w2rap - Suplementary Material

Bernardo J. Clavijo$^{1}$, Gonzalo Garcia Accinelli$^{1}$, Jonathan Wright$^{1}$, Darren Heavens$^{1}$, Katie Barr$^{1}$, Luis Yanes$^{1}$, Federica Di-Palma$^{1}$
$1$ Earlham Institute, Norwich Research Park, UK.

# w2rap-contigger processing steps

Each step during contig assembly uses significantly different algorithmic approaches and data. We segmented the w2rap-contigger processing into eight steps which can be run independently thus enabling us to make more efficient usage of resources when running multiple assemblies or sharing computational resources with other projects. This change produced two desired outcomes: (i) each step runs with the resources required for that step only thus avoiding a waste of computing resources on large-memory multi-processor machines and, (ii) the granularity of running shorter steps rather than all steps combined allows for better control over the assembly, and provides the opportunity for a detailed check of results from intermediate steps. These modifications are important when assembling large and complex genomes, where the contigging steps can take over 10 days.

Supplementary Table 1 describes each of the eight steps and their parameters.

**Supplementary Table 1:** w2rap-contigger execution steps

|  |  |  |
| --- | --- | --- |
| Step # | Description | Outputs |
| 1 | Read loading | binary-formatted reads |
| 2 | 60-mer counting and filtering | 60-mer data, kmer spectra |
| 3 | Build small k (k=60) graph from reads | small k graph, read paths |
| 4 | Build large K graph from small k graph and reads | large K graph, read paths |
| 5 | Clean large K graph | large K cleaned graph, read paths |
| 6 | Local assemblies on the large K graph “gaps” | large K completed graph, read paths |
| 7 | Graph simplification and PathFinder | large K simplified graph, read paths, raw/contig-lines GFA and fasta |
| 8 | PE-scale scaffolding across gaps in the large K graph | large K simplified graph with jumps, read paths, raw/lines GFA and fasta |

# Computational Performance of the w2rap-contigger vs. DISCOVAR denovo

## Conditions for the performance analysis

## OpenMP parallel processing vs. internal ad-hoc classes

## General memory usage considerations

## 60-mer counting and disk batches

## Computational gain by correct parametrisation of the assembly

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Process # | *A. thaliana*64t Peak Memory | *A. thaliana* 64t Runtime | *H. sapiens*64t Peak Memory | *H. sapiens* 64t Runtime |
| w2rap Step 1 | 12 GB | 4:52 | 240 GB | 1:59:28 |
| w2rap Step 2 (-d 0) | 110.9 GB | 11:17 |  |  |
| w2rap Step 2 (-d 16) |  |  | 443GB | 17:38:39 |
| w2rap Step 3 | 15.2 GB | 11:13 | 274GB | 6:44:20 |
| w2rap Step 4 | 12.5 GB | 3:16 | 299GB | 1:19:12 |
| w2rap Step 5 | 23.4 GB | 18:02 | 545GB | 26:13:59 |
| w2rap Step 6 | 18.2 GB | 10:36 |  |  |
| w2rap Step 7 | 1.7 GB | 1:20 |  |  |
| w2rap Steps 1-7 |  |  |  |  |
| DISCOVAR *denovo* |  |  |  |  |

**Suplementary table 1:** Peak Memory and Runtime when run with 64 threads on 64 CPUs, and with 128 threads on 128 CPUs on a NUMA system using independent steps of *w2rap-contigger* with default parameters and all steps at once, compared to DISCOVAR *denovo*, for the A.thaliana dataset and the H. sapiens Dataset. See supplementary material for memory usage profiles and further detail on how the software was run. (w2rap-contigger uses gnu malloc, discovar uses jemalloc).



Benefitial effect of correct parametrisation on wall-clock time: the same runs that achieve greater accuracy and contiguity for the *A. thaliana* dataset, show a decrease on computing time. The reason for this is that more of the assembly is solved early by less computing intensive heuristics, decreasing the runtime of following steps.

# Repeat resolution on the w2rap-contigger: PullAparter and PathFinder



Representation of the PullAparter (left) and PathFinder (right) repeat expansion heuristics. The PullAparter is a simple heuristic that finds 2-fold collapsed edges and expands them when read support maps each input to an output. The PathFinder also uses read support to match inputs and outputs, but solves more complex regions.

# GFA support

# Details for the example runs