Finding a best pair of paths for a diploid sequence

Problem: Given parameters ρ, μ and diploid $(0, 1, 2)$ sequence

$$\mathbf{w} = (w_0, w_1, \dots w_{n-1})$$

find best path $\mathbf{p_1}, \mathbf{p_2}$ through the reference array to best 'explain' $\mathbf{w}$. In our previous notation we find $\mathbf{w_1}, \mathbf{w_2}$ such that $\mathbf{w} = \mathbf{w_1} + \mathbf{w_2}$ and also $\mathbf{p_1}, \mathbf{p_2}$ so as to minimize

$$S(\mathbf{p}, \mathbf{w}) = S(\mathbf{p_1}, \mathbf{w_1}) + S(\mathbf{p_2}, \mathbf{w_2})$$

Basic strategy as before
Work from right to left, with branch and bound
Now we place pairs of intervals on the stack
After processing bit w_j we will have a stack of pairs of intervals (A_k^j, B_k^j) and an associated score S_k^j
We then carry out a pruning algorithm discarding intervals which can never yield the best path.

Notation:
A state $\mathcal{S}$ is a pair of intervals S_1, S_2 , often with an associates score $z(\mathcal{S})$
We will write $\mathcal{S} = [S_1, S_2]$ or $\mathcal{S} = [S_1, S_2, z]$
Let

$$\begin{aligned}
\mathcal{S} &= [S_1, S_2] \; and \; if \\
S_1 &= (a, b) \\
S_2 &= (c, d) \; write \\
a(\mathcal{S}) &= a \\
b(\mathcal{S}) &= b \\
c(\mathcal{S}) &= c \\
d(\mathcal{S}) &= d
\end{aligned}$$

We will also write T for the interval $[0, m-1]$ covering the whole set of reference haplotypes

$\mathcal{S}^n$ has one entry $[T, T, 0]$
For bit j, loop on $x = (x_1, x_2)$ where x_1, x_2 are ungarbled bits for the two paths.
(4 values for x)

We set ξ the contribution to the garble (mutation) score
The (obvious) rules: Write $y = x_1 + x_2$, $w = w_j$

1. If $w = y$ set $\xi = 0$
2. Else if w is missing set $\xi = 0$
3. Else if $w = 1$ or $y = 1$ set $\xi = \mu$
4. Else set $\xi = 2\mu$

Work multiplies by 4^t for a run of missing data of length t

For each stack entry $\mathcal{S}_k = [S_{k,1}, S_{k,2}, z_k]$ we compute

$$lfx(a(\mathcal{S}), b(\mathcal{S}), x_1, \&aa, \&bb)$$

and

$$lfx(c(\mathcal{S}), d(\mathcal{S}), x_2, \&cc, \&dd)$$

and if both are feasible add $[[aa, bb], [cc, dd], z_k + \xi]$ to the stack

A refinement:
if $\mathcal{S}$ is symmetric $(S_1 = S_2)$
for hets we can break symmetry
and disallow the case $x_1 = 0, x_2 = 1$.
This straightforward generalization of the haploid case deals with mutations

Recombination is more complicated than in the haploid case
We have 4 possibilities

1. No Jump
2. Jump on $\mathbf{P}_1$
3. Jump on $\mathbf{P}_2$
4. Jump on both paths

Let us discuss a jump on $\mathbf{P}_1$. We will (as we did with the haploid case) have 'horizontal' state transitions where state

$$[U, V, s] \to [T, V, s+\rho]$$

Evidently the best parent of $[T, V]$ is to choose U, V to have minimal score for fixed V
This can be determined (for all V) by a sort
in time $O(k \log k)$ where k is the stack size Thus fix V and define $[U_V, z_V]$ by

$$[U_V, V, z_v]$$

is on the stack and $z_v \leq z \ \forall [U, V, z]$
We then put $[T, V, z_v + \rho]$ on the stack
Of course there is a dual process for a jump on $\mathbf{P}_2$

The double jump is simpler. Let

$$[U_{verybest}, V_{verybest}, z_{verybest}]$$

be the stack entry with minimal score
We then have a transition

$$[U_{verybest}, V_{verybest}, z_{verybest}] \rightarrow [T, T, z_{verybest} + 2\rho]$$

Some diagrams:

No recombination:

$$\begin{tikzcd}
Stack\ (j+1) & {[U, V, z]} \arrow[d, "\xi"] \\
Stack\ (j) & {[U', V', z' = s + \xi]} \arrow[d, "\xi'"] \\
Stack\ (j-1) & {[U'', V'', z'' = z' + \xi']}
\end{tikzcd}$$

Jump on $\mathbf{P}_1$:

$$\begin{tikzcd}
Stack\ (j+1) & {[U, V, z]} \arrow[d, "\xi"] & \\
Stack\ (j) & {[U', V', z' = s + \xi]} \arrow[d, "\psi"] \arrow[r] & {[T, V', z' + \rho]} \arrow[d, "\xi'"] \\
Stack\ (j-1) & {[U'', V'', z'' = z' + \psi]} & {[U''', V''', z''' = z + \rho + \xi']}
\end{tikzcd}$$

Here $U' = V'_{best}$.

Jump on $\mathbf{P}_2$ is dual

Double Jump:

$$\begin{tikzcd}
Stack\ (j+1) & {[U, V, z]} \arrow[d, "\xi"] & \\
Stack\ (j) & {[U', V', z' = s + \xi]} \arrow[d, "\psi"] \arrow[r] & {[T, T, z' + 2\rho]} \arrow[d, "\xi'"] \\
Stack\ (j-1) & {[U'', V'', z'' = z' + \psi]} & {[U''', V''', z''' = z + 2\rho + \xi']}
\end{tikzcd}$$

Here $U' = U_{verybest}, V' = V_{verybest}$.

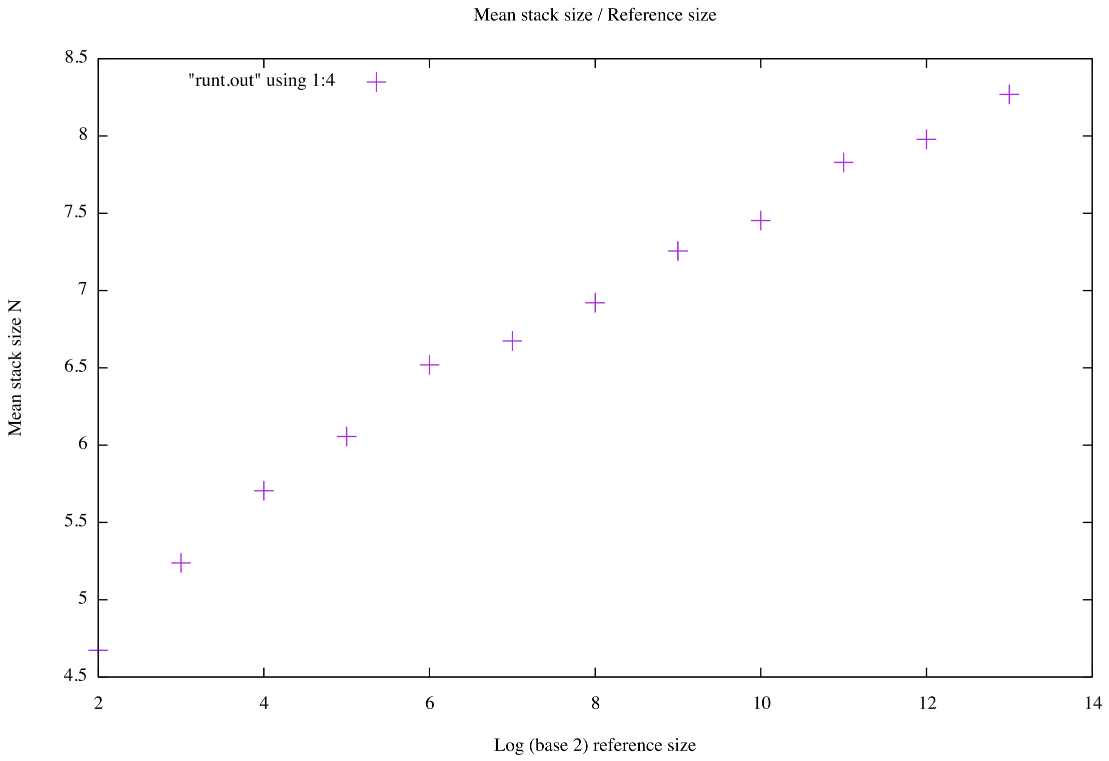
Pruning the stack

In the 1-dimensional case there is a well defined tree we can build for the intervals on the stack
every interval has a largest interval that contains it
The 2-dimensional case is more difficult – my current prototype has heuristics and is not worth a detailed description.
We do describe an issue
Twinning
If a state $[A, B, z]$ is on the stack by symmetry then $[B, A, z]$ is feasible
Call this the *twin*.

1. If a pair of twins are on stack remove a twin (and duplicates)

2. When setting up for pruning, put twins into the pruning list
   A state may be covered by a set of other states (with lower score) *and their twins*

3. Remove twins from pruned stack



Used reference set of $20,000$ phased imputed haplotypes from Ali Akbari

So work is (roughly) logarithmic in reference size
Slightly disappointing – will better pruning solve this?

An interesting Computer Science problem:

In the stack we have rectangles $R_1, R_2, \dots R_t$ with scores $z_1, z_2, \dots, z_t$
Sort R_i in ascending order of z_i.
Now we can remove R_i if

$$R_i \subseteq \cup_{j<i} R_j$$

Thus it is sufficient to compute $m_1, m_2, \dots m_t$ where

$$m_i = \sum_{j=1}^{i} area(\cup R_j)$$

How efficiently can we do this?
Computing m_t is a classic problem (Bennett 1977, Preparata and Shamos)
(Thanks to Harald Ringbauer for pointing this out to me)
and $O(t \log t)$ is optimal but this one seems harder.