Supporting Text 1: Order and Orientation Based on One-to-One Mapping

Perspective. The heaviest common subsequence (HCS) and strict runs or breakpoints have considerable precedent in genomic research, as well as considerable intuitive appeal and utility. The HCS provides a measure of global of correctness. Runs provide information about strictly local correctness. However, for many uses of a genome, an intermediate view may be useful. A very large inversion will affect the HCS in the same way as many smaller ones; these cases are very different in terms of their impact on downstream analyses. On the other hand, it does not discriminate between “near misses” (local discrepancies) and more dramatic displacements. Runs, on the other hand, runs are very sensitive to noise in the one-to-one mapping, or erroneous placement of small pieces, which may give an overly pessimistic view of an assembly’s utility for many purposes.

Thus, some form of “noise tolerant runs” would be helpful. Intuitively, we would like to identify the minimum set of matches such that if we ignore these matches, all remaining matches are in runs of at least a certain size. Once such “clumps” are computed, we may further analyze those matches that were ignored to determine whether they reflect relatively local discrepancies (i.e. are within some fixed distance of a clump) or are more isolated. The program CLUMPMAKER efficiently computes the guaranteed minimal set of matches that must be excluded such that the remaining matches are in “clumps” of at least X bases (X = 50,000 for results shown); clumps are slightly looser than runs in one respect, as they are allowed to overlap on the reference axis (to allow for interleaving due to incomplete assembly on the draft axis), but they are stricter than runs in another sense: consecutive matches in a clump must be separated from one another by no more than Y bases (Y = 500,000 here) on either axis. Results depend slightly on the exact values of arbitrary cutoff parameters, but we have explored the effects of varying the cutoffs and find qualitative agreement with the results. Results using X = 50,000 and Y = 500,000 for comparison of various assemblies against NCBI-34 are shown in Table 8.

CLUMPMAKER Details. A clump is a set of hits consistent with an alignment, possibly reverse-oriented (same orientation, same order), with the further constraint that the distance between adjacent matches within a clump be less than the “maximum jump” parameter, J. A compatible set of clumps is such that they do not (non-trivially) overlap on the query axis, though they may overlap on the reference axis (to handle nested or interleaved pieces). Let an optimal clump set be that set which maximizes total weight less a the number of clumps times a “clump open penalty”, O. (Thus, O is the minimal clump size, i.e., the value such that each clump is of weight at least O.) Not all matches in a one-to-one matching are assigned to a clump; among those matches that aren’t assigned to a clump, we can distinguish between matches near a clump and isolated matches. Matches near but not in clumps are “near misses.”

The O and J parameters make clumps tunably insensitive to noise and O&O discrepancies, but of course this means a given set of results will be somewhat dependent on the choice of O and J. In practice, we observe results to be qualitatively robust; nevertheless, it may be of interest to state certain guarantees with respect to the specific values of the tuning parameters. Increased *J* provably leads to: non-decreasing total weight in clumps; non-increasing number of clumps; and non-decreasing clump size (span and weight alike). Increased *O* provably leads to: non-increasing total weight in clumps; non-increasing number of clumps; and non-decreasing clump size (span and weight alike).

A dynamic program can compute the provably optimal clump set under the objective function $W - O*C$, where W = total weight of matches in clumps, O = clump open penalty, and C = number of clumps. Exact specification of the computation of an optimal clump set is as follows:

```
Sort matches in ascending position on reference axis
Set best = $-O$
Set bestend = -1
Set $t$ = -1 /* ($t$ = trailing pointer) */
Set $r(-1) = -O$ /*$r(i)$ = best score through $i$, $i$ continues a clump */
Set $s(-1) = 0$  /* $s(I)$ = best score through $I$, $I$ starts a clump */
For each match $i$
    While refend($t$)+$J$ < refbegin($i$),
        t++
    mybest = $-O$
    mybestpred = -1
    For $t <= j < I$
        $u = s(t)$ + weight($i$)
        if orientations for $i$, $t$ differ
          or $i$ precedes (follows) $t$ on the query axis if orientation forward
        (resp. rev)
          or the gap between $t$ and $I > J$ on the query axis
        then
          $u$ -= $O$
      if $u$ > mybest($i$) then
          mybest($i$) = $u$
          mybestpred($i$) = $t$

    If mybest($i$) > best then
        best = mybest($i$)
        bestend = $i$
Backtrace from bestend, breaking gaps where $i$ can't extend mybestpred($i$)
```