元素是 z。可以得到结论 $LCS(S_1',S_2')+z=LCS(S_1,S_2)$。

求解 LCS 问题的穷举法就是从定理 1 与定理 2 出发提出的一种算法。由于 $LCS(S_1,S_2)$的元素也已经固定。那么穷举法就是从 $LCS(S_1,S_2)$的结果开始,将 $LCS(S_1,S_2)$从第一个位置开始进行穷举。以基因序列为例来说明针对元素可穷举序列的 LCS 穷举法(为了叙述的方便,以下所说的序列如不特殊说明,都是特指仅由 ACTG 4 个元素组成的基因序列)。由于 $LCS(S_1,S_2)$的每一个元素一定是(A,T,C,G)中的一个。运用定理 2,每次可以假设所有的元素都可能是 $LCS(S_1,S_2)$的最后一个元素,循环地进行计算,直至能得到最后 LCS 结果。

可以看出,穷举法的效率是很低的,当 S_1'、S_2'还没有出现空集时,每一次都要计算 4 次,也就是计算次数几乎要达到 $4^{LLCS(S_1,S_2)}$次。比采用动态规划的效率要低很多。这是穷举法的困难,但是穷举法也有一个优势,上面的描述中可以看出,每一次计算后,可以确认已经计算出的 LLCS。充分利用此特性,想办法在每一次计算后减少下一次要计算的内容,以进行算法优化。下面给出一种基于穷举法的优化 LCS 算法,就是基于这个思想设计的。

2.3 基于穷举法的优化 LCS 算法

从上可以看出,穷举法存在大量的重复计算。比如(CG,"")这个序列队出现了多次。同样的问题在动态规划算法中也存在。优化的思路就是减少重复计算。下面首先给出定理 3。并在此定理的基础上给出基于穷举法的优化 LCS 算法[4,7,8]。

定理 3 令 $S_1=S_1(1)S_1(2)\cdots S_1(n)$和 $S_2=S_2(1)S_2(2)\cdots S_2(m)$为两个序列片段,$n\geqslant n_1\geqslant 0$;$m\geqslant m_1\geqslant 0$。$S_1'=S_1(1)S_1(2)\cdots S_1(n_1)$,$S_2'=S_2(1)S_2(2)\cdots S_2(m_1)$。则:

$$LLCS(S_1,S_2)\geqslant LLCS(S_1',S_2')$$

经过定理 3 的筛除,计算完成的 7 个序列对最后只剩下 2 个(GACG,GTA)、(G,AGTA),那么进入下一次计算时,就会大大缩少要计算的次数,从而提升算法效率。如图 2 穷举法与优化后的穷举法计算量比较是针对上节中的 S_1、S_2,分别采用穷举法与优化后的穷举法。在每一步所需要计算的序列对的个数。可以看出,效率得到了很大的提升。

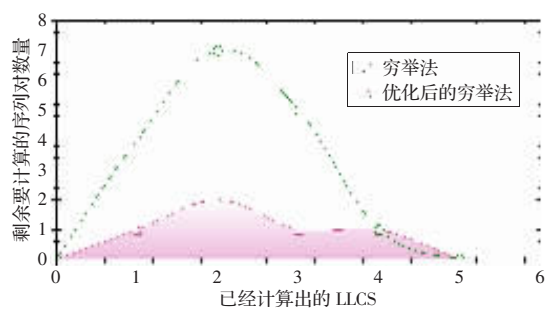

图 2 穷举法与优化后的穷举法计算量比较

因为应用定理 3 能够保证不会进行重复性的计算,所以在不考虑每一步计算之后删除多余序列对的过程情况下,效率会比动态规划的算法有明显提升。

3 快速确定基因序列相似性的 LCS 算法

问题的提出:如何能够快速地判断两个长度都为 n 序列的 LLCS 是否大于给定的数据 p。例如快速确定两个长度为 1 000 的基因序列其 LLCS 是否大于 800?

3.1 快速确定基因序列相似性的 LCS 算法

提高 LCS 穷举算法效率,就是在每一次计算出一批可能的 LCS 与剩余的序列对时,尽可能地减少下一次要计算的序列对。上一节中的算法是基于一种最简单定理进行的算法优化。本节通过给定的条件来进一步的减少中间计算的次数。

定理 4 令 $S_1=S_1(1)S_1(2)\cdots S_1(n)$和 $S_2=S_2(1)S_2(2)\cdots S_2(n)$为两个序列片段,长度都为 n。若其 $LLCS(S_1,S_2)>p$,则 $LCS(S_1,S_2)$中的任一点 X,X 在 S_1中的位置为 a,X 在 S_2中的位置为 b,必有$|a-b|<n-p$。

通过定理 4,实际上可以在采用穷举法计算两个序列 LCS 时多加入一个约束条件。此约束条件使得在完成一步计算后,进入下一步计算之前,将两个坐标的绝对值相差超过 $n-p$ 的序列去掉,以达到提高效率的目的。具体的实现方法此处就不再详述了。

3.2 快速确定基因序列相似性的 LCS 算法的实验数据

为了验证算法实际的效果，找了两组基因数据进行了演算，以验证快速 LCS 穷举法与快速确认相似的优化 LCS 算法效率上的差别。找了两组长达 1 K 的数据,分别采用优化的穷举法,以及快速确定基因序列相似度(设置范围为 $LLCS(S_1, S_2)\geqslant 200$)两种算法,每次计算后已计算出的 LLCS 与下一步要计算的序列对的关系如图 3 穷举法与优化后的穷举法 1 K 数据计算量比较图所示:

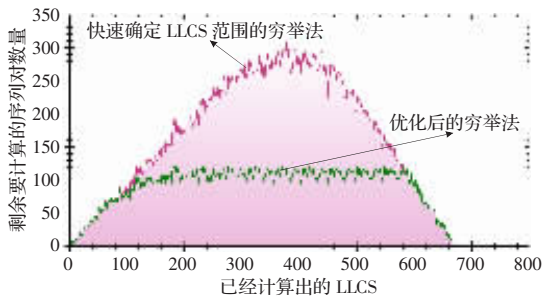

图 3 穷举法与优化后的穷举法 1 K 数据计算量比较

可以看出，快速确定 LCS 范围的穷举法计算的次数较优化后的穷举法减少了很多,效率得到提高。

4 结论与下一步研究

LCS 穷举法好处是每一步计算之后,已计算出的 LLCS 是固定值。根据此特性去设计算法,针对不同的应用,来减少下一步计算时的序列对。无论是优化后的穷举法,或是快速确定范围的 LCS 算法,是在这个方面进行了优化与再次优化。那么下一步的研究,主要还是针对这个方面进行再次的优化。

本算法是从基础上对 LCS 算法进行优化，所以针对 LCS 的并行算法以及其它方面的一些算法优化研究可以利用本算法的结论进行优化。

能够快速地确认序列之间的相似性,所以本算法可以应用于互联网的数据压缩与图像搜索等方面。

(收稿日期:2007 年 7 月)

参考文献:

[1] Yazdani N,Ozsoyoglu Z M.Sequence matching of images[C]//Proceedings of the IEEE International Conference on Multimedia Computing and Systems,Volume II,1996:53-62.

[2] Hunt J W,Szymanski T G.A fast algorithm for computing longest common subsequences[J].Communications of the ACM,1977,20(5):350-353.

[3] Sutinen E,Tarhio J.Approximate string matching with ordered q-grams[J].Nordic Journal of Computing,2004,11(4):321-343.

[4] Setubal,Meidanis J.Introduction to computation molecular biology.University of Campinas,Brazil,1997.

[5] Apostolico,Guerra C.The Longest Common Subsequence problem revisited[J].Algorithmica,1987(2):315-336.

[6] Deken.Some limit results for longest common subsequences[J].Discrete Mathematics,1979,26:17-31.

[7] Gusfield.Algorithms on strings,trees,and sequences:computer science and computational biology[D].Cambridge:Cambridge University Press,1997.

[8] Kececioglu,Sankoff D.Exact and approximation algorithms for the inversion distance between two permutations[J].Algorithmica,1995,13:180-210.

(上接 44 页)

在表 2 中,从连续单峰函数 $f1$ 和 $f2$ 可以看出,新算法均取得最好的平均适应值和最好适应值，与其他 3 种算法相比,新算法具有较快的收敛速度和较高的解精度。从非线性多峰函数 $f3$ 和 $f4$ 可以看出,新算法在保持群体多样性、全局搜索性能、逃离局部极值和避免早熟收敛能力方面均优于其他 3 种算法。

图 1 中的变化曲线更加清晰地表明了，新算法解精度高,容易逃离局部极值,与其他 3 种算法相比具有最佳的优化性能。

6 结论

本文借鉴生物进化规律,提出一种具有综合学习机制的粒子群算法。一方面,新算法通过用所有粒子的个体极值的平均值取代每一个粒子自身的个体极值,很好地利用了粒子间有益的信息,加强全局搜索,避免早熟收敛;另一方面,新算法通过自适应地改变接受概率,在搜索早期,使接受比自身更优粒子的个体极值作为全局极值占主导地位，以确保种群的多样性,避免过早收敛;在搜索后期,使接受群最优粒子的个体极值作为全局极值占主导地位,以提高解精度,从而较好地平衡了新算法在搜索过程中全局探测和局部开发之间的能力。实验结果显示,新算法在上述单峰和多峰基准函数优化中,显示出全局搜索能力强,求解精度高,收敛速度快,稳定好等特点。

(收稿日期:2007 年 3 月)

参考文献:

[1] Shi Y,Eberhart R C.A modified particle swarm optimizer[C]//IEEE International Conference of Evolutionary Computation,Anchorage,Alaska,May 1998.

[2] Clerc M.The swarm and the queen:towards a deterministic and adaptive particle swarm optimization[C]//Proceedings of the Congress on Evolutionary Computation.Piscataway,NJ:IEEE Service Center,1999:1951-1957.

[3] 王存睿,段晓东,刘向东,等.改进的基本粒子群优化算法[J].计算机工程,2004,30(21):35-37.

[4] 李宁,付国江,库少平,等.粒子群优化算法的发展与展望[J].武汉理工大学生学报:信息与管理工程版,2005,27(2):26-28.

[5] Shi Y,Eberhart R C.Empirical study of particle swarm optimization[C]//Proceeding of Congress on Evolutionary Computation.Piscataway,NJ:IEEE ServiceCenter,1999:1945-1949.

[6] Peram T,Veeramachaneni K.Fitness-distance-ratio based particle swarm optimization[J].IEEE Swarm Intelligence Symp,2003:174-181.

[7] Liang J J,Qin A K.Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J].IEEE Trans Evol Comput,2006:281-295.

[8] Beekman M,Ratnieks F L W.Long-rang foraging by the honey-bee,apis mellifera L[J].Functional Ecologicy,2000(14):490-496.