基于格代数的最长公共子序列近似求解

孙 焘 朱晓明
(大连理工大学创新创业学院 辽宁 116024)

摘 要 多条序列的最长公共子序列可以代表多条序列的公共信息,其在诸多领域里有着重要的应用,如信息检索、基因序列匹配等。求解多条序列的最长公共子序列是著名的NP难问题,本质为多解问题。一些近似算法虽然时间复杂度较低,但只能求出单解,对于有多解的序列集合,求得的结果信息量损失较大。因此提出一个新的近似算法来解决最长公共子序列问题。算法引入了代数结构"格",通过动态规划求解出两条序列的公共格,并递归求解当前格与当前序列的公共格。公共格中的路径保存了多条公共子序列使得最终求解出的最长公共子序列为多个。对算法的相关定理给出了理论证明,并通过实验验证了算法的正确性。

关键词 最长公共子序列,格,近似算法,贪心算法
中图法分类号 TP391.4 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.02.045

Computing Longest Common Subsequences Approximately Based on Lattice

SUN Tao ZHU Xiao-ming
(School of Innovation and Entrepreneurship, Dalian University of Technology, Liaoning 116024, China)

Abstract The longest common subsequences can represent the common information of a sequence set, and it has many important applications in many fields, such as computational genomics, information retrieval and so on. Computing longest common subsequences is a famous NP-hard problem with multiple solutions. Some approximate algorithms have a low time complexity, but the result set only has one subsequence. For the sequence set with many longest common subsequences, the information loss is overmuch. In this paper, we presented a new approximate algorithm for this problem. Our algorithm employs a specific mathematical structure called lattice. Firstly, the common lattice of two sequences is computed through dynamic programming and then the common lattice of the current lattice and the current sequence recursively combined with the greedy algorithm are also computed. As the paths in a common lattice contain many common subsequences, the result set contains many longest common subsequences of the original sequence set. And the validity of our algorithm is proved through theory and experiment.

Keywords Longest common subsequences (LCS), Lattice, Approximate algorithm, Greedy algorithm

1 引言

计算多条序列的最长公共子序列问题(LCS)是著名的NP难问题[1]。LCS问题可以表述如下:对于给定的 d 条序列集合 $S=\{s_1,s_2,\cdots,s_d\}$,若一条序列是集合 S 中所有序列的子序列,那么这条序列称为公共子序列(CS),最长的公共子序列被称作最长公共子序列(LCS)。最长公共子序列在基因工程领域与信息检索等领域有着重要的应用[3-4,8-9]。

对这个问题已经做了很多的研究[2],但是当数据集规模增大时,精确算法计算最长公共子序列所需要的时间依旧有明显增加。近似算法计算最长公共子序列的时间复杂度可以保持在比较低的量级,但只能求出单解。集合中序列的最长公共子序列可以提供当前集合中序列特征的公共信息,最长公共子序列的数量越多,可以提供的公共信息也越多。这意味着当前集合的最长公共子序列数量越多,则单解算法所求的结果信息量损失越大。

因此本文提出一个新的近似算法——格方法(lattice-approach)来计算多条序列的最长公共子序列。该算法基于传统贪心策略,通过动态规划得到的生成矩阵中保存的信息是一个格结构,使用格结构来保存更多有关于两条序列的最长公共子序列的信息,并递归地计算当前格与当前序列的公共格,直到集合中的序列都被计算过,算法结束。在得到的公共格中,寻找最长路径所对应的序列即为序列集合的最长公共子序列。由于格中具有多条路径,因此算法所得的解也为多个。

本文第2节给出相关工作,介绍一些计算最长公共子序列的近似算法;第3节提出了本文算法,并且介绍本文算法的原理、相关概念以及相关定理的证明;第4节通过实验来检验本文算法的效果;最后总结全文。

到稿日期:2016-01-23 返修日期:2016-05-16
孙 焘(1975—),博士,副教授,主要研究方向为数据挖掘、时间序列分析;朱晓明(1990—),硕士,主要研究方向为数据挖掘。

2　相关工作

2.1　贪心算法

对于给定的多条序列的集合，其中的两条序列记为 s_i 与 s_j，把求 s_i 与 s_j 最长公共子序列的过程记为 $LCS(s_i,s_j)$。贪心算法[5]的两个策略分别表述如下。

策略 1　把 LCS 操作得到的最长公共子序列与下一个序列再次做 LCS 操作，直到给定的序列集合中的全部序列都已经被计算，算法结束。策略 1 的贪心算法过程的表述如下：

$$Greedy(s_1,s_2,\cdots,s_d)=LCS(LCS(LCS(s_1,s_2),\cdots,s_{d-1}),s_d)$$

策略 2　将集合中每两条序列做 LCS 操作得到它们的最长公共子序列，再对每两条最长公共子序列做 LCS 操作，直到只剩一条最长公共子序列，算法结束。策略 2 的贪心算法过程的表述如下：

$$Greedy(s_1,s_2,\cdots,s_d)=LCS(LCS(s_1,s_2),\cdots,LCS(s_{d-1},s_d))$$

贪心算法为近似求解最长公共子序列的基本算法。

2.2　Rizvi-Agarwal 算法

Rizvi-Agarwal 算法[6]以启发式方法为基础。首先，建立若干对于给定字符集 Σ 的集合，称作“桶”(bucket)，桶的数量由字符集 Σ 中的字符数量决定。之后，对于每个桶中给定序列集合中的每条序列，将每个在字符集 Σ 中出现的字符在序列中出现的位置记录在桶中，并通过评估函数计算 X,Y,Z，找到最小的 Z 所对应的字符，把这个字符加在当前所求的最长公共子序列的末尾，并以这个字符作为基准，对各个桶中的数值进行更新操作。重复上述步骤，直到其中某一个桶中的数值耗尽，算法结束。评估函数如下：

$$\begin{cases} X_C=B_C[r][s_1]+B_C[r][s_2]+\cdots+B_C[r][s_d] \\ Y_C=|B_C[r][s_1]-B_C[r][s_2]|+|B_C[r][s_2]- \\ \quad B_C[r][s_3]|+\cdots+|B_C[r][s_{d-1}]-B_C[r][s_d]| \\ Z_C=X_C+Y_C \end{cases}$$

得到的最长公共子序列即为所求的近似解。

2.3　相关位置算法

相关位置算法[5]以启发式方法为基础，基于 Rizvi-Agarwal 算法的评估函数，对 X,Y,Z 的值进行计算，并将 Z 中最小值对应的字符加入到当前所求的最长公共子序列的末尾。同时，将各条序列中第一次出现该字符之前的字符(包括该字符)全部标记为已用。递归地重复上述步骤对剩下的序列进行计算，直到某一条序列耗尽，算法结束。得到的最长公共子序列即为所求。

相关位置算法在 Rizvi-Agarwal 算法公式的基础上增加了剪枝操作，以减少计算量，提高算法的计算速度。

3　本文方法

本文介绍一个新的计算最长公共子序列的近似算法，称为格方法(lattice-approach)。格方法基于动态规划算法与贪心算法，引入了代数结构格(lattice)，通过格结构来尽可能多地保存每步贪心过程中的动态规划信息，使结果集中的序列数量与精确度均比传统近似算法有一定的提升。

3.1　相关概念与定义

考察两条序列 $\boldsymbol{x}(x_1,x_2,x_3,\cdots,x_m)$ 与 $\boldsymbol{y}(y_1,y_2,y_3,\cdots,y_n)$ 的最长公共子序列(LCS)，其中 $x_i,y_j\in\Sigma(i=1,\cdots,m;j=1,\cdots,n)$，$\Sigma$ 为指定的字符集。为后续理论表述方便且不失一般性，可设 $x_1=y_1=\alpha,x_m=y_n=\beta$(其中 $\alpha\neq\beta;\alpha,\beta\notin\Sigma$)。$\Sigma'=\Sigma\cup\{\alpha,\beta\}$。用 $|\cdot|$ 表示集合的元素个数。

定义 1(代数结构 $\mathbf{L}(U,\prec^{(0)})$)　给定两条序列 $\boldsymbol{x}(x_1,x_2,x_3,\cdots,x_m)$ 与 $\boldsymbol{y}(y_1,y_2,y_3,\cdots,y_m)$。设 U 为形式为 $(\mu,\omega,\lambda,\sigma)$ 的四元组集合，且为 $\{1,2,\cdots,m\}\times\{1,2,\cdots,n\}\times\{1,2,\cdots,|U|\}\times\Sigma'$ 上的子集，其中 U 中元素满足 $\boldsymbol{u}(1,1,1,\alpha)\in U,\boldsymbol{u}(m,n,|U|,\beta)\in U$。

记 $\boldsymbol{u}.\mu,\boldsymbol{u}.\omega,\boldsymbol{u}.\lambda,\boldsymbol{u}.\sigma$ 分别为元素 $\boldsymbol{u}$ 的 4 个分量，其中 μ 与 ω 为元素 $\boldsymbol{u}$ 对应序列 $\boldsymbol{x}$ 与序列 $\boldsymbol{y}$ 中的元素下标，λ 为 $\boldsymbol{u}$ 在集合 U 中的下标，σ 为元素 $\boldsymbol{u}$ 上的字符。

对于 $\forall\boldsymbol{u}(i,j,k,v)\in U$ 有 $x_i=y_j=v(v\in\Sigma')$。令“$\prec^{(0)}$”为 U 上的二元关系，满足对任意 $\boldsymbol{u}(i_a,j_a,k_a,v_a)\in U$ 与 $\boldsymbol{u}(i_b,j_b,k_b,v_b)\in U$，若 $\boldsymbol{u}(i_a,j_a,k_a,v_a)\prec^{(0)}\boldsymbol{u}(i_b,j_b,k_b,v_b)$ 当且仅当有 $i_a<i_b,j_a<j_b$。其中“$\prec^{(0)}$”为 U 上的二元关系符。

定义 2(代数结构 $\mathbf{L}(G,\prec^{(t+1)})(t=0,1,2,\cdots)$)　给定格 $\mathbf{L}(U,\prec^{(t)})(t=0,1,2,\cdots)$ 与序列 $\boldsymbol{x}(x_1,x_2,\cdots,x_m)$。递归定义代数结构 $\mathbf{L}(N,\prec^{(t)})$ 与序列 $\boldsymbol{x}$ 上的代数结构 $\mathbf{L}(G,\prec^{(t+1)})$ 如下：

设 G 为形式为 $(\mu,\omega,\lambda,\sigma)$ 的四元组集合，为 $\{1,\cdots,|U|\}\times\{1,\cdots,|X|\}\times\{1,\cdots,|G|\}\times\Sigma'$ 上的子集，其中 $U=\{\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_{|U|}\},X=(x_1,x_2,\cdots,x_m),(1,1,1,\alpha)\in G,(|U|,|X|,|G|,\beta)\in G$。

记 $\boldsymbol{g}.\mu,\boldsymbol{g}.\omega,\boldsymbol{g}.\lambda,\boldsymbol{g}.\sigma$ 分别为元素 $\boldsymbol{g}$ 的 4 个分量。其中 μ 与 ω 分别对应代数结构 $\mathbf{L}(U,\prec^{(t)})$ 与序列 $\boldsymbol{x}$ 中的元素下标，λ 为元素 $\boldsymbol{g}$ 在集合 G 中的下标，σ 为元素 $\boldsymbol{g}$ 上的字符。

对于 $\forall\boldsymbol{g}(i,j,k,v)\in G$ 有 $\boldsymbol{u}_i.\sigma=x_j=v(v\in\Sigma')$。满足对任意 $\boldsymbol{g}(i_1,j_1,k_1,v_1),\boldsymbol{g}(i_2,j_2,k_2,v_2)\in G$ 有 $\boldsymbol{g}(i_1,j_1,k_1,v_1)\prec^{(t+1)}\boldsymbol{g}(i_2,j_2,k_2,v_2)$ 当且仅当有 $u_{i_1}\prec^{(t)}u_{i_2},j_1<j_2$ 成立。其中“$\prec^{(t+1)}$”为 G 上的二元关系符。

定理 1　设 $\mathbf{L}(G,\prec^{(t)})(t=0,1,2,\cdots)$ 为由定义 1 和定义 2 生成的代数结构，则 $\mathbf{L}(G,\prec^{(t)})$ 为格。

证明：若由定义 1 和定义 2 生成的代数结构 $\mathbf{L}(G,\prec^{(t)})$ 为格，则要满足“$\prec^{(t)}$”为 G 上的严格偏序，且 G 中元素均有上下确界。用数学归纳法进行证明，先证当 $t=0$ 时命题成立。

(1)对于集合 G，若“$\prec^{(0)}$”，“$\prec^{(t+1)}$”为 G 上的严格偏序，则满足反自反性、非对称性、传递性。

反自反性：对于 $\forall\boldsymbol{a}\in G$，有 $\boldsymbol{a}\not\prec^{(0)}\boldsymbol{a}$。

证明：对于 $\forall\boldsymbol{a}\in G$，若有 $\boldsymbol{a}\prec^{(0)}\boldsymbol{a}$，则 $i_a<i_a,j_a<j_a$，而 $x_{i_a}=x_{i_a},y_{j_a}=y_{j_a}$，矛盾。“$\prec^{(0)}$”反自反性得证。

非对称性：$\forall\boldsymbol{a},\boldsymbol{b}\in G,\boldsymbol{a}\prec^{(0)}\boldsymbol{b}\Rightarrow\boldsymbol{b}\not\prec^{(0)}\boldsymbol{a}$。

证明：由 $\boldsymbol{a},\boldsymbol{b}\in G$ 与 $\boldsymbol{a}\prec^{(0)}\boldsymbol{b}$，可知 $i_a<i_b,j_a<j_b$。若 $\boldsymbol{b}\prec^{(0)}\boldsymbol{a}$ 成立，当且仅当 $i_b<i_a,j_b<j_a$ 成立，矛盾。“$\prec^{(0)}$”非对称性得证。

传递性：$\forall\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}\in G$，由 $\boldsymbol{a}\prec^{(0)}\boldsymbol{b}\wedge\boldsymbol{b}\prec^{(0)}\boldsymbol{c}\Rightarrow\boldsymbol{a}\prec^{(0)}\boldsymbol{c}$。

证明：对于 $a,b,c\in G$，由 $a\prec^{(0)}b$，可知 $i_a<i_b,j_a<j_b$；由 $b\prec^{(0)}c$，可知 $i_b<i_c,j_b<j_c$；故可得，$i_a<i_b<i_c,j_a<j_b<j_c$。所以有 $i_a<i_c,j_a<j_c$，推出 $a\prec^{(0)}c$。“$\prec^{(0)}$”传递性得证。

(2)对于集合 G 中任意元素，均有上确界与下确界。

由定义 1 和定义 2 可知，元素 $g(1,1,1,\alpha)$ 与 $g(|U|,|X|,|G|,\beta)$ 分别是集合 G 中元素的上确界与下确界，命题得证。

假设命题 $t=h$ 时成立，当 $t=h+1$ 时，类似上述证明方法，可分别证明反自反性、非对称性、传递性。因此“$\prec^{(t)}(t=0,1,2,\cdots)$”是 G 上的严格偏序。

证毕。

对于格 $\mathbf{L}(U,\prec^{(t)})$，为叙述方便，在 t 给定的情况下，可将其简记为 $\mathbf{L}_U$。

定理 1 证明了定义 1 与定义 2 的代数结构均为格，因此称定义 1 中 $\mathbf{L}(U,\prec^{(0)})$ 为序列 x,y 的“公共格”。定义 2 中，$\mathbf{L}(G,\prec^{(t+1)})$ 为格 $\mathbf{L}(U,\prec^{(t)})$ 与序列 x 的“公共格”。

定理 1 也表明了定义 1 与定义 2 的偏序关系和递归定义的合理性。

定义 3(前驱集合 $Prec(u)$) 给定格 $\mathbf{L}(U,\prec^{(t)})$，定义 $Prec(u)\subset U\rightarrow 2^U$ 为结点 u 的前驱集合，满足 $Prec(u)=\{u_i\mid \exists u_j\in\mathbf{L}_U,\text{s.t. }u_i\prec^{(t)}u_j\prec^{(t)}u\}$。

定义 4(路径集合 $Paths(\mathbf{L}_U,u_i,u_j)$) 给定格 $\mathbf{L}(U,\prec^{(t)})$ 中的结点 u_i 与 u_j，称 $e(u_i,u_{i_2},u_{i_3},\cdots,u_{i_{(L-1)}},u_j)$ 为格 $\mathbf{L}(U,\prec^{(t)})$ 中结点 u_i,u_j 间的一条路径(其中 L 为路径长度)，当且仅当 $u_i\in Prec(u_{i_2})\wedge u_{i_2}\in Prec(u_{i_3})\wedge\cdots\wedge u_{i_{(L-1)}}\in Prec(u_j)$ 且 $\exists u_k\in\mathbf{L}_U\wedge u_k\neq\alpha,\beta$ 使得 $u_k\in Prec(u_i)\vee u_j\in Prec(u_k)$。记 $Paths(\mathbf{L}_U,u_i,u_j)$ 为格 $\mathbf{L}(U,\prec^{(t)})$ 中 u_i,u_j 结点间所有路径的全集。

定义 5(路径上的序列 $Seq(e)$) 给定格 $\mathbf{L}(U,\prec^{(t)})$ 的路径全集 $Paths(\mathbf{L}_U,\alpha,\beta)$，对任意一条路径 $e(u_{i_1},u_{i_2},\cdots,u_{i_L})\in Paths(\mathbf{L}_U,\alpha,\beta)$，定义 $Seq(e)$ 为取路径 e 上的序列，有 $Seq(e)=(u_{i_1}.\sigma,u_{i_2}.\sigma,\cdots,u_{i_L}.\sigma)$。其中 $u_{i_t}.\sigma$ 为取格 $\mathbf{L}_U$ 中元素 u_{i_t} 上的字符。

定理 2 设代数结构 $\mathbf{L}(U,\prec^{(t)})$ 为序列 $x(x_1,x_2,\cdots,x_m)$ 和 $y(y_1,y_2,\cdots,y_m)$ 上的生成格，则对 $\forall e\in Paths(\mathbf{L}_U,\alpha,\beta)$ 有：序列 $Seq(e)$ 为序列 x 和序列 y 的一条公共子序列(CS)。

证明：已知格 $\mathbf{L}(U,\prec^{(t)})$ 的路径全集 $PathSet(\mathbf{L}_U,\alpha,\beta)$，不失一般性，可设其中的一条路径为 $e(u_{i_1},u_{i_2},\cdots,u_{i_L})$，可知路径中的结点满足 $u_{i_1}\prec^{(t)}u_{i_2}\prec^{(t)}\cdots\prec^{(t)}u_{i_L}$，$e$ 所对应的序列为 $(u_{i_1}.\sigma,u_{i_2}.\sigma,\cdots,u_{i_L}.\sigma)$。

由定义 4，可得 $u_{i_1}\in Prec(u_{i_2})\wedge u_{i_2}\in Prec(u_{i_3})\wedge\cdots\wedge u_{i_{(L-1)}}\in Prec(u_{i_L})$。把格中的结点 u_{i_t} 对应序列 x 与序列 y 中的元素分别记为 $x_{i_{tx}}$ 与 $y_{i_{ty}}$，根据定义 1，有 $i_{1x}<i_{2x}<\cdots<i_{Lx},i_{1y}<i_{2y}<\cdots<i_{Ly}$。

因此可知序列 $(x_{i_{1x}},x_{i_{2x}},\cdots,x_{i_{Lx}})$ 为序列 $x(x_1,x_2,\cdots,x_m)$ 的子序列，序列 $(y_{i_{1y}},y_{i_{2y}},\cdots,y_{i_{Ly}})$ 为序列 $y(y_1,y_2,\cdots,y_n)$ 的子序列。

由定义 1，有 $x_{i_{1x}}=y_{i_{1y}},x_{i_{2x}}=y_{i_{2y}},\cdots,x_{i_{Lx}}=y_{i_{Ly}}$，可得序列 $(x_{i_{1x}},x_{i_{2x}},\cdots,x_{i_{Lx}})$ 与序列 $(y_{i_{1y}},y_{i_{2y}},\cdots,y_{i_{Ly}})$ 中序列元素相同，且与序列 $(u_{i_1}.\sigma,u_{i_2}.\sigma,\cdots,u_{i_L}.\sigma)$ 元素相同。因此路径 $e(u_{i_1},u_{i_2},\cdots,u_{i_L})$ 上的序列 $Seq(e)$ 为序列 x 与序列 y 的公共子序列。

证毕。

定理 2 表明，在格中，由于路径为多条，因此格中保存的公共子序列也为多个。

定理 3 给定格 $\mathbf{L}(U,\prec^{(t)})$ 与序列 x 和它们的公共格 $\mathbf{L}(G,\prec^{(t+1)})$，对于任意一条路径 $e_1\in Paths(\mathbf{L}_U,\alpha,\beta)$，必有 $\exists e_2\in Paths(\mathbf{L}_G,\alpha,\beta)$，使得序列 $Seq(e_2)$ 为序列 $Seq(e_1)$ 与序列 x 的公共子序列(CS)。

证明：同定理 3 的证明方法，易证 e_1 中的结点满足偏序关系“$\prec^{(t)}$”，序列 x 中的结点满足先后关系，且 e_2 中元素与 e_1,x 中对应的元素相同，因此 $Seq(e_2)$ 为 $Seq(e_1)$ 与序列 x 的公共子序列。

图 1 示出了一个格结构的示例，记图 1 中的格结构为 $\mathbf{L}(U,\prec^{(t)})$，其中 $u_1=\alpha,u_{16}=\beta$。由定义 1 与定义 2 可知 $\mathbf{L}(U,\prec^{(t)})$ 中结点的偏序关系，例如 $u_2\prec^{(t)}u_{13},u_4\prec^{(t)}u_{10}$ 等。由定义 4，可知 $Prec(u_7)=\{u_2,u_3,u_4\}$ 且有 $u_2\in Prec(u_7)$。对于 $\mathbf{L}(U,\prec^{(t)})$ 的路径全集 $Paths(\mathbf{L}_U,\alpha,\beta)$，有 $e(u_2,u_6,u_{10},u_{15})\in Paths(\mathbf{L}_U,\alpha,\beta)$。

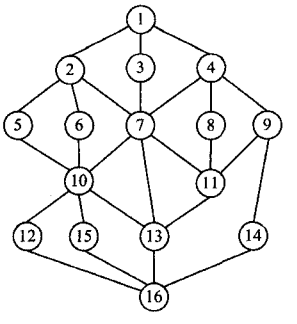

图 1

3.2 公共格的求解

3.2.1 本文算法思想

给定 d 个元素的序列集合 $S=\{s_1,s_2,\cdots,s_d\}$，记 $LAT(s_i,s_j)$ 为求解序列 s_i 与 s_j 的公共格操作，其中 $s_i,s_j\in S$。则对于集合 S 中 d 条序列的公共格的求解，可以表示为：

$$\mathbf{L}(G,\prec^{(t)})=LAT(LAT(LAT(s_1,s_2),s_3),\cdots,s_d)$$

每步 LAT 操作得到一个公共格，由定理 3 与定理 4，可知公共格的路径上的序列为原格与原序列的公共子序列。在求解出格 $\mathbf{L}(G,\prec^{(t)})$ 后，寻找 $\mathbf{L}(G,\prec^{(t)})$ 中最长路径上的序列，即为 d 条序列的最长公共子序列。

每步 LAT 操作需要对公共格进行剪枝操作，限制公共格的结点数量。剪枝操作可避免在进行多次 LAT 操作后，公共格的结点数量呈指数级增大，使后续的计算时间大幅度增加。通过调整每步贪心算法中获得的公共格中的结点数量，可以控制算法的计算时间与结果集的精确度。结点数量越多，则解集的精确度越好。

本文算法总体流程的伪代码如下。

算法 1 计算多条序列的最长公共子序列算法

输入：d 条序列的集合 S={$s_1,s_2,\cdots,s_d$}，公共格中结点数量的期望值 size

输出：集合 S 的最长公共子序列集合 C

LatticeApproach(S, size)：

1. let h=s_1, C={}
2. for i=2 to d do
3. h=GetLattice(s, s_i, size)
4. get LCSs of S from h and put them into C
5. return C

其中 GetLattice 方法为公共格生成算法，详见算法 3。

3.2.2 公共格匹配矩阵的计算

计算匹配矩阵元素 r_{ij} 的值的公式如下：

$$r_{ij}=\begin{cases}0, & u_i.\sigma=\alpha \vee x_j\neq\alpha \\ \max\limits_{p}(r_{p,j-1})+1, & u_i.\sigma=x_j\neq\alpha \\ \max\begin{cases}r_{i,j-1}\\ \max\limits_{p}(r_{pj})\end{cases}, & u_i.\sigma\neq x_j \wedge u_i.\sigma, x_j\neq\alpha\end{cases}$$

(where $p\in\{k\mid u_k\in Prec(u_i)\}$)

设格 $L(U,\prec^{(t)})$ 与序列 $x(x_1,x_2,\cdots,x_m)$ 的匹配矩阵为 $R_{|U|\times m}$，R 中元素记为 r_{ij}，r_{ij} 对应 $L(U,\prec^{(t)})$ 与 x 中的结点记为 u_i,x_j。对于 $\forall e\in Paths(L_U,\alpha,u_i)$，$r_{ij}$ 的值表示 $Seq(e)$ 与 x 的最长公共子序列的最大长度。

定义 6(匹配点集合 $Match(R)$)　对于匹配矩阵 R 中的元素 r_{ij}，若 r_{ij} 满足 $u_i.\sigma=x_j$ 并对 $\forall u_p\in Prec(u_i)$，有 $r_{ij}>r_{i,j-1}$，$r_{ij}>r_{pj}$ 成立，则称 r_{ij} 为“匹配点”。R 中所有“匹配点”的全集记为 $Match(R)$。

算法 2　匹配点集合求解算法

输入：格 L (U, $\prec^{(t)}$) 与序列 x

输出：匹配点集合 W=Match(R)

GetMatchPointSet(L (U, $\prec^{(t)}$), x)：

1. let R[|U|+1][|x|+1]={0}, w={}
2. for i=1 to |U| do
3. for j=1 to |x| do
4. if $u_i.\sigma=x_j$ then
5. for each $u_p\in$Prec(u_i) do
6. r_{ij}=max($r_{p,j-1}$)+1
7. if r_{ij} is a match point then
8. add r_{ij} to W
9. else
10. for each $u_p\in$Prec(u_i) do
11. r_{ij}=max($r_{i,j-1}$, r_{pj})
12. return W

匹配点集合求解算法采用动态规划思想计算出匹配矩阵。对于匹配矩阵 R 中任意一个元素 r_{ij}：

若 $u_i.\sigma=x_j$，则 r_{ij} 的值为对于 $u_p\in Prec(u_i)$ 与 x_{j-1}，有 $r_{ij}=max(r_{p,j-1})+1$。$r_{p,j-1}$ 不代表 R 中的一个单独的元素，而是集合 $\{u_p\mid u_p\in Prec(u_i)\}$ 与 x_{j-1} 中，每个 u_p 结点与 x_{j-1} 所对应的 $r_{p,j-1}$ 位置的元素的全体。

若 $u_i.\sigma\neq x_j$，对于 $u_p\in Prec(u_i)$ 与 x_{j-1}，有 $r_{ij}=\max(r_{pj}, r_{i,j-1})$。$r_{pj}$ 与 $r_{i,j-1}$ 不代表 R 中的单独元素，而是对应位置的元素的全体。

3.2.3 公共格的生成

定义格 $L(U,\prec^{(t)})$ 与序列 $x(x_1,x_2,\cdots,x_m)$ 的匹配矩阵为 $R_{|U|\times m}$，对任意 $r_{ij}\in R$，r_{ij} 对应 $L(U,\prec^{(t)})$ 与 x 中的结点记为 u_i,x_j。根据定义 6，令 $W=Match(R)$ 为匹配矩阵 R 中的匹配点全集。

定义 7(支配关系“$\lhd$”)　若对 $r_{pq},r_{ij}\in R$，满足 $u_p\prec^{(t)}u_i$ 且 $q<j$，那么称 r_{pq} 支配 r_{ij}，记为 $r_{pq}\lhd r_{ij}$。

对于 $W=Match(R)$，把 W 中元素的最大值记为 z，新生成的格记为 $L(G,\prec^{(t+1)})$。由于匹配矩阵提取格的过程是一个递归的过程，不失一般性，可设 $w_{ij}\in\{w_{ab}\mid w_{ab}\in W, w_{ab}=c\}$ $(1<c\leqslant z)$ 作为当前“起始点”，若 w_{ij} 对应的结点 $g'(i,j,k,v)$ 不在集合 G 中，则 $G=\{g'(i,j,k,v)\}\cup G$，否则 $g'(i,j,k,v)$ 已在 G 中。

算法 3 给出了格生成算法的伪代码。若 $w_{ij}=1$，则返回到递归的上一层调用；否则对于 $w_{pq}\in\{w_{ab}\mid w_{ab}\in W, w_{ab}=c-1\}$，若满足 $w_{pq}\lhd w_{ij}$，则：

若 $g'(p,q,k,v)\notin G$，则 $G=\{g'(p,q,k,v)\}\cup G$，将 w_{pq} 设为当前起始点递归地重复上述过程。

若 $g'(p,q,k,v)\in G$，则继续寻找 $\{w_{ab}\mid w_{ab}\in W, w_{ab}=c-1\}$ 中不同于 w_{pq} 的点 $w_{p'q'}$ 且满足 $w_{p'q'}\lhd w_{ij}$，并判断 $w_{p'q'}$ 对应点 $g''(p',q',k',v')$ 是否已在集合 G 中。直到对于 $\forall w_{ab}\in W$ 有 $g'(p,q,k,v)\in G$ 或 $|G|>size$，则格的生成算法结束。当格的生成结束后，会对求得的新公共格中的结点的下标进行重新赋值，因此格的生成算法中，k 的值对新公共格的生成无本质影响。不妨令 $k=0$。

算法 3　公共格生成算法

输入：格 L (U, $\prec^{(t)}$) 与序列 x，公共格结点数量的期望值 size

输出：一个格结构 L (G, $\prec^{(t+1)}$)(|G| 约束)

GetLattice(L (U, $\prec^{(t)}$), x, size)：

1. let G={}
2. W=GetMatchPointSet(L (U, $\prec^{(t)}$), x);
3. for each w_{ij} in $\{w_{xy}\mid w_{xy}\in W, w_{xy}=z\}$ do
4. add g'(i,j,k,v) to G
5. LatticeRecursion (w_{ij}, G, size)
6. let f=1
7. for each g in G
8. let g. λ=f; f=f+1
9. return (L (G, $\prec^{(t+1)}$)

LatticeRecursion (w_{ij}, G, size)：

1. if w_{ij}=1 then
2. return
3. else
4. for each w_{pq} in $\{w_{ab}\mid w_{ab}\in W, w_{ab}=c-1\}$ do
5. if $w_{pq}\lhd w_{ij}$ then
6. mark g'(p,q,k,v′)$\prec^{(t+1)}$g(i,j,k,v)
7. if g'(p,q,k,v′)$\notin$G then
8. G=$\{g'(p,q,k,v')\}\cup$G
9. if |G|>size then
10. terminate the algorithm
11. else
12. LatticeRecursion (w_{pq}, G, size)

4　实验分析

本节通过实验来检验本文的算法。实验集合的字符集为 $\Sigma=['A','G','C','T']$，序列为字符集 Σ 中的字符产生的

随机序列。实验分为两个部分:1)比较本文算法与其他算法的表现;2)检测当每步贪心算法公共格中的结点数量不同时本文算法的表现。

由定义 1 与定义 2 可知,格中的 α,β 结点对计算结果无本质影响,在实际计算中可省略。

4.1 3 个算法的实验比较

本节比较本文算法与其他近似算法的表现。其中 N 为原始序列集合中序列的平均长度。数据集中 10(50)代表 10 条长度为 50 的序列集合,以此类推。相关位置算法简称为相位算法。

表 1 本文算法与其他算法的实验结果

数据集	贪心算法	相位算法	本文算法(1N)	
	LCS 长度	LCS 长度	LCS 长度	LCS 数量
10(50)	12	13	14	8
10(100)	25	29	29	2
40(100)	18	20	21	8
100(200)	40	39	40	2
100(300)	62	59	63	2

从表 1 的实验结果中可以归纳出,在设定公共格的结点数量是数据集中的序列平均长度的 1 倍时,对于不同的数据集,本文的算法得到的 LCS 长度均优于传统贪心算法与相关位置算法;且本文算法所求得的 LCS 为多条,相比于传统贪心算法与相关位置算法,可以保留更多的信息。该实验也验证了本文算法最终求得的解为多解的结论的正确性。

4.2 公共格的结点数量不同时本文算法的表现

从表 2 所列的实验结果中可以归纳出,当公共格的结点数量分别为 $1N,3N,5N$(N 为原始序列集合中序列的平均长度)时,若求解出的 LCS 的长度没有改变,那么解集中 LCS 的条数便会增多。若解集中 LCS 的长度增加,则原解集中 LCS 均变为了次长公共子序列,因此解集中 LCS 的数量会减少。

表 2 格期望不同的情况下的算法表现

格期望	1N		3N		5N	
数据集	长度	数量	长度	数量	长度	数量
10(50)	14	8	14	18	15	1
10(100)	29	2	29	20	30	2
40(100)	21	8	21	17	22	98
100(200)	40	2	42	8	42	20
100(300)	63	2	64	2	64	14

可以总结为:若增加公共格的结点数量,公共格中保存的公共子序列的信息也会增加,则公共格中的路径数量就会增加,就会有更大的可能去寻找到更长的公共子序列。即使 LCS 的长度没有改变,格中包含的比原来更多的信息也可以使得最终求解出的 LCS 的数量增加。

结束语 求解最长公共子序列在诸多领域内有着重要的应用,如信息检索、基因序列匹配等。求信息序列的最长公共子序列可以提取信息序列的公共信息,从而进行进一步的检索与分类。求基因序列的最长公共子序列可以获取基因序列间的匹配度。

本文算法引入了代数结构“格”,通过动态规划求解出两条序列的公共格,并结合贪心策略递归求解当前格与当前序列的公共格。公共格的路径保存了多条公共子序列,由于格结构的多路径性质,使得求解出的最长公共子序列有多个。对算法的相关定理给出了理论证明。通过实验验证了本文算法的正确性,归纳讨论了本文算法在公共格结点不同的情况下所求得的解集的变化趋势与原因。

参 考 文 献

[1] MAIER D. The complexity of some problem on subsequences and superseuqences[J]. Journal of the ACM (JACM), 1978, 25(2): 322-366.

[2] BERGROTH L, HAKONEN H, RAITA T. A survey of longest common subsequence algorithms[C]// Processings. Seventh International Symposium on String Processing and Information Retrieval, 2000(SPIRE 2000). IEEE, 2000: 39-48.

[3] ATTWOOD T, FINDLAY J. Fingerprinting g-protein-coupled receptors[J]. Protein engineering, 1994, 7(2): 195-203.

[4] SANKOFF D, BLANCHETTE M. Phylogenetic invariants for genome rearrangements[J]. Journal of Computational Biology, 1999, 6(3/4): 431-445.

[5] SHUKLA A, AGARWAL S. A relative position based algorithm to find out the longest common subsequence from multiple biological sequences[C]// Proceedings of the 2010 International Conference on Computer and Communication Technology (ICCCT). 2010: 496-502.

[6] RIZVI S, AGARWAL P. A new index-based parallel algorithm for finding longest common subsequence in multiple dna sequences[C]// International Conference in Cognitive Systems. Citeseer, 2005.

[7] HAKATA K, IMAI H. Algorithms for the longest common subsequence problem for multiple strings based on geometric maxima[J]. Optimization Methods and Software, 1998, 10(2): 233-260.

[8] BOURQUE G, PEVZNER P A. Genome-scale evolution: reconstructing gene orders in the ancestral species[J]. Genome research, 2002, 12(1): 26-36.

[9] SHERIDAN R P, VENKATARAGHAVAN R. A systematic search for protein signature sequences[J]. Proteins-Structure Function and Bioinformatics, 1992, 14(1): 16-28.

(上接第 261 页)

[14] SUN S P. Research on Chinese Micro-blog Hot Topic Detection and Tracking[D]. Beijing: Beijing Jiaotong University, 2011. (in Chinese)
孙胜平. 中文微博客热点话题检测与跟踪技术研究[D]. 北京: 北京交通大学, 2011.

[15] MI W L, SUN Y X. Microblog Hot Topics Discovery Method based on Probabilistic Topic Model[J]. Computer Systems & Applications, 2014, 23(8): 163-167. (in Chinese)
米文丽, 孙曰昕. 利用概率主题模型的微博热点话题发现方法[J]. 计算机系统应用, 2014, 23(8): 163-167.

[16] ZHENG L. Reserch and Application of Topic Detection on Micro-Blog[D]. Harbin: Harbin Institute of Technology, 2012. (in Chinese)
郑磊. 微博客话题检测的研究与实现[D]. 哈尔滨: 哈尔滨工业大学, 2012.