短块移动排序的 $14/11$ 近似算法

姜海涛, 朱大铭*

山东大学计算机科学与技术学院, 济南 250101

* 通信作者. E-mail: dmzhu@sdu.edu.cn

收稿日期: 2008–11–27; 接受日期: 2010–03–25

国家自然科学基金 (批准号: 60970003, 61070019) 和教育部博士点基金 (批准号: 20090131110009) 资助项目

摘要 块移动是基因重组的一种重要形式. 短块移动是将排列中的元素最多移动到偏离原来两个位置的块移动. Heath 和 Vergara 最先给出短块移动排序近似度为 $4/3$ 的多项式时间算法. 本文设计了近似性能比为 $14/11$ 的短块移动排序新算法. 首先讨论了具有伞形结构排列图的子排列的排序方法, 并将这种子排列称为‘伞’, 设计了特殊子排列伞短块移动排序的多项式时间精确算法. 然后给出关联伞子排列短块移动排序的贪心算法. 讨论了 5 种特殊子排列的短块移动排序方法, 证明了它们短块移动距离的新下界, 从而证明此贪心算法的近似性能比为 $14/11$, 这是目前解答短块移动排序问题近似性能比最小的多项式近似算法.

关键词 短块移动 算法 近似度

1 引言

推导物种演化历史, 一种十分有前途的方法就是比较两个基因组中基因排列的顺序. 两个物种往往含有相同的基因集合, 而基因的排列顺序不同^[1,2]. 这促使人们利用重组事件追踪物种的演化过程. 基因组重组是一个十分复杂的过程, 但 Sankoff 等^[3] 将基因组重组抽象为 3 种基本的基因重组操作, 分别称为 Reversal, Translocation 和 Transposition. 关于基因组 Reversal 与 Translocation 排序问题的算法与复杂性研究结果可见文献 [4–11].

Transposition 是关于一个整数排列的重组操作. 一次 Transposition 将整数排列的一个片段从排列中分离出来, 转移并连接到该排列的另一个位置. Transposition 相当于交换排列中相邻的两个片段. Bafna 和 Pevzner^[12] 最先设计了一个近似度为 1.5 的 Transposition 排序多项式算法, 其时间复杂性为 $O(n^2)$, 其中 n 为排列中元素的个数. Hartman 与 Shamir^[13] 简化了该算法, 并将时间复杂性改进为 $O(n\sqrt{n\log n})$, 最近 Elias 与 Hartman^[14] 将文献 [13] 算法的近似度改进为 1.375. Feng 与 Zhu^[15] 利用排列树将文献 [13] 的算法的时间复杂性改进为 $O(n\log n)$. Transposition 排序的复杂性目前仍是未知的.

Transposition 也称为块移动 (Block-Move). 块移动排序过程中, 一个片段移动距离较远的可能性不大. 因此 Heath 和 Vergara 研究了整数排列的短块移动问题 (Short Block-Move). 短块移动是相邻

引用格式: 姜海涛, 朱大铭. 短块移动排序的 $14/11$ 近似算法. 中国科学: 信息科学, 2011, 41: 33–47

两段的总长度不超过 3 的块移动. Heath 与 Vergara^[16] 给出短块移动问题近似度为 $4/3$ 的多项式时间算法, 并对 bitonic 序列和 double-strip 序列给出 $O(n^3 \log n)$ 的精确算法, 是目前所知最好的正面结果.

本文首先讨论了具有伞形结构排列图的子排列的排序方法, 并将这种子排列称为‘伞’, 设计了特殊子排列‘伞’短块移动排序的多项式时间精确算法. 然后给出关联伞子排列短块移动排序的贪心算法. 我们讨论了 5 种特殊子排列的短块移动排序方法, 证明了它们短块移动距离的新下界, 从而证明此贪心算法的近似性能比为 $14/11$.

2 短块移动问题简介

若一个排列含有 n 个符号, 则采用整数集 $I_n = \{1, 2, \dots, n\}$ 表示该排列的所有元素符号, 例如 $\pi = [3, 4, 1, 5, 6, 2]$, 为一个含有 6 个符号的排列. 在排列 $\pi = [g_1, g_2, \dots, g_n]$ 中, $g_i \in I_n (1 \leq i \leq n)$, 称 g_i 为 g_{i+1} 的左邻, g_{i+1} 为 g_i 的右邻 ($1 \leq i \leq n-1$). π 的一个非空子排列称为排列的一个块, 将块 $[g_i, \dots, g_{j-1}]$ 由 π 中脱离原位置, 转移到 g_{k-1} 和 g_k 之间, 形成一个新排列 $\pi' = [g_1, \dots, g_{i-1}, g_j, \dots, g_{k-1}, g_i, \dots, g_{j-1}, g_k, \dots, g_n]$, 称为 π 的一次块移动, $1 \leq i < j < k \leq n+1$, 这个块移动也称为子排列 $[g_i, \dots, g_{j-1}]$ 和 $[g_j, \dots, g_{k-1}]$ 的换位. 其中若 $i = 1$, 则 $[g_1, \dots, g_{i-1}] = \emptyset$; 若 $k = n+1$, 则 $[g_k, \dots, g_n] = \emptyset$. 若 $k-i \leq 3$, 则称这种换位为短块移动. 若位置 i 确定, 则可能的短块移动操作有 3 种, 即:

1. Skip: $\rho([i], [i+1])$, $[g_i]$ 与 $[g_{i+1}]$ 换位, 也称 g_i 或 g_{i+1} 的 Skip;
2. Right-Hop: $\rho([i], [i+1, i+2])$, $[g_i]$ 与 $[g_{i+1}, g_{i+2}]$ 换位, 也称 g_i 的右跳;
3. Left-Hop: $\rho([i, i+1], [i+2])$, $[g_i, g_{i+1}]$ 与 $[g_{i+2}]$ 换位, 也称 g_{i+2} 的左跳.

我们也将 Skip 称为 Skip 移动或 Skip 操作, 将 Hop 称为 Hop 移动或 Hop 操作. 下面我们也直接用 $\rho(g_i, g_{i+1})$ 表示 Skip, 用 $\rho(g_i, g_{i+1}g_{i+2})$ 或 $\rho(g_i g_{i+1}, g_{i+2})$ 表示 Hop.

短块移动 ρ 将排列 π 转化为 π' , 记为 $\pi \cdot \rho = \pi'$. 短块移动问题要求利用短块移动操作将任意排列 $\pi = [g_1, g_2, \dots, g_n]$ 转化为单位排列 $\iota = [1, 2, \dots, n]$, 所用移动次数最少. 形式化描述为

实例: 整数排列 $\pi = [g_1, g_2, \dots, g_n], g_i \in I_n, 1 \leq i \leq n$.

询问: 求短块移动序列 $\rho_1, \rho_2, \dots, \rho_t$, 使 $\pi \cdot \rho_1 \cdot \rho_2 \cdots \rho_t = [1, 2, \dots, n]$, 且 t 最小.

将排列 π 转化为单位排列 ι 所需最少的短块移动次数称为 π 的短块移动距离, 记为 $BK_3(\pi)$. 设 $\pi = [g_1, g_2, \dots, g_n]$, 如果 g_i 和 g_j 满足 $g_i > g_j$, 且 $i < j$, 则称 g_i 和 g_j 是逆序排列对, π 中所有的逆序排列对个数简称 π 的逆序数. 显然在单位排列中不存在任何逆序. 将能够至少减少 π 的一个逆序的短块移动称为正确移动. Heath 和 Vergara 在文献 [17] 中已证明, 寻找一个排列的最优短块移动序列时, 只需要考虑正确的移动操作即可. 即:

定理 1 给定排列 π , 必存在最优的短块移动序列 $\rho_1, \rho_2, \dots, \rho_t$, 将 π 变为单位排列, 且每个移动 ρ_i 都是正确的, $1 \leq i \leq t$.

因此下文讨论排列的短块移动距离时, 总考虑正确的移动.

给定排列 $\pi = [g_1, g_2, \dots, g_n]$, 构造有向图 $G(\pi) = (V, E)$ 如下: $V = \{g_1, g_2, \dots, g_n\}$, $E = \{(g_i, g_j) \mid g_i > g_j \text{ 且 } i < j\}$. $G(\pi)$ 称为 π 的排列图. 例如 $\pi = [4, 2, 1, 3, 6, 5]$, 其排列图如图 1 所示.

排列图中的弧表示排列中的逆序. 我们称弧 (g_i, g_j) 为 g_i 的右关联弧, g_j 的左关联弧. 在排列图中, 称两条弧是相关的, 当且仅当它们有相同的头或尾. 排列图的弧 (g_i, g_{i+k}) 的长度定义为 $k (k \geq 1)$,

图 1 $\pi=[4, 2, 1, 3, 6, 5]$ 的排列图

并将长度为 1 的弧称为短弧, 长度不小于 2 的弧称为长弧. 在后面的描述中, 我们有时直接将 π 的排列图中的弧简称为 π 中的弧.

首先容易观察到, 任意正确的短块移动操作只能消除一条弧或两条相关的弧. 为方便, 我们不区分逆序与其对应的弧, 也不区分排列及其对应的排列图. 文献 [16] 由此给出短块移动距离的上界和下界如下:

$$\frac{|E(G(\pi))|}{2} \leq BK_3(\pi) \leq |E(G(\pi))|, \quad (1)$$

即任意排列的短块移动距离不大于其排列图的弧数且不小于排列图弧数的一半.

3 相关结论及算法

若排列图 $G(\pi)$ 的弧 (g_i, g_{i+1}) 与其他任意弧均不相关, 则称 (g_i, g_{i+1}) 为 $G(\pi)$ 的独立弧或 π 的独立弧, 或直接将子排列 $[g_i, g_{i+1}]$ 称为独立弧, 以下将独立弧简记为 Lone. 在排序 π 时, 每个独立弧必须通过一次 Skip 消除, 称消除独立弧的移动为 L -Skip. 显然一个 L -Skip 移动一定是正确的.

定义 1 设 $\pi=[g_1, g_2, \dots, g_n]$ 的子排列 $I=[g_i, g_{i+1}, g_{i+2}, g_{i+3}]$ 满足: $g_{i+2} > g_i > g_{i+3} > g_{i+1}$; 则称子排列 I 为蘑菇 (mushroom).

蘑菇的排列图子图由 4 个顶点、3 条弧组成. 一个正确的 Hop 必定消除两条弧. 一个 L -Skip 消除一条独立弧, 并且不可能产生新的独立弧. Heath 证明了下面两个重要的引理.

引理 1 如果排列中存在正确 Hop, 则必存在一个 Hop, 至多产生一条新的独立弧 [16].

引理 2 如果一个排列未排序且其中没有正确的 Hop, 则必由独立弧和蘑菇组成. 两次移动可消除一个蘑菇, 且不产生独立弧 [16].

4 改进算法

定义 2 设排列 $\pi=[g_1, g_2, \dots, g_n]$. 若子排列 $I=[g_i, g_{i+1}, \dots, g_{i+m}]$ 满足: $1 \leq m \leq n-i$; $g_i > g_{i+k}$; $k=1, 2, \dots, m$, 则称子排列 I 为 π 的一个伞 (Umbrella). 其中 g_i 称为伞的头 (head); g_{i+k} ($1 \leq k \leq m$) 称为伞内符号 (顶点); m 为伞的长度, 记为 $s[I]$. 在后面的描述中, 我们也将 $m=0$ 的伞称为零伞, 其长度为 0. 为清楚起见, 也用 $g_i[I]$ 表示 I 的第 i 个伞内符号, $g_0[I]$ 表示 I 的伞头. 显然长度为 1 的伞仅为一条短弧. 长度大于 1 的伞称为长伞. 显然一个长度为 m 的伞必存在 m 条以伞头为端点的两两相关的弧, 称为伞的头弧, 将两个端点均为伞内顶点的弧称为该伞的内弧, 头弧和内弧总称为伞内的弧或伞的弧. 下面将伞 I 的排列图 $G(I)$ 中所有弧的集合记为 $E(I)$. 如果 $|E(I)|$ 为奇数, 则称 I 为奇伞; $|E(I)|$ 为偶数, 则称 I 为偶伞, 零伞也为偶伞.

4.1 伞排序

若 π 的子排列 I 为伞, 且 I 内任意 3 个连续伞内符号的子排列均不形成 Hop, 则称 I 为简单伞 (simple umbrella), 简记为 SU. 为简单, 我们将长度为 1 或 2 的伞也称为简单伞. 我们下面说明, 对于简单伞, 总是存在一组 Hop 和至多一个 Skip 消除伞内的所有弧.

定理 2 给定简单伞 I , 则存在 $\lceil \frac{|E(I)|}{2} \rceil$ 次移动消除 I 内的弧. 其中前 $\lfloor \frac{|E(I)|}{2} \rfloor$ 个移动均为 Hop^[18].

4.2 关联伞排序

若一个伞为偶伞, 仅用 Hop 就可消除伞内的所有弧. 但若一个伞为奇伞, 则需要一个 Skip 才能消除伞内的所有弧. 若多个奇伞的顶点之间存在相互关联的弧, 我们能够使用更多 Hop 来消除其中的弧, 而不必每个伞使用一个 Skip.

引理 3 对于任何伞 $U=[g_i, g_{i+1}, \dots, g_{i+k}]$, 总存在一系列的 Hop 移动使 U 变为简单伞 $\hat{U}=[g_j, g_{j+1}, \dots, g_{j+t}]$, $\hat{U}$ 满足: (1) $g_j = g_i, t \in \{k, k-1\}$, $g_{j+m} \in \{g_{i+1}, \dots, g_{i+k}\}, 1 \leq m \leq t$. (2) $g_{j+1} = \min\{g_{j+1}, \dots, g_{j+t}\}$.

下面将一个伞 U 中最小的数记为 $\min(U)$. 设 $U=[g_i, g_{i+1}, \dots, g_j]$ 为正向简单伞, $g_{i+1}=\min(U)$. 设 g_l 是位于 g_i 左端、满足 $g_l > g_{i+1}$ 的所有顶点中, 离 g_i 最近的顶点, $l < i$, 则 g_l 有一条指向 g_{i+1} 的弧, 我们称 g_l 为 U 的关联点. 若 $g_l=g_{i-1}$, 则 Hop 移动 $\rho([i-1, i], [i+1])$ 可消除两条弧 (g_l, g_{i+1}) 和 (g_i, g_{i+1}) , 并改变 U 的奇偶性; 若 $l < i-1$, 则 $U_1=[g_l, \dots, g_{i-1}]$ 是伞, 伞头为 g_l . 我们称 U_1 关联 U , (g_l, g_{i+1}) 为 U_1 与 U 的关联弧. 若 $U=[g_j, g_{j+1}, \dots, g_{j+t}]$ ($t \geq 1$) 为满足 $g_{j+1} = \min(U)$ 的简单伞, 则称 U 为可左关联的简单伞, 简记为 MSU. 显然不存在长度为 2 的奇 MSU.

引理 4 设伞序列 $U_1, U_2, \dots, U_k$ 满足 U_2 关联 $U_1, \dots, U_k$ 关联 $U_{k-1}, U_1, \dots, U_{k-1}$ 均为奇 MSU, 则存在 $\lceil \frac{|E(U_1)|+|E(U_2)|+\dots+|E(U_k)+k-1|}{2} \rceil$ 次移动消除伞 $U_1, U_2, \dots, U_k$ 内的弧和其中的 $k-1$ 条关联弧.

证明 设 U_i 的伞头为 $g_0[U_i]$, $g_i[U_j]$ 表示伞 U_j 的第 i 个伞内符号 $1 \leq j \leq k, 1 \leq i \leq s[U_j]$. 由定理 2, 若 U_k 为偶伞, 利用 $\frac{|E(U_k)|}{2}$ 个 Hop 可消除 U_k 的所有弧; 若 U_k 为奇伞, 利用 $\lfloor \frac{|E(U_k)|}{2} \rfloor$ 次 Hop 与 1 次 Skip 可消除 U_k 的所有弧. U_k 中的符号排序后, $g_0[U_k]$ 移到 $g_0[U_{k-1}]$ 左邻, 此时可用 Hop 移动 $\rho(g_0[U_k], g_0[U_{k-1}], g_1[U_{k-1}])$ 消除 U_{k-1} 的弧 $(g_0[U_{k-1}], g_1[U_{k-1}])$ 和关联弧 $(g_0[U_k], g_1[U_{k-1}])$, U_{k-1} 变为偶伞. 由定理 2, 可用 $\frac{|E(U_{k-1})|-1}{2}$ 次移动消除 U_{k-1} 的其他所有弧, 此时 $g_0[U_{k-1}]$ 移动到 $g_0[U_{k-2}]$ 左邻. 因后面的伞均为奇伞, 可用与消除 U_{k-1} 弧相同的方法依次消除 $U_{k-2}, \dots, U_1$ 内的弧和另外 $k-2$ 条关联弧, 每次移动均为 Hop. 因此消除 $U_1, U_2, \dots, U_k$ 内的弧与其中的 $k-1$ 条关联弧所用移动次数为 $\lceil \frac{|E(U_1)|+|E(U_2)|+\dots+|E(U_k)+k-1|}{2} \rceil$, 其中含有 $\lfloor \frac{|E(U_1)|+|E(U_2)|+\dots+|E(U_k)+k-1|}{2} \rfloor$ 次 Hop 移动, 最多 1 次 Skip 移动.

根据引理 4 的证明, 给出关联伞排序算法, 见算法 1, 2.

设 $\mu=[U_k, U_{k-1}, \dots, U_1]$ 为 π 的子排列, 满足 U_2 关联 $U_1, \dots, U_k$ 关联 U_{k-1}, U_k 为简单伞, $U_1, \dots, U_{k-1}$ 均为奇 MSU. 则执行 Sort-Related-U($U_k, U_{k-1}, \dots, U_1$) 后, 排列 μ 变为如下形式:

$$\mu_1 = l_k, \text{re}(U_k), l_{k-1}, h_k, \text{re}(U_{k-1}), \dots, l_2, h_3, \text{re}(U_2), l_1, h_2, \text{re}(U_1), h_1. \quad (2)$$

其中 $l_i=g_1[U_i]$ ($1 \leq i \leq k-1$), $l_k=\min(U_k)$, $h_i=g_0[U_i]$ ($1 \leq i \leq k$), $\text{re}(U_i)$ ($1 \leq i \leq k$) 表示伞 U_i 中的符号 $g_2[U_i], g_3[U_i], \dots, g_{s[U_i]}[U_i]$ 排序后得到的新子排列. μ_1 满足如下性质:

算法 1 Sort-Related-U($U_k, U_{k-1}, \dots, U_1$)

输入: $\mu=[U_k, \dots, U_1]$, U_{i+1} 关联奇 MSU U_i , $1 \leq i \leq k-1$,

$U_i=[g_o[U_i], g_1[U_i], \dots, g_{s[U_i]}[U_i]]$, $1 \leq i \leq k$.

输出: 该伞的短块移动距离 d , 移动序列 $\rho[0 \sim d-1]$

```

1   $d=0$ , 构造排列图  $G(\mu)$ .
2  for  $i=k$  to 1 {
3      Sort-S-U( $U_i$ ).
4      if( $i > 1$ ){
5           $\rho[d]=\rho(g_o[U_i] \ g_o[U_{i-1}], g_1[U_{i-1}])$ 
6           $\mu=\mu \cdot \rho[d]$ ,  $d=d+1$ .  $U_{i-1}=[g_o[U_{i-1}], g_2[U_{i-1}], \dots, g_{s[U_{i-1}]}[U_{i-1}]]$ .  $s[U_{i-1}]=s[U_{i-1}]-1$ . }
7  返回  $d$ ,  $\rho[0 \sim d-1]$ .
```

算法 2 Sort-All-Related-U($U_k, U_{k-1}, \dots, U_1$)

```

1  Sort-Related-U( $U_k, U_{k-1}, \dots, U_1$ ).
2  for  $j=2$  to  $k-1$  {
3      if ( $s[U'_j] \geq 1$ ) Sort-S-U( $U'_j$ ).}
4      if( $s[U'_k] \geq 2$ ) Sort-S-U( $U'_k$ ).
5  返回  $d$ ,  $\rho[0 \sim d-1]$ .
```

(1) 若 $s[U_k]=0$, 则 l_k 为空; 若 $s[U_i]=1$, 则 $re(U_i)$ 为空, $1 \leq i \leq k$.

(2) 设 $x_i \in re(U_i)$, $1 \leq i \leq k$, 则对于满足 $1 \leq i < j \leq k$ 的 i, j , 有 $l_j < x_j < l_i < x_i$.

所以, 当 Sort-Related-U($U_k, U_{k-1}, \dots, U_1$) 执行完后, 排列 μ_1 中 l_i 和 $x_i \in re(U_i)$ 均无右关联弧, 只可能存在 h_i ($2 \leq i \leq k$) 的右关联弧. 如果 $h_1, h_2, \dots, h_i$ ($2 \leq i \leq k-1$) 都无右关联弧, 则 h_{i+1} 及它的所有右关联弧对应的子排列为伞 U'_{i+1} 且 U'_{i+1} 内无其他的弧. 因此, 我们可依次将以 $h_2, \dots, h_k$ 为伞头的伞 $U'_2, \dots, U'_k$ 用简单伞排序算法 Sort-S-U($\cdot$) 消除.

下面给出关联伞内所有符号的排序算法 Sort-All-Related-U($\cdot$). 算法结束时, 至多剩余一条关联伞排列的内弧 (U'_k). 该算法实际上可分为两阶段:

(1) 调用 Sort-Related-U($U_k, U_{k-1}, \dots, U_1$), 得到如 (2) 式形式的子排列 μ' ;

(2) 执行算法 2 的第 2~4 步, 消除 μ' 中的弧. 注意若 $s[U'_k]=1$, 则 U'_k 中的弧并不消除.

4.3 短块移动排序近似算法

给定 $\pi=[g_1, \dots, g_n]$ 的子排列 $I=[g_i, \dots, g_j]$, 设 $\min(I)$ 和 $\max(I)$ 分别表示 I 中最小的数和最大的数. 如果 $g_{i-x} < \min(I)$, $1 \leq x \leq i-1$, 则称 I 在 π 中左端独立; 若 $g_{j+y} > \max(I)$, $1 \leq y \leq n-j$, 则称 I 在 π 中右端独立. 将形如 (g_l, g_{i+t}) 的弧, $1 \leq l \leq i-1, 0 \leq t \leq j-i$ 称为 I 的左关联弧; 同样形如 (g_{i+t}, g_{j+r}) 的弧, $0 \leq t \leq j-i, 1 \leq r \leq n-j$ 称为 I 的右关联弧. I 的左关联弧和右关联弧均称为 I 的关联弧. 若 I 在 π 中左右端均独立, 则称 I 在 π 中是独立的, 即 I 为 π 的独立子排列, 称 I 中的弧为 I 内的弧或 I 的内弧. 将不包含其他独立子排列的独立子排列称为 π 的最小独立子排列.

首先将 π 划分为若干个最小独立子排列, 只须利用短块移动将每个最小独立子排列排序, 即可完成 π 的排序. 本文短块移动排序算法的主要想法是: 针对 π 的一个最小独立子排列找到一组由右向左

关联的简单伞, 并消除它们内部的所有弧, 重复这样的步骤直到将 π 变为 $[1, 2, \dots, n]$.

算法采用贪心技术. 首先从 g_n 由右向左扫描, 找到 g_x 和 g_y , 满足 $g_x = \max\{g_t | g_t \neq t, 1 \leq t \leq n\}$, $y = \max\{t | g_t \neq t, 1 \leq t \leq n\}$, 则 $U_1 = [g_x, \dots, g_y]$ 为伞. 以后找到的任何伞都是下列 3 种情况之一:

(1) U_1 为偶伞, 则可利用伞排序算法 Sort-S-U(U_1) 消除 U_1 内所有的弧.

(2) U_1 为奇伞且 $g_2[U_1] < g_1[U_1] < g_4[U_1]$, 则执行 $g_0[U_1]$ 的 1 次 Right-Hop, 可将 U_1 变为偶伞.

(3) U_1 通过 Hop 移动变为 MSU, 则能够找到 U_1 的左关联伞 U_2 , U_2 的左关联伞 $U_3, \dots$, 直到 U_i 满足 (1) 或 (2) 式或者左端独立.

算法第 2 步主循环最多执行 $O(n)$ 次, 第 3~6 步需要 $O(n^2)$ 时间, 第 7 步可在 $O(n^3)$ 时间内完成. 所以, 整个算法的时间复杂性是 $O(n^4)$.

在算法执行过程中, 将调用一次 Sort-All-Related-U($U_k, U_{k-1}, \dots, U_1$), 消除关联伞内弧的计算过程称为一轮, 即主 While 循环 (2~15) 执行 1 次为 1 轮. 算法中的第 t 轮, 如果执行第 13 步, 则弧 $e_1 = (g_1[U_k[t]], g_2[U_k[t]])$ 未被消除. 调用 Sort-All-Related-U($U_k[t], U_{k-1}[t], \dots, U_1[t]$) 一次, 最多剩余 $[U_k[t], U_{k-1}[t], \dots, U_1[t]]$ 中的一条弧 e_2 未被消除. 在 $t+1$ 轮, 算法找到另外一组关联伞 $U_h[t+1], U_{h-1}[t+1], \dots, U_1[t+1]$. 若 e_2 在第 t 轮第 11 或 15 步被消除, 则弧 e_1 或者为伞 $U_1[t+1]$, 或者为伞 $U_1[t+1]$ 的内弧; 否则, e_1 和 e_2 都是 $U_1[t+1]$ 的内弧. 显然都会被 Sort-All-Related-U($U_h[t+1], U_{h-1}[t+1], \dots, U_1[t+1]$) 消除. 因此算法 Short-Block-Move(π) 能消除最小独立子排列 π 中的所有弧.

注意, 除了最后一轮, 其他的轮调用 Sort-All-Related-U($\cdot$) 时, 最左边的伞都是偶伞.

算法 3 Short-Block-Move (π)

输入: $\pi = [g_1, g_2, \dots, g_n]$, $g_i \in \{1, 2, \dots, n\}$, $1 \leq i \leq n$.

输出: 该伞的短块移动距离 d , 移动序列 $\rho[0 \sim d-1]$

```

1   $d=0, g_x=g_y=0$ , 构造排列图  $G(\pi)$ .
2  While ( $\pi \neq [1, \dots, n]$ ) do {
3       $g_x = \max\{g_t | g_t \neq t, 1 \leq t \leq n\}$ ,  $y = \max\{t | g_t \neq t, 1 \leq t \leq n\}$ ,  $k=0, U_0=U=[g_x, \dots, g_y]$ ,  $g_0[U]=g_x$ .
4      执行 Hop 将  $U$  变为简单伞记录到  $\rho[d \sim d_1]$ .  $d=d_1+1, k=k+1, U_k=U$ .
5      while( $U_k$  为奇伞, 且  $g_1[U_k] < g_2[U_k]$  或  $g_2[U_k] < g_4[U_k] < g_1[U_k]$ ) {
6          执行 Hop 将  $U$  变为 MSU 记录到  $\rho[d \sim d_1]$ ,  $d=d_1+1$ .
7          If(存在  $U_k$  的左关联伞  $U$ ) {
8              执行 Hop 将  $U$  变为简单伞记录到  $\rho[d \sim d_1]$ .  $d=d_1+1, k=k+1, U_k=U$ .}
9      If( $U_k$  为偶伞, 或  $U_k$  为奇伞且左端独立) {
10         Sort-All-Related-U( $U_k, U_{k-1}, \dots, U_1$ )
11         if( $U'_k$  为 Lone) {执行 Skip  $\rho[d]$  消除  $U'_k$ ,  $d=d+1$ .} }
12     If( $U_k$  为奇伞, 且  $g_2[U_k] < g_1[U_k] < g_4[U_k]$ ) {
13          $\rho[d] = \rho(g_0[U_k], g_1[U_k], g_2[U_k])$ ,  $d=d+1$ .  $U_k = [g_0[U_k], g_3[U_k], \dots, g_s[U_k][U_k]]$ 
14         Sort-All-Related-U( $U_k, U_{k-1}, \dots, U_1$ )
15         if( $U'_k$  为 Lone) {执行 Skip  $\rho[d]$  消除  $U'_k$ ,  $d=d+1$ .} }
16  返回  $d, \rho[0 \sim d-1]$ .
```

图 2 双蘑菇的结构

图 3 9 种特殊子排列

5 算法的近似性能比分析

为证明算法近似性能比为 $14/11$, 我们需要先讨论特殊排列短块移动距离的下界.

5.1 特殊排列短块移动距离下界

在下面的短块移动距离下界讨论中, 需要用到排列图具有如下结构的子排列.

定义 3 若 π 的一段子排列 $I=[g_i, g_{i+1}, \dots, g_{i+5}]$ 满足 (1) $g_{i+4} > g_{i+2} > g_i > g_{i+5} > g_{i+3} > g_{i+1}$; (2) 任意 g_i 左端元素 $g_{i-x} < g_{i+5}$, $1 \leq x \leq i-1$, 任意 g_{i+5} 右端元素 $g_{i+y} > g_i$, $6 \leq y \leq n-i$, 则称 I 为双蘑菇, 简记为 DM(double mushroom). g_i 称为 DM 的左端点, g_{i+5} 称为 DM 的右端点. 双蘑菇的排列图结构如图 2, 虚线表示可能与双蘑菇之外的顶点关联的弧.

若两个双蘑菇 $G_1=[g_i, g_{i+1}, \dots, g_{i+5}]$ 和 $G_2=[g_j, g_{j+1}, \dots, g_{j+5}]$ 共享一个蘑菇, 即 $j=i+2$ 或 $i=j+2$, 则称 G_1 与 G_2 相邻. 以下称 G_1 与 G_2 相邻, 则总假设 G_2 位于 G_1 右边, 即 $j=i+2$. 若双蘑菇 G_1 与 G_2 相邻, 且 $g_i < g_j+5$, 则称 G_1 和 G_2 简单相邻, 即 G_1 的左端点没有指向 G_2 右端点的弧. 若 $G_1, G_2, \dots, G_k$ 均为 DM, 且 G_1 与 G_2, G_2 与 $G_3, \dots, G_{k-1}$ 与 G_k 简单相邻, 则称由 $G_1, G_2, \dots, G_k$ 形成的子排列为 π 的一个 DM 阵列, $k \geq 1$. G_1 的左端为 DM 阵列的左端, G_k 的右端为 DM 阵列的右端.

一个 DM 阵列含有偶数个符号. 若 DM 阵列含有 $2l$ 个符号, 则其中含有 $l-1$ 个蘑菇, $l-2$ 个双蘑菇. 我们将一个 DM 阵列中双蘑菇的个数称为该 DM 阵列的长度, 即一个含有 $2l$ 个符号的 DM 阵列长度为 $l-2, l \geq 3$, 长度不小于 2 的 DM 阵列称为长 DM 阵列.

定义 4 在 π 的排列图中, 若短弧 (g_i, g_{i+1}) 满足 $g_{i-x} < g_i, 1 \leq x \leq i-1$, 则称 (g_i, g_{i+1}) 是左无关的; 若 $g_{i+1} < g_{i+y}, 2 \leq y \leq n-i$ 则称 (g_i, g_{i+1}) 是右无关的. 称一条左右均无关的短弧为无关短弧.

显然, 长度为 k 的 DM 阵列中, 有 $k+2$ 条无关短弧.

引理 5 一次正确 Hop 最多产生 2 条无关短弧.

证明 假设 Hop 移动 $\rho(a, bc)$ 是排列 π 中的正确 Hop, 则 $\pi \cdot \rho$ 为 $[\dots, b, c, a, \dots]$ 的形式. 显然符号 a 与符号 c 之间无弧. 因此新产生的无关短弧只能是 a 的右关联弧或者 b 的左关联弧或者 (b, c) , 而 b 的左关联弧和 (b, c) 不同时构成无关短弧.

引理 6 给定排列 $\pi=[g_1, \dots, g_n]$, $I=[g_i, \dots, g_{i+2k+3}]$ 是长度为 k 的 DM 阵列. (a) 若在 π 的排列图 $G(\pi)$ 中存在由 I 左端顶点指向 I 内部顶点的弧 (g_{i-x}, g_y) , 则 $g_y \in \{g_{i+1}, g_{i+3}\}$; (b) 若存在由 I 内部顶点指向 I 右端顶点的弧 $(g_y, g_{i+2k+3+x})$, 则 $g_y \in \{g_{i+2k}, g_{i+2k+2}\}$.

在算法中, 还须用到 9 种特殊的子排列, 它们的排列图如图 3(a)~(i). 若 π 的子排列 I 的排列图与图 3(a) 同构, 则称 I 或 I 的排列图为 α -蘑菇, $\alpha \in \{a, b, c, d, e, f, g, h, i\}$. 不难验证, 若 I 为独立的 α -蘑菇, $\alpha \in \{a, b, c, d, e, f, g, h, i\}$, 则 I 内共有 $3m$ 条弧 ($m=1, 2, 3$), 且消除 I 内弧的最优短块移动序列中, 恰有 m 次 Hop 和 m 次 Skip. 我们还会用到排列图与图 3(a) 同构但并不独立的子排列, 下面的性质给出将一端独立的 α -蘑菇排序所需的移动次数界限, $\alpha \in \{f, g, h, i\}$.

性质 1 若 $I=[g_i, g_{i+1}, \dots, g_{i+5}]$ 为 f -蘑菇, 在 π 中左端独立且 $g_i < g_{i+x}, x \geq 6$, 则消除 I 内部的弧至少需要 1 次 Skip 移动.

对应于图 3(g, h, i) 的排列都拥有相似的性质. 如果枚举上述排列的所有可行的短块移动, 不难验证上述性质的正确性. 设子排列 $\pi=[g_i, \dots, g_j]$ 的左端独立, 如果 $[g_i, \dots, g_{i+7}]$ 为 i -蘑菇且 $g_{i+2} < g_{i+x}, x \geq 8$, 则称 π 的左端为 i -蘑菇; 否则, 如果 $[g_i, \dots, g_{i+5}]$ 为 f -蘑菇且 $g_i < g_{i+x}, x \geq 6$, 则称 π 的左端为 f -蘑菇. 类似于性质 1, 对应图 3(g, h), 可定义 π 的左端为 g -蘑菇或 h -蘑菇. 同样, 可定义 π 的右端为 α -蘑菇, $\alpha \in \{f, g, h, i\}$.

引理 7 设 $I=[a, \dots, b, \dots, x, \dots, c]$ 是 π 的子排列, 若 $b < x < c < a$, 则存在将 π 排序的最优短块移动序列, 其中不存在同时消除弧 (a, b) 和 (a, c) 的 Hop 移动.

证明 (a, b) 和 (a, c) 能被一个 Hop 移动消除, 当且仅当在排序中将 π 变为中间序列 $\pi'=[\dots, a, b, c, \dots]$. 此时若 x 在 π' 中位于 a 之前, 则增加了弧 (x, b) ; 若 x 位于 c 之后, 则增加了弧 (c, x) . 与定理 1 矛盾.

引理 8 设 $J=[g_{i-2}, g_{i-1}, \dots, g_j]$ 为左端独立的子排列, 其中 $I=[g_i, g_{i+1}, \dots, g_j]$ 为长度为 k 的 DM 阵列, $i+2k+3=j$. 若子排列 $[g_{i-2}, g_{i-1}, \dots, g_{i+3}]$ 为 h -蘑菇, 则消除 J 内部的弧至少需要执行 $k+1$ 次 Skip 移动 (证明见附录 A).

推论 1 设 $J=[g_i, \dots, g_j, g_{j+1}, g_{j+2}]$ 为右端独立的子排列, 其中 $I=[g_i, g_{i+1}, \dots, g_j]$ 为长度为 k 的 DM 阵列, $i+2k+3=j$. 若子排列 $[g_{j-3}, g_{j-2}, \dots, g_{j+2}]$ 为 f -蘑菇, 则消除 J 内部的弧至少需要执行 $k+1$ 次 Skip 移动.

$\pi=[g_i, g_{i+1}, \dots, g_j]$ 为左 (右) 端独立的子排列, 如果 $[g_{i+2}, g_{i+3}, \dots, g_{i+2k+5}]$ ($[g_{j-2k-5}, g_{i+1}, \dots, g_{j-2}]$) 为长度为 k 的 DM 阵列且 $[g_i, \dots, g_{i+5}]$ ($[g_{j-5}, \dots, g_j]$) 为 h -蘑菇 (f -蘑菇), 则称 $[g_i, \dots, g_{i+2k+5}]$ ($[g_{j-2k-5}, \dots, g_j]$) 为 h -DM(k)(DM(k)- f) 子排列, 称 π 的左 (右) 端为 h -DM(k)(DM(k)- f) 子排列. 如果 π 的左端和右端分别为 h -DM(k) 子排列和 (DM(k)- f) 子排列, 且 $j=i+2k+7$, 则称 π 为 h -DM(k)- f 子排列.

推论 2 设 $J=[g_i, g_{i+1}, \dots, g_j]$, $i+2k+3=j$ 为长度为 k 的 DM 阵列, 若 J 在 π 中独立, 则消除 J 内部的弧至少需要 $k+1$ 次 Skip 移动; 若 J 在 π 中左 (右) 端独立, 则消除 J 内部的弧至少需要 k 次 Skip 移动.

若 DM 阵列在最小独立子排列 π 的左 (右) 端独立, 则称 π 的左 (右) 端为 DM.

引理 9 设 $J=[g_i, g_{i+1}, \dots, g_j]$, $i+2k+3=j$ 为长度为 k 的 DM 阵列, $k \geq 2$. 若该阵列两端均不独立, 则消除 J 内部的弧至少需要 $k-1$ 次 Skip 移动.

引理 9 的证明与引理 8 类似, 都是对蘑菇阵列的长度进行归纳证明.

定理 3 最小独立子排列 $\pi=[g_1, g_2, \dots, g_n]$ 短块移动排序距离的下界为

$$BK_3(\pi) \geq \left\lceil \frac{m-l-r-\sum_{i=1}^s(k_i-1)}{2} \right\rceil + l + r + \sum_{i=1}^s(k_i-1), \quad (3)$$

其中, m 表示 $G(\pi)$ 中的边数. π 中含有 s 个 DM 阵列, 其长度依次为 $k_1, k_2, \dots, k_s$. 对 (3) 式中的 l 和 r 的取值分析如下:

- (1) 如果 π 为 i- 蘑菇, 则 $l+r=3$;
- (2) 否则如果 π 的左 (右) 端为 h-DM(k)(DM(k)-f) 子排列或 i- 蘑菇, 则 $l=2(r=2)$;
- (3) 否则如果 π 的左 (右) 端为 α - 蘑菇 ($\alpha \in \{f, g, h\}$) 或 DM, 则 $l=1(r=1)$;
- (4) 其他情况下 $l=0(r=0)$.

我们称定理 3 中的 $\sum_{i=1}^s(k_i-1) + l + r$ 次 Skip 为必要的 Skip.

5.2 算法近似性能比证明

设算法 Short-Block-Move($\cdot$) 在第 t 轮找到的关联伞为 $U_k[t], U_{k-1}[t], \dots, U_1[t], U_{i+1}[t]$ 是 $U_i[t]$ 的关联伞, $1 \leq i \leq k-1$, 显然 $U_1[t], \dots, U_{k-1}[t]$ 均为奇 MSU. $g_j[U_i[t]]$ 表示 $U_i[t]$ 的第 j 个伞内符号, $1 \leq j \leq s[U_i[t]]$, $h_i[t] = g_0[U_i[t]]$ 表示 $U_i[t]$ 的伞头, $l_i[t] = \min(U_i[t])$, $U'_i[t]$ 表示 Sort-All-Related- $U(U_k[t], U_{k-1}[t], \dots, U_1[t])$ 第一阶段完成后得到的以 $h_i[t]$ 为伞头的伞, $2 \leq i \leq k$.

引理 10 对于奇简单伞 $U=[g_i, g_{i+1}, \dots, g_{i+m}]$, 若满足 $g_{i+2} < g_{i+4} < g_{i+1} < g_{i+3}$, 则将 U 变为 MSU 最多产生 2 个新 DM, 且执行的 Hop 数不小于产生的 DM 数.

证明 根据奇伞转变为 MSU 的方法 (见引理 3 的证明), g_{i+2} 的左跳最多产生一条无关短弧, 最多产生一个新 DM. 之后 g_{i+4} 的左跳也最多产生一条无关短弧, 最多产生一个新 DM. 所以最多产生 2 个 DM, 至少执行了 2 次 Hop.

定理 4 算法 Short-Block-Move($\cdot$) 的每轮如果剩余一条弧未被消除, 则最多产生 2 个新的 DM; 否则最多产生 1 个新的 DM.

证明 如果第 t 轮结束时没有剩余弧, 则最多产生 1 条新的无关短弧, 最多产生 1 个新的 DM. 否则, 最多产生 2 条新的无关短弧, 显然最多生成 2 个新 DM.

引理 11 如果第 t 轮产生 2 个 DM, 则 $U_k[t]$ 为零伞且 $U'_k[t]=U_1[t+1]$.

证明 若 $U_k[t]$ 为奇伞, 则第 t 轮剩余弧 $(g_1[U_k[t]], g_2[U_k[t]])$, 要产生 2 个 DM 必需产生 2 条无关短弧, 则一定是 $(g_1[U_k[t]], g_2[U_k[t]])$ 和 $U'_k[t]$, 但这 2 条独立短弧不构成蘑菇, 不可能产生 2 个 DM, 从而 $U_k[t]$ 必为偶伞. 此时两条弧中右面的一定是 $U'_k[t]$, 该弧由 $h_k[t]$ 指向另一个符号, 且剩余子排列中不存在其他 $h_i[t]$ 的右关联弧. 又 2 条独立短弧构成蘑菇, 观察 (2) 式中, 左边的独立短弧一定是 $l_{k-1}[t]$ 的左关联弧, 故 $U_k[t]$ 为零伞. 根据 DM 的定义 $h_k[t]$ 无左关联弧, 则 $U_1[t+1]$ 即为该无关短弧.

引理 12 如果第 t 轮产生 1 个新的 DM 且 $U_k[t]$ 奇伞, 则 $U_1[t+1]=[g_1[U_k[t]], g_2[U_k[t]]]$.

证明 由于 $U_k[t]$ 为奇伞, 算法必执行 $\rho(g_0[U_k[t]], g_1[U_k[t]], g_2[U_k[t]])$, 剩余右无关弧 $(g_1[U_k[t]], g_2[U_k[t]])$, 要生成 DM 则 $U_1[t+1]=[g_1[U_k[t]], g_2[U_k[t]]]$.

引理 13 如果 $U'_k[t]$ 为 Lone 且第 t 轮产生 1 个新的 DM, 则 $s[U_1[t+1]] \leq s[U_k[t]]+1$.

证明 显然, 如果 $U'_k[t]$ 为 Lone, 要产生新的 DM, $U_1[t+1]$ 中的所有伞内符号都是第 t 轮中的符号. 由 (2) 式, $h_k[t]$ 左边只有 $s[U_k[t]]+1$ 个符号, 从而 $s[U_1[t+1]] \leq s[U_k[t]]+1$.

引理 14 设算法一轮得到由右到左的关联伞 $U_1, U_2, \dots, U_k$, 若 U_k 为偶伞, 则算法该轮最多执行 $k-1$ 个 Skip.

证明 显然算法 Sort-Related-U($U_k, U_{k-1}, \dots, U_1$) 不执行 Skip, Sort-Related-U($U_k, U_{k-1}, \dots, U_1$) 执行完毕, 得到排列 μ_1 如 (2) 式. 由 μ_1 的性质 (2), μ_1 中最多含有 $k-1$ 个伞, 消除每个伞中的弧最多执行一次 Skip, 至多需要 $k-1$ 个 Skip.

引理 15 给定一组由右到左的关联伞 $U_1, U_2, \dots, U_k, U_1, U_2, \dots, U_{k-1}$ 均为奇 MSU. 如果: (1) 排列 $[U_k, \dots, U_1]$ 中不存在长 DM 阵列; (2) 执行 Sort-Related-U($U_k, \dots, U_1$) 后得到的排列中, 含有两个由连续伞头 $h_{i-1}, h_i (4 \leq i \leq k-1)$ 形成的 Lone, 则存在 $j \in \{i-2, i-1, i\}, s[U_j] \geq 3$. 即连续三个关联伞中至少有一个伞长度不小于 3, 消除它的弧至少执行一次 Hop.

证明 当执行 Sort-Related-U($U_k, \dots, U_1$) 完毕时, 序列变成 μ_1 如 (2) 式, $l_k, re(U_k), l_{k-1}, h_k, re(U_{k-1}), \dots, l_1, h_2, re(U_1), h_1$. 显然, Lone 只能有两种形式: (1) (h_i, l_{i-2}) ; (2) $(h_i, \min(re(U_{i-1})))$. μ_1 含有 Lone: (h_i, l_{i-2}) , 说明 $s[U_{i-1}] = 1$; μ_1 含有 Lone: $(h_i, \min(re(U_{i-1})))$ 则说明 $s[U_{i-1}] \geq 3$, 这是因为 $re(U_{i-1})$ 不为空, 且 U_{i-1} 为奇 MSU. 如果在 μ_1 中含有 h_{i-1} 和 h_i 形成的 Lone (h_{i-1}, l_{i-3}) 和 (h_i, l_{i-2}) , 则 $s[U_{i-2}] = s[U_{i-1}] = 1$. 若此时有 $s[U_i] = 1$, 则原始序列中存在长度为 2 的 DM 阵列 $[U_i, U_{i-1}, U_{i-2}, h_{i-3}, l_{i-3}]$. 因此, $U_{i-2}U_{i-1}U_i$ 中至少有一个伞长度不小于 3, 消除它的弧至少执行一次 Hop.

引理 16 给定一组由右到左的关联伞 $U_1, U_2, \dots, U_k, U_1, U_2, \dots, U_{k-1}$ 均为奇 MSU. 执行 Sort-Related-U($U_k, \dots, U_1$) 得到的序列 μ_1 . 如果 (1) 两个连续的伞头 h_{i-1}, h_i 在 μ_1 中都有右关联弧; (2) h_{i-1}, h_i 不同时形成 Lone. 则在 Sort-All-Related-U($U_k, \dots, U_1$) 的第二阶段, 至少执行一次 h_{i-1} 或 h_i 或 $h_{i+1} (3 \leq i \leq k-1)$ 的 right-Hop.

证明 如果 h_{i-1} 形成 Lone, 则或者 $s[U'_{i+1}] \geq 2$, 算法执行 h_{i+1} 的 right-Hop; 或者 $s[U'_i] \geq 2$, 算法执行 h_i 的 right-Hop. 如果 h_i 产生 Lone, 则 h_{i-1} 不产生 Lone, $s[U'_{i-1}] \geq 2$, 算法执行 h_{i-1} 的 right-Hop.

定理 5 给定一组关联伞 $U_1, U_2, \dots, U_k, U_1, U_2, \dots, U_{k-1}$ 均为奇 IMSU, U_k 为偶伞. 如果排列 $I = [U_k, \dots, U_1]$ 中不存在长 DM 阵列, 设算法 Short-Block-Move(I) 执行的 Hop 数为 C_{hop} , 执行的 Skip 数为 C_{skip} , 则 $C_{hop} - C_{skip} \geq \lfloor \frac{k-2}{3} \rfloor + 1$ (证明见附录 B).

注意, 在定理 5 的证明中, 我们假设 $|H_k| = 0$ 和 $|S_k| = 1$, 此时 $C_{hop} - C_{skip}$ 最小.

推论 3 给定一组关联伞 $U_1, U_2, \dots, U_k, U_1, U_2, \dots, U_{k-1}$ 均为奇 MSU, U_k 为偶伞. 如果排列 $I = [U_k, \dots, U_1]$ 中不存在长 DM 阵列, 且 I 的右端不为 DM、f- 蘑菇, 设算法 Short-Block-Move(I) 执行的 Hop 数为 C_{hop} , 执行的 Skip 数为 C_{skip} , 则 $C_{hop} - C_{skip} \geq \lfloor \frac{k-1}{3} \rfloor + 1$.

推论 4 给定一组关联伞 $U_1, U_2, \dots, U_k, U_1, U_2, \dots, U_{k-1}$ 均为奇 MSU, U_k 为奇伞. 如果排列 $I = [U_k, \dots, U_1]$ 中不存在长 DM 阵列, 且 I 的左端不为 DM、f- 蘑菇, 设算法 Short-Block-Move(I) 执行的 Hop 数为 C_{hop} , 执行的 Skip 数为 C_{skip} , 则 $C_{hop} - C_{skip} \geq \lfloor \frac{k-2}{3} \rfloor + 1$.

推论 5 给定一组关联伞 $U_1, U_2, \dots, U_k, U_1, U_2, \dots, U_{k-1}$ 均为奇 MSU, U_k 为偶伞. 如果排列 $I = [x, U_k, \dots, U_1]$ 中不存在长 DM 阵列 (其中 x 是 h_k 的左邻), 且 I 的左端不为 h- 蘑菇、g- 蘑菇, 设算法 Short-Block-Move(I) 执行的 Hop 数为 C_{hop} , 执行的 Skip 数为 C_{skip} , 则 $C_{hop} - C_{skip} \geq \lfloor \frac{k-2}{3} \rfloor + 1$.

上述 3 个推论的证明仅仅是在定理 5 的基础上进行更细致的分析.

给定一组关联伞 $U_1, U_2, \dots, U_k, U_1, U_2, \dots, U_{k-1}$ 均为奇 MSU. 如果子排列 $\mu = [U_k, \dots, U_1]$ 中存在 DM 阵列由右至左依次为 $D_1, D_2, \dots, D_r$, 长度依次为 $q_1, q_2, \dots, q_r (q_i \geq 2, 1 \leq i \leq r)$. 根据引理 9, 至少需要 $q_i - 1$ 个 Skip 才能消除 D_i 中的弧. 设 D_i 对应的 $q_i + 2$ 个伞的伞头由右至左依次为 $h_{j_i}, h_{j_i+1}, \dots, h_{j_i+q_i+1}$. 称消除 D_i 中的弧必须执行的 $q_i - 1$ 个 Skip 是必要的, 注意我们并不关心这 $q_i - 1$ 个必要的 Skip 由哪些符号参与. 在算法 Sort-All-Related-U($U_k, \dots, U_1$) 第二阶段, $h_{j_i+2}, \dots,$

$h_{j_i+q_i}$ 均执行了 Skip, 因此我们规定 $S_p=\emptyset$ ($j_i+2 \leq p \leq j_i+q_i$), 即 $|H_p|-|S_p|=1$ ($j_i+2 \leq p \leq j_i+q_i$).

对子排列 $\mu=[U_k, \dots, U_1]$ 做如下压缩变换:

(1) 删除 $[h_{j_i+q_i}, \dots, l_{j_i+2}]$, 此时只删除 D_i 中的边;

(2) 将 l_{j_i} 替换为 $l_{j_i+q_i-1}$, 将 l_{j_i+1} 替换为 $l_{j_i+q_i}$.

这样将 D_i 替换为长度为 1 的 DM, 同时关联伞中减少了 q_i-1 个伞, 并且 D_i 与排列中其他符号之间的弧保持不变. 对 $D_1, D_2, \dots, D_r$ 都做上述变换, 得到的新排列 ν , ν 中无 DM 阵列. 显然 Short-Block-Move(ν) 比 Short-Block-Move(μ) 少执行 $\sum_{i=1}^r (q_i-1)$ 个 Hop 和 $\sum_{i=1}^r (q_i-1)$ 个必要的 Skip.

在算法 Short-Block-Move($\cdot$) 中, 寻找关联伞序列 $U_1, U_2, \dots, U_k$, 并将 $U_1, U_2, \dots, U_{k-1}$ 变为 MSU, 可能产生新的 DM. 每个新的 DM 可能与原有的 DM 构成新的长度为 2 的 DM 阵列, 或者使原有的 DM 阵列的长度增加 1. 由引理 10 知, 每产生 1 个新的 DM 都至少执行了一次 Hop. 如果产生新的 DM 为 X_i , 执行的 Hop 为 ρ , X_i 对应的三个伞的伞头为 h_i, h_{i-1}, h_{i-2} , 则规定 $\rho \in H_i$, 因而满足 $|H_i|-|S_i|=1$. 对 $\mu=[U_k, \dots, U_1]$ 进行压缩变换得到子排列 ν , 则对于任意被删除的伞头 h_j , 有 $|H_j|-|S_j|=1$, ν 中不存在 DM 阵列.

如果子排列 $\mu=[x, U_k, \dots, U_1]$ 的右 (左) 端为 DM 或 α -蘑菇 (其中 x 是 h_k 的左邻或者 x 为空, $\alpha \in \{f, g, h\}$), 则可对进行相应的压缩变换得到排列 ν , Short-Block-Move(ν) 比 Short-Block-Move(μ) 少执行 1 个 Hop 和 1 个必要的 Skip. $\mu=[x_1, x_2, \dots, x_6, \dots, y_1, y_2, \dots, y_6]$, 如果 μ 右端独立, 如果 $[y_1, \dots, y_6]$ 构成 DM, 删除 y_1, y_2, y_6 替换为 y_4, y_4 替换为 y_5 ; 如果 $[y_1, \dots, y_6]$ 构成 f -蘑菇, 删除 y_1, y_2, y_5 替换为 y_4, y_4 替换为 y_2 . 如果 μ 左端独立. 如果 $[x_1, \dots, x_6]$ 构成 h -蘑菇, 删除 x_5, x_6, x_2 替换为 x_3, x_3 替换 x_5 ; 如果 $[x_1, \dots, x_6]$ 构成 g -蘑菇, 删除 x_2, x_5 及他们的关联弧; 如果 $[x_1, \dots, x_6]$ 构成 DM, 删除 x_1, x_2 及他们的关联弧. 因此有如下定理.

定理 6 给定一组关联伞 $U_1, U_2, \dots, U_k, U_1, U_2, \dots, U_{k-1}$ 均为奇 MSU. 设算法 Short-Block-Move($\cdot$) 执行的 Hop 数为 C_{hop} , 执行的不必要的 Skip 数为 C_{skip} , 则 $C_{\text{hop}} - C_{\text{skip}} \geq \lfloor \frac{k-2}{3} \rfloor + 1$.

为方便, 如果最后一轮得到的关联伞为 Lone, 则将消除该 Lone 的 Skip 记入其前一轮, 因此不妨设算法 Short-Block-Move($\cdot$) 每一轮消除的弧都不是 Lone. 设算法共执行 s 轮, 第 t 轮的子排列记为 $\mu[t]$, 其中含有 q_t 个关联伞, 并 t 轮执行的 Hop 操作次数为 $C_{\text{hop}}[t]$, 不必要的 Skip 操作次数为 $C_{\text{skip}}[t]$. 对于任意子排列 μ , $C_{\text{hop}}(\mu)$ 和 $C_{\text{skip}}(\mu)$ 分别表示算法 Short-Block-Move(μ) 执行的 Hop 次数和 unnecessary 的 Skip 次数, 则 $C_{\text{hop}}(\pi) = \sum_{t=1}^s C_{\text{hop}}[t]$, $C_{\text{skip}}(\pi) = \sum_{t=1}^s C_{\text{skip}}[t]$. 我们证明对于任意的最小独立子排列 π , $\frac{C_{\text{hop}}(\pi)}{C_{\text{skip}}(\pi)} \geq \frac{4}{3}$, 从而算法的近似性能比不大于 $\frac{14}{11}$.

引理 17 若算法第 t 轮得到的关联伞中不含有前一轮新产生的 DM, $2 \leq t \leq s-1$, 则 $\frac{C_{\text{hop}}[t]}{C_{\text{skip}}[t]} \geq \frac{4}{3}$.

证明 由定理 6 知, $C_{\text{hop}}[t] - C_{\text{skip}}[t] \geq \lfloor \frac{q_t-2}{3} \rfloor + 1$, 由引理 18 知 $C_{\text{skip}}[t] \leq q_t - 1$. 因此,

$$\frac{C_{\text{hop}}[t]}{C_{\text{skip}}[t]} \geq 1 + \frac{1 + \lfloor \frac{q_t-2}{3} \rfloor}{q_t - 1} \geq \frac{4}{3}. \quad (4)$$

引理 18 若算法第 s 轮得到的关联伞中不含有前一轮新产生的 DM, 则 $\frac{C_{\text{hop}}[1] + C_{\text{hop}}[s]}{C_{\text{skip}}[1] + C_{\text{skip}}[s]} \geq \frac{4}{3}$.

证明 由推论 3 及定理 6 知, $C_{\text{hop}}[1] - C_{\text{skip}}[1] \geq \lfloor \frac{q_1-1}{3} \rfloor + 1$, 又 $C_{\text{skip}}[1] \leq q_1 - 1$. 由推论 4 和 5 及定理 6 知, $C_{\text{hop}}[s] - C_{\text{skip}}[s] \geq \lfloor \frac{q_s-2}{3} \rfloor + 1$, 又 $C_{\text{skip}}[s] \leq q_s$. 因此有

$$\frac{C_{\text{hop}}[1] + C_{\text{hop}}[s]}{C_{\text{skip}}[1] + C_{\text{skip}}[s]} \geq 1 + \frac{1 + 1 + \lfloor \frac{q_1-1}{3} \rfloor + \lfloor \frac{q_s-2}{3} \rfloor}{q_1 + q_s - 1} \geq \frac{4}{3}. \quad (5)$$

定理 7 对于任意的最小独立子排列 π , 算法 Short-Block-Move(π) 执行的不必要的 Skip 数为 $C_{\text{skip}}(\pi)$, Hop 操作数为 $C_{\text{hop}}(\pi)$, 则 $\frac{C_{\text{hop}}(\pi)}{C_{\text{skip}}(\pi)} \geq \frac{4}{3}$.

证明 由引理 10 知, 第 t 轮最多使第 $t+1$ 轮增加 2 个新 DM. $\mu[t+1]$ 有 4 种情况 ($1 \leq t \leq s-1$):

- (a) $\mu[t+1]$ 有两个新产生的 DM (如引理 11 所述);
- (b) $\mu[t+1]$ 有一个新产生的 DM, $s[U_1[t+1]] \geq 3$. (如引理 13 所述);
- (c) $\mu[t+1]$ 有一个新产生的 DM, 并且该 DM 恰为 $\mu[t+1]$ 右端的 6 个符号 (如引理 12 所述);
- (d) $\mu[t+1]$ 无新产生的 DM (引理 17 和 18 已经证明).

为方便描述, $C_{\text{hop}}(\phi(t))$ 和 $C_{\text{skip}}(\phi(t))$ 分别表示第 t 轮执行的 $|h_{q_t}[t]|$ 的 Hop 数和 Skip 数. $C_{\text{hop}}(\delta(t))$ 和 $C_{\text{skip}}(\delta(t))$ 表示对 $\mu[t+1]$ 进行压缩变换后少执行的 Hop 数和 Skip 数. $C_{\text{hop}}(\mu'[t])$ 和 $C_{\text{skip}}(\mu'[t])$ 分别表示对 $\mu'[t]$ 排序执行的 Hop 移动次数和 Skip 移动次数, 两者总计作 $C(\mu'[t])$. 显然, $C(\mu'[1]) = C(\mu[1])$, $C(\mu[t]) = C(\mu'[t]) + C(\delta(t-1))$ 则算法中的短块移动操作序列可描述为

$$C(\mu'[1]), C(\delta(1)), C(\mu'[2]), C(\delta(2)), \dots, C(\mu'[s-1]), C(\delta(s-1)), C(\mu'[s]). \quad (6)$$

对任意的 t , 如果 $C_{\text{skip}}(\phi(t))=0$, 则在对 $\mu'[t]$ 排序过程中, 执行 $l_{q_t-1}[t]$ 的 left hop 以后, 将 $\mu'[t]$ 变为 $\mu''[t] = [U_{q_t-1}[t], U_{q_t-2}[t], \dots, U_1[t]]$ 其中 $U_{q_t-1}[t]$ is even. 指定该 hop 属于 $C_{\text{hop}}(\phi(t))$, 并将 $\mu'[t]$ 替换为 $\mu''[t]$. 但是, 如果 $C_{\text{skip}}(\phi(t))=1$, 则 $\mu''[t]=\mu'[t]$. 显然在 $\mu'[t]$ 中不存在前一轮新产生的 DM, 由引理 17 和 18 知,

$$\frac{\sum_{t=1}^{s-1} \{C_{\text{hop}}(\mu'[t]) - C_{\text{hop}}(\phi(t))\} + C_{\text{hop}}(\mu'[s])}{\sum_{t=1}^{s-1} \{C_{\text{skip}}(\mu''[t])\} + C_{\text{skip}}(\mu'[s])} \geq \frac{4}{3}. \quad (7)$$

我们证明在第 t 轮为 (a)-(d) 中任何一种情况时, 在 (7) 式左端的分子中加上 $C_{\text{hop}}(\delta(t))$ 和 $C_{\text{hop}}(\phi(t))$, 分母中加上 $C_{\text{skip}}(\delta(t))$ 都能保持 (7) 式成立.

(a) 由引理 11 知 $C_{\text{hop}}(\phi(t))=1$, $C_{\text{skip}}(\phi(t))=0$. $C_{\text{hop}}(\delta(t))=2$, $C_{\text{skip}}(\delta(t))=2$. 显然能保证 (7) 式成立.

(b) 如果 $C_{\text{skip}}(\phi(t))=1$ 由引理 13 知 $C_{\text{hop}}(\phi(t)) \geq 1$; 如果 $C_{\text{skip}}(\phi(t))=0$, 由我们的指定知 $C_{\text{hop}}(\phi(t))=1$, $C_{\text{hop}}(\delta(t))=1$, $C_{\text{skip}}(\delta(t))=1$. 显然能保证 (7) 式成立.

(c) $C_{\text{hop}}(\delta(t))=1$, $C_{\text{skip}}(\delta(t))=1$.

(c.1) $1 \leq t \leq s-2$ 由定理 6, $C_{\text{hop}}(\mu'[t+1]) - C_{\text{hop}}(\phi(t+1)) - C_{\text{skip}}(\mu'[t+1]) \geq \lfloor \frac{q_{t+1}-3}{3} \rfloor + 1$, 由引理 14 知 $C_{\text{skip}}(\mu'[t+1]) + C_{\text{skip}}(\phi(t+1)) \leq q_{t+1} - 3$.

因此, $\frac{C_{\text{hop}}(\mu'[t+1]) - C_{\text{hop}}(\phi(t+1))}{C_{\text{skip}}(\mu'[t+1])} \geq 4/3$. 能保证 (7) 式成立.

(c.2) $t = s-1$, 由推论 3, $C_{\text{hop}}(\mu'[1]) - C_{\text{skip}}(\mu'[1]) \geq \lfloor \frac{q_1-1}{3} \rfloor + 1$, 由引理 14, $C_{\text{skip}}(\mu'[1]) \leq q_1 - 1$.

所以, $\frac{C_{\text{hop}}(\mu'[1]) + C_{\text{hop}}(\delta(s-1))}{C_{\text{skip}}(\mu'[1]) + C_{\text{skip}}(\delta(s-1))} \geq 4/3$. 由定理 6, $C(\delta(t))C_{\text{hop}}(\mu'[s]) - C_{\text{skip}}(\mu'[s]) \geq \lfloor \frac{q_s-3}{3} \rfloor + 1$, 由引理 14, $C_{\text{skip}}(\mu'[s]) \leq q_s - 2$. 因此, $\frac{C_{\text{hop}}(\mu'[s])}{C_{\text{skip}}(\mu'[s])} \geq 4/3$. 显然能保证 (7) 式成立.

(d) $C_{\text{hop}}(\delta(t))=0$, $C_{\text{skip}}(\delta(t))=0$. 显然能保证 (7) 式成立.

综上 (a)~(d), 将所有 $C_{\text{hop}}(\phi(t))$, $C_{\text{hop}}(\delta(t))$ 和 $C_{\text{skip}}(\delta(t))$ 都加入到 (7) 式中可得

$$\frac{C_{\text{hop}}(\pi)}{C_{\text{skip}}(\pi)} \geq \frac{\sum_{t=2}^s \{C_{\text{hop}}(\mu'[t]) + C_{\text{hop}}(\delta(t-1))\} + C_{\text{hop}}(\mu'[1])}{\sum_{t=2}^s \{C_{\text{skip}}(\mu'[t]) + C_{\text{skip}}(\delta(t-1))\} + C_{\text{skip}}(\mu'[s])} \geq \frac{4}{3}. \quad (8)$$

对于最小独立子排列 π , 定理 7 证明 Short-Block-Move(π) 中执行 Hop 数与不必要的 Skip 数之比至少为 4/3, 假设算法共执行 $4m$ 次 Hop 和 $3m$ 次不必要的 Skip, 则共有 $11m$ 条弧被消除, 消除这些弧的任何最优短块移动序列至少包含 $\lceil 11m/2 \rceil$ 次操作, 从而算法 Short-Block-Move($\cdot$) 的近似性能比为 $\frac{4m+3m}{\lceil 11m/2 \rceil} \leq 14/11$.

6 结论

短块移动是块移动的一种特殊形式. 本文设计了近似性能比为 $14/11$ 的短块移动排序新算法. 首先给出伞型子排列短块移动排序的精确算法, 然后给出关联的伞型子排列短块移动排序的贪心算法. 我们证明每组关联伞子排列短块移动排序中 Hop 移动次数与不必要的 Skip 移动次数比值不小于 $4/3$, 因而证明了算法的近似性能比为 $14/11$. 为证明排序中 Hop 移动次数与不必要的 Skip 移动次数比值不小于 $4/3$, 我们分析了 10 种特殊子排列的短块移动排序性质, 证明了它们短块移动距离的新下界. 算法时间复杂度为 $O(n^4)$, n 为排列中元素个数.

短块移动排序问题的复杂性至今未知. 本文中提出的伞结构对于设计精确算法或者近似度更好的算法有可能有重要的作用. 作者猜想对 DM 阵列的深入研究有助于证明该问题的复杂性.

参考文献

- 1 Palmer J D, Herbon L A. Tricircular mitochondrial genomes of brassica and raphanus: reversal of repeat configurations by inversion. Nucl Acid Res, 1986, 14: 9755–9764
- 2 Hoot S B, Palmer J D. Structural rearrangements, including parallel inversions within the chloroplast genome of anemone and related genera. J Mocol Evol, 1994, 38: 27–281
- 3 Sankoff D, Leduc G, Antoine N, et al. Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. In: Proceedings of the National Academy of Science of United States of America, 1992, 89. 6575–6579
- 4 Bafna V, Pevzner P. Genome rearrangements and sorting by reversals. SIAM J Comput, 1996, 25: 272–289
- 5 Hannenhalli S, Pevzner P. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J ACM, 1999, 46: 1–27
- 6 Caprara A. Sorting permutations by reversals and Eulerian cycle decompositions. SIAM J Discr Math, 1999, 12: 91–110
- 7 Christie D. A $3/2$ -approximation algorithm for sorting by reversals. SODA, 1998, 244–252
- 8 Berman P, Hannanhalli S, Karpinski M. 1.375 -approximation algorithm for sorting by reversals. ESA, 2002, 200–210
- 9 Hannanhalli S. Polynomial time algorithm for computing translocation distance between genomes. Discr Appl Math, 1996, 71: 137–151
- 10 Wang L, Zhu D, Liu X, et al. An $O(n^2)$ algorithm for signed translocation. J Comput Syst Sci, 2005, 70: 284–299
- 11 Zhu D, Wang L. On the complexity of unsigned translocation distance. Theor Comput Sci, 2006, 352: 322–328
- 12 Bafna V, Pevzner P. Sorting by transpositions. SIAM J Discr Math, 1998, 11: 224–240
- 13 Hartman T, Shamir R. A simpler 1.5 -approximation algorithm for sorting by transpositions. CPM, 2003, 156–169
- 14 Elias I, Hartman T A. 1.375 -approximation algorithm for sorting by transpositions. WABI, 2005, 204–215
- 15 Feng J, Zhu D. Faster algorithms for sorting by transpositions and sorting by block-interchanges. TAMC, 2006, 128–137
- 16 Heath L S, Vergara J P C. Sorting by short block-moves. Algorithmica, 2000, 28: 323–354
- 17 Heath L S, Vergara J P C. Sorting by bounded block-moves. Discr Appl Math, 1998, 88: 181–206
- 18 Jiang H, Zhu D. A $(1+e)$ -approximation algorithm for sorting by short block-moves. CSO, 2009, 1: 580–583

附录 A 引理 8 的证明

证明 对 DM 阵列的长度 k 进行归纳证明.

当 $k=1$ 时, (g_{j-1}, g_j) 可能被 3 种移动消除.

(a) Skip 移动 $\rho([j-1], [j])$. 消除该弧后, $[g_{i-2}, \dots, g_{j-2}]$ 是左端独立的 h-蘑菇. 根据 h-蘑菇的性质, 消除排列 $[g_{i-2}, \dots, g_{j-2}]$ 中的弧至少还需要 1 次 Skip 移动. 消除 J 中弧至少需要 2 次 Skip 移动.

(b) Hop 移动 $\rho([j-1], [j, j+1])$. 不妨设 ρ 消除了弧 (g_{j-1}, g_j) 和弧 (g_{j-1}, g_{j+1}) , 若 $g_{j+1} < g_{i+2}$, 则该移动执行后形成左端独立序列 $[g_{i-2}, g_{i-1}, g_i, g_{i+1}, g_{i+2}, g_{i+3}, g_{i+5}, g_{i+6}]$, 该序列为 i-蘑菇, 根据 i-蘑菇的性质, 消除 J 中的剩余的弧至少需要 2 次 Skip. 若不存在 $x \geq 1$, 使 $g_{j+x} < g_{i+2}$, 则执行 ρ 后序列 $J_1 = [g_{i-2}, g_{i-1}, g_i, g_{i+1}, g_{i+2},$

g_{i+3}, g_{i+5} 为独立的 Woven Double-Strip 序列, 至少需要 2 次 Skip 移动. 若存在 $x \geq 1$ 使 $g_{j+x} < g_{i+2}$, 则 J_1 内只有 (g_{i+2}, g_j) 可与 (g_{i+2}, g_{j+x}) 形成 Hop. 若在 π 排序中未执行消除 (g_{i+2}, g_j) 与 (g_{i+2}, g_{j+x}) 的 Hop, 则在 π 排序中 J_1 相当于独立排列, 至少需要 2 次 Skip 移动. 若在 π 排序中执行了消除 (g_{i+2}, g_j) 与 (g_{i+2}, g_{j+x}) 的 Hop, 则在 π 排序中, J_1 内的弧除 (g_{i+2}, g_j) 外, 均不可能与不在 J_1 内的弧同时以 Hop 操作消除, 而不包含弧 (g_{i+2}, g_j) 的独立子排列 $[g_{i-2}, g_{i-1}, g_i, g_{i+1}, g_{i+2}, g_{i+3}, g_{i+5}]$ 也为 Woven Double-Strip 序列, 至少需要 2 次 Skip 移动.

(c) Hop 移动 $\rho([j-2, j-1], [j])$. 即在 π 排序中, 存在 Hop 移动同时消除弧 (g_{j-1}, g_j) 与 (g_{j-3}, g_j) , 所以若存在 $x \geq 1$ 使 $g_{j+x} < g_{j-1}$, 则不存在 Hop 移动同时消除 J 内的一条弧与弧 (g_{j-1}, g_{j+x}) . 因 J 内只有 (g_{j-3}, g_j) 可与 (g_{j-3}, g_{j+x}) 形成 Hop, 根据引理 7, 也不存在 Hop 移动可同时消除 J 内弧与 (g_{j-3}, g_{j+x}) . 从而, J 内所有的弧都不与 J 外的弧匹配被 Hop 移动消除. 这相当于在排序中将 J 作为独立子排列消除其中的弧, 因 J 为 Woven Double-Strip 序列, 至少需要 3 次 Skip 移动才能消除 J 内弧.

综上, $k=1$ 时引理成立. 下面考虑 $k > 1$ 的情况, 假设 DM 阵列长度不大于 $k-1$ 时引理成立, 我们证明阵列长度为 k 时引理也成立. 仍考虑消除弧 (g_{j-1}, g_j) 的 3 种可能移动.

(a) Skip 移动 $\rho([j-1], [j])$. 则 $I_1=[g_i, \dots, g_{j-2}]$ 是一个长度为 $k-1$ 的 DM 阵列, $J_1=[g_{i-2}, \dots, g_{j-2}]$ 左端独立, 且 $[g_{i-2}, g_{i-1}, \dots, g_{i+3}]$ 为 h-蘑菇. 根据归纳假设, 消除 J_1 内剩余的弧至少需要 k 次 Skip. 因而消除 J 内的弧至少需要 $k+1$ 次 Skip 移动.

(b) Hop 移动 $\rho([j-1], [j, j+1])$. 执行 ρ .

(b-1) 若存在一个 Hop, 同时消除弧 (g_{j-3}, g_j) 与弧 (g_{j-5}, g_j) , 设 $p=\min\{x|x \geq 1, g_{j+x} < g_{j-3}\}$. 由 DM 定义, 在排列图中不存在 g_{j-5} 指向 g_{j+p} 的弧, 由引理 7, 不存在同时消除 (g_{j-3}, g_{j+p}) 和 J 中剩余弧的 Hop 移动. 设 $J_1=[g_{i-2}, \dots, g_{j-2}]$, 则 J_1 内的弧不可能与 J_1 的关联弧匹配为 Hop 移动, 这相当于 J_1 为独立子排列而被排序. 因 J_1 为 Woven Double-Strip 序列, 消除 J_1 内的弧至少需要 $k+1$ 次 Skip.

(b-2) 若存在一个 Hop 移动同时消除弧 (g_{j-3}, g_j) 与 (g_{j-3}, g_{j-2}) , 则该移动将 J 变为 $[g_{i-2}, g_{i-1}, \dots, g_{j-4}, g_{j-2}, g_j, g_{j-3}, g_{j+p}, g_{j-1}]$. 设 $J_1=[g_{i-2}, g_{i-1}, \dots, g_{j-4}, g_{j-2}, g_j]$, 则 J_1 为独立 Woven Double-Strip 序列. 消除 J_1 内的弧至少需要 $k+1$ 次 Skip.

(b-3) 若存在一个 Hop 同时消除弧 (g_{j-3}, g_j) 与 (g_{j-3}, g_{j+p}) , 由引理 7, 弧 (g_{j-5}, g_j) 与 (g_{j-3}, g_{j-2}) 至多有一条可与弧 (g_{j-5}, g_{j-2}) 匹配为 Hop, 即 (g_{j-5}, g_j) 与 (g_{j-3}, g_{j-2}) 至少有一条被 Skip 移动消除. 若 (g_{j-5}, g_j) 被 Skip 消除, 因 $J_1=[g_{i-2}, g_{i-1}, \dots, g_{j-2}]$ 为左端独立的子排列, 其中 $I_1=[g_i, \dots, g_{j-2}]$ 是一个长度为 $k-1$ 的 DM 阵列, 子排列 $[g_{i-2}, g_{i-1}, \dots, g_{i+3}]$ 为 h-蘑菇, 则根据归纳假设, 消除 J_1 内的所有弧至少需要执行 k 次 Skip 移动. 若 (g_{j-3}, g_{j-2}) 被 Skip 消除, 则 Hop 移动消除弧 (g_{j-3}, g_j) 与弧 (g_{j-3}, g_{j+p}) 后, 设 $J_1=[g_{i-2}, g_{i-1}, \dots, g_{j-4}, g_{j-2}, g_j]$, 由 DM 的定义知, g_{j-5} 无右关联弧, 则 J_1 为 Woven Double-Strip 序列, 消除其中的弧至少需要 $k+1$ 次 Skip.

综上, 弧 (g_{j-1}, g_j) 与弧 (g_{j-1}, g_{j+p}) 匹配, 即执行 Hop 移动 $\rho([j-1], [j, j+1])$ 时, 消除 J 中的弧至少需要 $k+1$ 次 Skip.

(c) 若在 π 排序中, 存在 Hop 移动同时消除弧 (g_{j-1}, g_j) 与 (g_{j-3}, g_j) , 由引理 7 及 DM 的定义知, J 中只有 (g_{j-1}, g_j) 与 (g_{j-3}, g_j) 可与 J 的右关联弧匹配为 Hop, 从而 J 中的弧都不能与 J 的右关联弧匹配形成 Hop. 在 π 排序中, J 内的弧是被独立地消除的, 因此可将 J 看作独立的 Woven Double-Strip 序列, 至少需要 $k+2$ 次 Skip.

综上所述, 当 DM 阵列长度为 k 时, 引理成立.

附录 B 定理 5 的证明

证明 设 h_{i-1}, h_i, h_{i+1} 是三个连续关联伞的伞头 ($3 \leq i \leq k-2$), 我们证明算法得到的移动序列至少满足下列条件之一.

- (i) 存在 $j \in \{i-1, i, i+1\}$, 在 Sort-All-Related-U($U_k, \dots, U_1$) 的第二阶段不执行 h_j 的 Skip.
- (ii) 存在 $j \in \{i-1, i, i+1\}$, 在 Sort-All-Related-U($U_k, \dots, U_1$) 中, 至少执行 h_j 的一次 right-Hop.
- (iii) $s[U'_{i+2}]=2$, 即算法至少执行 h_{i+2} 的一次 right-Hop, 不执行 h_{i+2} 的 Skip.

如果 $s[U_{i+1}], s[U_i], s[U_{i-1}]$ 不同时为 1, 则显然 (ii) 成立. 否则, 执行 Sort-Related-U($U_k, \dots, U_1$) 得到排列 μ_1 如 (2) 式, 我们重写如下:

$$\begin{aligned} & l_k, \text{re}(U_k), l_{k-1}, h_k, \text{re}(U_{k-1}), \dots, \\ & l_{i+1}, h_{i+2}, l_i, h_{i+1}, l_{i-1}, h_i, l_{i-2}, h_{i-1}, \text{re}(U_{i-2}), \dots, \\ & l_2, h_3, \text{re}(U_2), l_1, h_2, \text{re}(U_1), h_1. \end{aligned}$$

(a) 根据引理 15, 若 μ_1 中同时含有 h_i, h_{i+1} 形成的 Lone, 则 (ii) 成立. 下面假设 μ_1 中不同时含有 h_i, h_{i+1} 形成的 Lone.

(b) 根据引理 16, 若 μ_1 中不同时含有 h_i, h_{i-1} 形成的 Lone, 则 (i) 或 (ii) 成立.

(c) 综合 (a), (b), 下面假设 μ_1 中 h_i, h_{i-1} 均形成 Lone, h_{i+1} 不形成 Lone, 则 $[h_{i+2}, l_i, h_{i+1}, l_{i-1}]$ 必为蘑菇. 显然 $s[U'_{i+2}]=2$, 算法执行 h_{i+2} 的一次 right-Hop.

综上我们证明了 (i)~(iii) 至少有一条成立.

如果算法的某一轮含有 k 个关联伞 $U_k, \dots, U_1$, 则将该轮执行的 Hop 划分为 k 个集合: $H_i (1 \leq i \leq k)$. 规定:

(1) 算法 Sort-Related-U($U_k, \dots, U_1$) 中的 Hop 移动 $\rho(g_0[U_{i+1}] g_0[U_i], g_1[U_i]) \in H_i$;

(2) Sort-S-U(U_i) 执行的 Hop 属于集合 H_i ;

(3) 上述 (iii) 中消除伞 U'_{i+2} 执行的 Hop 移动 $\rho(h_{i+2}, l_i, l_{i-1}) \in H_{i+1}$;

(4) Sort-S-U(U'_i) 执行的不符合 (iii) 的 Hop 属于集合 H_i .

显然算法只可能执行 $h_2, \dots, h_k$ 的 Skip, 再将 Skip 移动分为 $k-1$ 个集合: $S_i (2 \leq i \leq k)$.

规定: 如果执行 h_i 的 Skip 移动 ρ , 则 $\rho \in S_i$, 否则 $S_i = \emptyset$. 从而 $C_{\text{hop}} = \sum_{i=1}^k |H_i|$; $C_{\text{skip}} = \sum_{i=2}^k |S_i|$.

综合 (i)~(iii) 可知, 在 $h_2, h_3, \dots, h_{k-1}$ 中, 对任意连续的 3 个伞头 h_{i-1}, h_i, h_{i+1} , $|H_{i-1}| + |H_i| + |H_{i+1}| - |S_{i-1}| - |S_i| - |S_{i+1}| \geq 1$.

显然 $|H_1| \geq 1, |S_k| \leq 1, |H_k| - |S_k| \geq -1$. 又因为执行 h_2 的 Skip 消除的弧一定是 $(h_2, \min(\text{re}(U_1)))$, U_1 是奇 MSU, 故 $s[U_1] \geq 3$, 即 $|H_1| \geq 2$. 因此, 若 Short-Block-Move($\cdot$) 执行 h_2 的 Skip, 则 $C_{\text{hop}} - C_{\text{skip}} \geq \lfloor \frac{k-2}{3} \rfloor + 1$; 若不执行 h_2 的 Skip, 当 $k \geq 5$ 时我们证明 $|H_2| + |H_3| + |H_4| - |S_3| - |S_4| \geq 2$. 反证法, 假设 $|H_2| + |H_3| + |H_4| - |S_3| - |S_4| \leq 1$, 显然 $|H_2| = |H_3| = |H_4| = |S_3| = |S_4| = 1$ 则排列中存在长度为 2 的 DM 阵列, 与题设矛盾. 因此当 $k \geq 5$ 时, $|H_2| + |H_3| + |H_4| - |S_3| - |S_4| \geq 2$ 总成立. 而当 $k \leq 4$ 时, 显然 $C_{\text{hop}} - C_{\text{skip}} \geq 1$. 从而, $C_{\text{hop}} - C_{\text{skip}} \geq \lfloor \frac{k-2}{3} \rfloor + 1$.

A 14/11-approximation algorithm for sorting by short block-moves

JIANG HaiTao & ZHU DaMing*

School of Computer Science and Technology, Shandong University, Jinan 250101, China

* E-mail: dmzhu@sdu.edu.cn

Abstract Block-move is one of the popular operations for genome rearrangement. A short block-move is an operation on a permutation that moves an element at most two positions away from its original position. Heath and Vergara investigated the problem of finding a minimum-length sorting sequence of short block-moves for a given permutation and devised a 4/3-approximation algorithm. In this paper, we present a new 14/11-approximation algorithm for this problem. Firstly, we devise an exact polynomial time algorithm for sorting a special kind of sub-permutations called umbrella; then we split the permutation into a series of related umbrellas and sort them greedily. We obtain a new lower bound of the short block-move distance by exploiting the properties of five kinds of sub-permutations. After some complicated analysis, we prove that the approximation ratio of the new algorithm is at most 14/11.

Keywords short block-move, approximation, algorithm