Computational Biology
Lecture 17: Genome rearrangements, sorting by reversals (cont.)
Saad Mneimneh

We continue with the problem of sorting signed permutations by reversal. We first establish a new lower bound on the number of reversals that take the number of hurdles into account. At this point it is beneficial to review the definitions for bad and good components, simple and super hurdles, and a fortress.

A new lower bound

We will show that $d_\beta(\alpha) \geq n+1-c_\beta(\alpha)+h_\beta(\alpha)$. First note that any reversal ρ can "destroy" at most two hurdles. If ρ is defined by two reality edges in the same component (whether hurdle or not), then it only affects that component. If ρ is defined by two reality edges in different components, then: (1) if both are hurldes, no hurdle can separate them and hence only these two hurdles are affected, (2) if non of them is a hurdle, then at most two hurdles can separate them and hence at most two hurdles are affected, (3) if only one of them is a hurdle, then at most one hurdle can separate them hence at most two hurdles are affected. Let $\Delta h = h(\alpha\rho) - h(\alpha)$, then $\Delta h \geq -2$. We define similarly $\Delta c = c(\alpha\rho) - c(\alpha)$ and $\Delta(c-h) = \Delta c - \Delta h$. It is enough to show that $\Delta(c-h) \leq 1$ to prove that $n+1-c_\beta(\alpha)+h_\beta(\alpha)$ is a lower bound (bounding the increase in $c-h$ after every reversal by 1 and using the fact that $c_\beta(\beta) - h_\beta(\beta) = n+1-0 = n+1$).

Proof: Given a reversal ρ, we have three cases.

case 1: $\Delta c = 1$, then ρ acts on a good cycle and $\Delta h = 0$ (no hurdles destroyed), $\Delta(c-h) = 1$.

case 2: $\Delta c = 0$, then ρ acts on a bad cycle and $\Delta h \geq -1$ (at most one hurdle destroyed), $\Delta(c-h) \leq 1$.

case 3: $\Delta c = -1$, then $\Delta h \geq -2$ anyway (at most two hurdles destroyed), $\Delta(c-h) \leq 1$.

Achieving the lower bound

We will achieve the lower bound $n+1-c(\alpha)+h(\alpha)$ for a non-fortress permutation α. Our approach is the following. In every time, we will find a reversal ρ such that $c(\alpha\rho) - h(\alpha\rho) = c(\alpha) - h(\alpha) + 1$. We call such reversal a *safe* reversal. Since $c_\beta(\alpha) - h_\beta(\alpha)$ can be at most $n+1$ (when $\alpha = \beta$); therefore, we will perfrom only $n+1-c(\alpha)+h(\alpha)$ safe reversals, thus achieving the lower bound.

We will define three kinds of safe reversals and show that whenever α is a fortress, is is always possible to determine a safe reversal of some kind that can be performed.

Safe reversal kind I

Reversal: A reversal ρ defined by two divergent reality edges of a good cycle (i.e. in a good component) that does not lead to the creation of bad components.

Safe: $c(\alpha\rho) - c(\alpha) = 1$ and $h(\alpha\rho) - h(\alpha) = 0$. Therefore, $c(\alpha\rho) - h(\alpha\rho) = c(\alpha) - h(\alpha) + 1$.

FACT: if there is a good component, then there exists a safe reversal of kind I (we are not going to prove this).

Safe reversal kind II: Hurdle Merging

Define two opposite hurdles A and B such that the number of hurdles between A and B is the same on either sides of the circle.

Reversal: A reversal ρ defined by two reality edges of two opposite hurdles (i.e. the number of hurdles must be even).

Safe: ρ destroys two hurdles which become a good component with any non-hurdle that separates them. Moreover, ρ does not create new hurdles (because A and B are opposite). Therefore, $c(\alpha\rho) - c(\alpha) = -1$ and $h(\alpha\rho) - h(\alpha) = -2$. Therefore, $c(\alpha\rho) - h(\alpha\rho) = c(\alpha) - h(\alpha) + 1$.

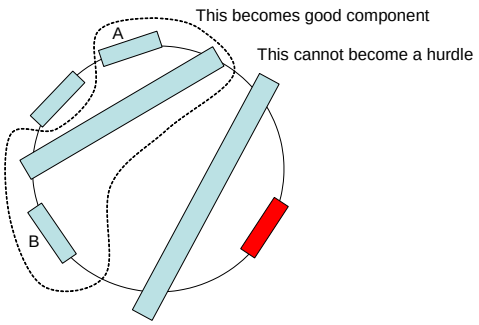

Figure 1: Merging opposite hurdles

Safe reversal kind III: Hurdle Cutting

Reversal: A reversal ρ defined by two convergent reality edges of a bad cycle in a simple hurdle when the number of hurdles in odd.

Safe: ρ destroys the hurdle (makes it a good component), and does not create new hurdles (otherwise the hurdle would be a super hurdle), and results in even number of hurdles. $c(\alpha\rho) - c(\alpha) = 0$ and $h(\alpha\rho) - h(\alpha) = -1$. Therefore, $c(\alpha\rho) - h(\alpha\rho) = c(\alpha) - h(\alpha) + 1$.

If α is not a fortress, then we can always find a safe reversal.

Proof: If there is a good component, then there is a kind I safe reversal (stated above). If there are no good components and the number of hurdles is even, then there is a Kind II safe reversal (hurdle merging). If there are no good components and the number of hurdles is odd, then there is a simple hurdle (otherwise α would be a fortress), and there is a Kind III safe reversal (hurdle cutting). After that point the number of hurdles is always even so there will be always a kind I or kind II safe reversal.

Here's an algorithm for transforming α into β with the minimum number of reversals.

Algorithm

given α and β

if $\alpha \neq \beta$
  then repeat
    if there is a good component in $RD_\beta(\alpha)$
      then pick a Kind I reversal
    else if $h_\beta(\alpha)$ is even
      then pick a Kind II reversal [*merging two opposite hurdles*]
    else if $h_\beta(\alpha)$ is odd
      then pick a Kind III reversal [*cutting simple hurdle*]
  until $\alpha = \beta$

What if α is a fortress? Then we have an odd number of hurdles non of which is simple. We cannot cut a super hurdle since this will definitely create a new hurdle. We can merge two super hurdles (not opposite thought), but there is a danger of creating a new hurdle!

FACT: merging two super hurdles in a fortress with $h > 3$ super hurdles results in a fortress with $h - 2$ super hurdles.

Therefore, merging of two super hurdles creates a new hurdle only when we end up with a 3-fortress (a fortress with 3 super hurdles), resulting in $n + 1 - c_\beta(\alpha) + h_\beta(\alpha) + \mathbf{1}$ reversals for a fortress permutation. This is optimal for a fortress because any fortress has to have one unsafe reversal, but we will not prove it here.

Here's a modified algorithm that works for any permutation α.

Algorithm

given α and β

if $\alpha \neq \beta$
  then repeat
    if there is a good component in $RD_\beta(\alpha)$
      then pick a Kind I reversal
    else if $h_\beta(\alpha)$ is even
      then pick a Kind II reversal [*merging two opposite hurdles*]
    else if $h_\beta(\alpha)$ is odd and there is a simple hurdle
      then pick a Kind III reversal [*cutting simple hurdle*]
    else [*fortress*]
      merge any two super hurdles
      [*this will result in a fortress with* $h_\beta(\alpha) - 2$ *super hurdles if* $h_\beta(\alpha) > 3$]
      [*otherwise, unsafe reversal, but only once when* $h_\beta(\alpha) = 3$]
  until $\alpha = \beta$

This is a polynomial time algorithm but let us analyze it's running time. Constructing $RD(\alpha)$ takes $O(n)$ time. For each cycle we have to determine whether it is good or bad, this takes $O(n)$ time for each cycle for a total of $O(n^2)$ time. Interleaving of cycles can be determined in $O(n^2)$ time by examining every pair of cycles. Determining good and bad components, non-hurdles, simple hurdles, and super hurdles can be done in $O(n)$ time once we have the the good/bad and cycle interleaving information (how?). Therefore, a total of $O(n^2)$ time is needed to determine $RD(\alpha)$ with all the associated information.

The most time consuming part is identifying whether a safe reversal of Kind I exists. Since a reversal is defined by two edges, we have $O(n^2)$ reversals to try. For each reversal ρ we have to see whether a bad component will be created, and this can be done in $O(n^2)$ time as discussed above by computing $RD(\alpha\rho)$. Therefore, we spend $O(n^4)$ time in each step. This is performed at most $O(n)$ times ($d(\alpha) \leq n + 2$) yielding an $O(n^5)$ time algorithm.

References

Setubal J., Meidanis J., Introduction to Computational Molecular Biology, Chapter 7.
Pevzner P., Computational Molecular Biology, Chapter 10.