PEACE: Parallel Environment for Assembly and Clustering of Gene Expression

A Distance metrics, heuristics and filters

A.1 The distance metric

d^2, as outlined by *Hide et al.* [1], is an *alignment free* distance pseudo-metric which quantifies local similarity between sequences based on a simple word count. Let $c_x(w)$ denote the number of times word w occurs in string x. We search for similarity between strings x and y by looking at the difference between $c_x(w)$ and $c_y(w)$ for different words w. Specifically, for all words of a fixed length k, we calculate:

$$d_k^2(x,y) = \sum_{|w|=k} (c_x(w) - c_y(w))^2$$

However, applying such a definition to two sequences as a whole leads to a measure of global similarity, while we want to measure local similarity (thus, for example, assigning two sequences with sufficiently long overlapping ends to be at a distance of zero – or a small distance if errors are present). For this, instead of comparing the entire two strings, we compare sliding windows from each string of a fixed size r. Formally, for sequences x and y ($|x| \geq r$, $|y| \geq r$), we define:

$$d^2(x,y) = \min\left\{d^2(u,v) \ : \ u \sqsubseteq x, v \sqsubseteq y, |u| = |v| = r\right\} \tag{S1}$$

(where $u \sqsubseteq x$ denotes that u is a substring of x). Defined as such, d^2 is, in a mathematical sense, a *pseudo-metric*: $d^2(x,y) = 0$ does not imply $x = y$.

The PEACE implementation of d^2 was initially based on the description from *Hazelhurst* [2]. For parameters we adapted those used by the WCD clustering tool [3]: a word size $k = 6$ and a window size $r = 100$.

A.2 Improved Distance Matrix

Our "two-pass d^2" algorithm works by sampling a subset of window pairs evenly distributed across the sequences, narrowing down a smaller region in which to search for the

best scoring window-pair. In the first pass, the algorithm look at every length r window on one sequence, but on the other sequence we skip the window by s bases between every sampling. It then take the best such window pair, extend each of these two windows by s bases in each direction, and apply (S1) to this limited region.

In PEACE, we use $k = 6$, $r = 100$, and $s = 50$.

A.3 Filtering heuristics

PEACE uses the u/v and t/v heuristics, roughly as described by *Hazelhurst et al.* [3] as filters that allows us to avoid over 99% of the potential d^2 calculations. Each heuristic takes two sequences and estimates whether it is worth proceeding to the d^2 computation by sampling and comparing word frequency across the two sequences. The u/v heuristic looks at every h-th word of size v on one sequence, and rejects if it does not find at least u occurrences of these words on the other sequence. The t/v heuristic demands that there be at least t size v words on one sequence that occur within a l base range within the second sequence. If a sequence pair meets the requirements of both these filters, then we compute the d^2 distance.

In PEACE, we set $h = 16$, $v = 8$, $u = 4$, $t = 65$, and $l = 100$.

B MST-Based Calculations

While d^2 serves to quantify sequence distance, the basis for the clustering algorithm is the minimum spanning tree (MST). Viewing the ESTs as nodes and the data set as an d^2-weighted graph, the derivation of a minimum spanning tree results in the placement of nodes of a given tree into a restricted neighborhood of the graph. By then removing larger edges, we are left with connected components corresponding to the gene-based clusters. The MST-based clustering method has been used effectively in other applications, but to our knowledge this is the first such use for the EST clustering problem [4, 5].

B.1 Calculation of the MST

To calculate this MST we use Prim's algorithm [6]. For a graph of n nodes and e edges, Prim's can be implemented such that the algorithm has an $O(e + n \log n)$ worst-case runtime. However, the *narrow-band* nature of the model allows us to reduce this bound. If we were to model only the edges connecting adjacent nodes, we would find the node degrees to be very small relative to n: any EST overlaps only a few others, and has no connection to a vast majority of the data set members. Since we can quickly eliminate most of these excess edges with the u/v and t/v heuristics (removing more than 99% of all edges before applying Prim's), we find in practice that e is a very small fraction of n^2. In the runtime results (Figure 4), we find that when holding the EST size distribution

constant, the tool as a whole has a runtime of $O(n^2)$ – appearing to be dominated by the time required to apply the filtering heuristics to every EST pair.

B.2 Removing edges

Once the MST has been calculated, our last step is to remove all edges exceeding a threshold weight T, taking the resulting components as our clusters. WCD, when faced with a similar challenge, sets the threshold at $T = 40$, hypothesizing that EST pairs with a d^2 distance of greater than 40 are unlikely to overlap. We have set our threshold to the same $T = 40$ (with discussion below on the effects of changing this).

C Adaptive d^2

In the above discussion of d^2 we assume sequences of at least 100 bp, and in practice use a window of size 100. While the technique should be, in principle, adaptable to short-read sequences (decreasing window size as necessary), there are some problems in the details. Diminishing window sizes implies both diminishing accuracy (requiring the threshold be adjusted) and increasing runtime (as there are potentially more window pairs). More importantly, there is the problem of non-homogeneous sequence lengths: short-read technologies tend to produce short reads, but some technologies can still include long reads in their output. Hence it is not immediately clear how to compare a very short read against a very long one.

We address these issues with our *adaptive* d^2 strategy that works as follows: we partition sequences by length into disjoint groups. For each group we pick a constant window size and threshold tailored to the length range of the group (as well as filtering heuristic constants), and use those values when computing the d^2 distance for two sequences in the same group. We do not compute distances between sequences in non-adjacent groups (i.e. a member of the "shortest groups" against a member of the "longest groups" if there is an intervening sized group). Once computed, we assign to the MST edge the ratio of the d^2 score to the group threshold, thus normalizing the weight for comparisons "between groups" and effectively giving us an edge-removal threshold of 1.

By restricting d^2 comparisons to within a group, we minimize the effects of the heterogeneous sequences lengths by restricting our comparisons to only sequences of similar lengths. By not restricting ourselves to a single window size, we allow ourselves to, when possible, avoid the performance hit of using a small window. As we are not comparing sequences from groups corresponding to radically different sizes we bypass the question of how to compare a long sequence against a short one. And because we allow for the comparison of sequences in neighboring groups, we still get implicit comparisons between long and short sequences in terms of the resulting path of the minimum spanning tree that makes use of these "linking" nodes.

In the PEACE implementation, we subdivide into three groups:

- Less than 150bp ($w = 50$, $T = 50$, $t = 20$, $u = 4$).
- Between 150bp and 400bp ($w = 75$, $T = 75$, $t = 35$, $u = 6$)
- Greater than 400bp ($w = 100$, $T = 100$, $t = 50$, $t = 50$).

D Simulated Test Results

In *Hazelhurst et al.* [3] the authors conduct an investigation of WCD (version 0.5.1) against a number of clustering tools. As they make a convincing argument that WCD is returning better results then the tools against which they compare, we limit our analysis to a comparison of PEACE against WCD and the CAP3 assembly tool [7] – a tool which implicitly clusters while performing assembly. We do so by applying all three tools to a number of data sets, both simulated and real.

D.1 Simulation Tool and Parameters

For simulated data sets we use the **ESTsim** tool to generate simulated EST data sets [8], using the collection of zebra fish genes that served as a basis for the WCD simulations [3]. In generating the ESTs, ESTsim models three types of error (general base read errors, errors due to polymerase decay, and primer interference), and allows those errors to take the form of substitutions, deletions, and insertions (of bases and Ns). See the paper for a discussion of the probability distributions and default parameters. For generating our simulated data, we use the default values for all parameters that are not explicitly being subjected to variation in our experiments – paralleling the testing of the WCD tool.

D.2 Methodology

Each estimate given in the main paper, or in the following, is averaged over 30 trials. Each trial consists of the application of all three tools to a simulated data set, the set having been derived from the application of the ESTsim tool to a set of 100 zebra fish gene sequences [8]. All confidence intervals are calculated at a 95% level of significance.

D.3 Result Quality

Our primary measurement tools if measurement are sensitivity, the Jaccard Index, Type 1 error and Type 2 error (Figure 3). Sensitivity is the ratio for true positives to actual positives ($tp/(tp + fn)$) – the fraction of sequences from the same transcript that are identified as such. The *Jaccard* index combines both sensitivity and specificity, calculated

as the ratio of true positives to the sum of true positives, true negatives, and false positives ($tp/(tp + fn + fp)$) [3].

Type 1 and Type 2 errors, as defined in *Wang et al.* [9], measure quality at the level of the genes from which the ESTs were derived. For Type 1 error, we look at the fraction of genes that were broken into two or more partitions, while in Type 2 error we look at the fraction of clusters that contain two or more genes.

A fifth metric is the number of singletons produce by a tool: sequences that could not be assigned to any cluster. In practice such a sequence could represent complete coverage of a very short transcript (one that may be otherwise had to identify), but is more likely to be an error in the clustering algorithm. In order to identify the first it is important to minimize the second. In Figure S1 we look at the number of singletons produced by each tool when run on simulated data in which there should be no singletons produced, seeing a very slight improvement of PEACE over WCD in certain error ranges.

One of the difficulties in clustering EST data is dealing with highly similar genes. Genes with a high degree of similarity will produce ESTs that reflect that similarity, hence appear to overlap – resulting in the incorrect clustering of ESTs from separate genes. Assembly tools such as CAP3 may have more ability to discriminate between clusters given their more intensive investigation of overlaps, but highly similar sequences are going to cause a problem for any tool.

In Figure S2 we look at the ability of each tool to separate duplicates as a function of the % divergence between the duplications. Unsurprisingly, CAP3 (the assembler) does the best here, able to effectively separate duplicates at 92% similarity or less. PEACE and WCD are roughly comparable, both clearly able to separate duplicates out at a similarity level of 78% – but completely unable to distinguish sharing a similarity of 88% or more.

D.4 Runtime and Memory Footprint

Runtime of the tool is analyzed in Figure 4 of the main paper. In Figure S3 we present a plot of the memory footprint of each tool as a function of input size. Data points were generated by the application of each tool to simulated sets of a specified size (generated by ESTSim), measuring the maximum amount of memory required for the run. We see that while all tools require memory linear in the input size, WCD requires slightly less than does PEACE, while both require significantly less memory than Cap3. In general we find the footprint of PEACE to be relativily small, as compared to other clustering tools (discussed in *Hazelhurst et al.* [3]).

All runs were conducted on 3.0 GHz Intel Xeon EM64T CPUs with 2 MB caches and a 800 MHz front side bus, model number Xeon LV 3.0 (2005)

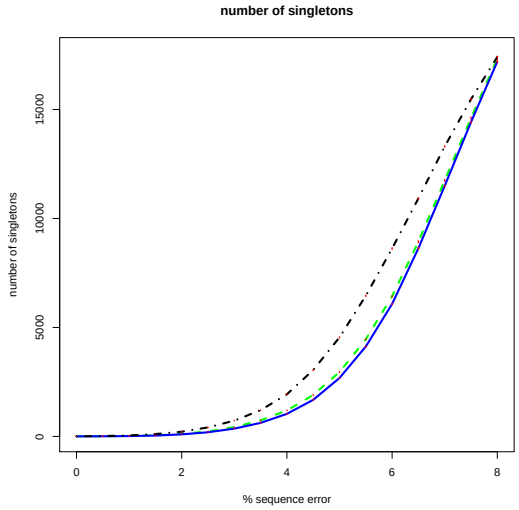

Figure S1: Number of singletons produced by each tool in a simulated run where a fully correct clustering would have no singletons. Values average over 30 trials. Blue/Solid = PEACE, Green/Dashed = WCD, Black/Dot-Dashed = CAP3.

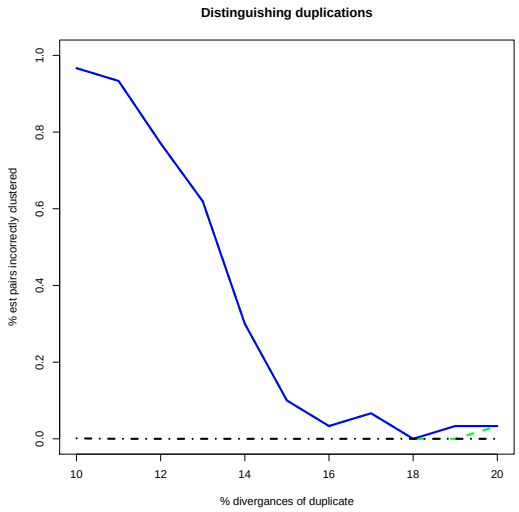

Figure S2: Ability to distinguish duplicates as a function of divergence. Estimates averaged over 30 trials; one trial consists of taking a random gene zebra fish gene, copying it and stochastically changing bases at the specified rate, then using the two genes as the bases for generating a simulated set. Blue/Solid = PEACE, Green/Dashed = WCD (mostly obscured by the blue line), Black/Dot-Dashed = Cap3; variance was too small for visible plotting of confidence intervals.

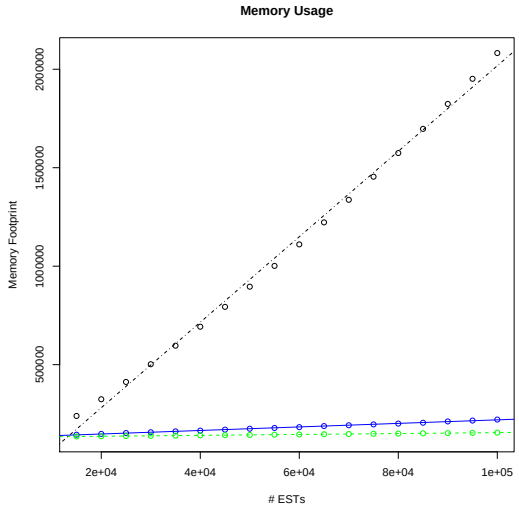

Figure S3: Plots of memory footprint against sequence set size for a single processor run on simulated data for each tool: PEACE (blue), WCD (green) and Cap3 (balck), along with the lesat-squares fit for each. For each fit we have an associated regression coefficient of $r > 0.99$.

D.5 Raising the Threshold

The WCD tool uses a d^2 threshold of 40 to determine whether two sequences are in the same cluster. As this is equivalent to our T threshold for dismissing MST edges, we have set it to the same default value when used for Sanger sequences. However, we did experiment with higher values, with interesting results. An analysis of the preformance of d^2 indicated that a T value as high as 130 should not have a significant impact on its effectiveness as a discriminator. And in fact this was born out in simulation, as shown in Figure S4. In contrast with the equivalent results for a threshold of $T = 40$, as shown in the manuscript, we see considerable improvement in PEACE over WCD in all metrics except Type 2 error, in which WCD slightly better. Unsurprisingly, this also makes a significant difference in the tool's ability to distinguish duplications: we see fairly poor performance in Figure S5.

Most importantly, while the higher threshold appeared to be better in theory and in analysis, the result did not hold in practice. In Table S2 we show the figures corresponding to the manuscript Table 1 when raising the threshold to 130, and observe the poor performance of PEACE when looking at the Jaccard index. It is clear that in the real data, the level of transcript similarity is considerably higher then in the simulations, thus making the reduced specificity of the higher threshold considerably more of a problem.

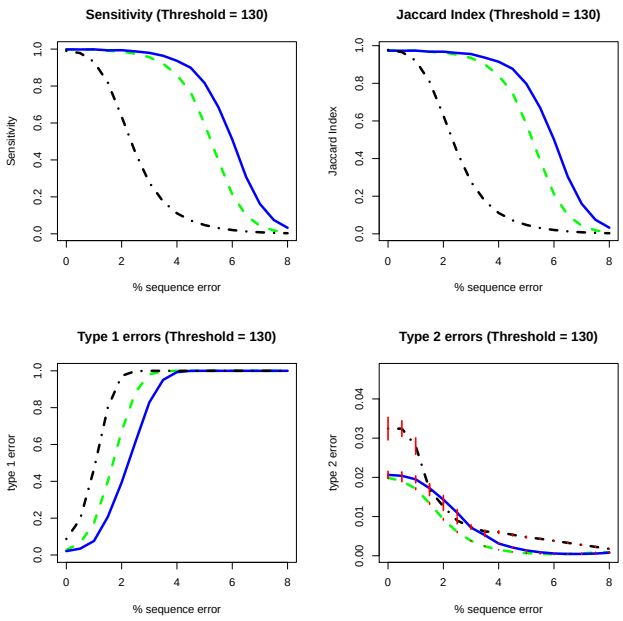

Figure S4: Comparisons of Sensitivity, Jaccard Index, Type 1 error and Type 1 error using the elevated threhold ($T = 130$), providing a contrast against Figure 3 ($T = 40$). Based on the average over 30 simulated Sanger Sequence ESTs sets derived from 100 zebra fish genes. Blue/Solid = PEACE, Green/Dash = WCD, Black/Dot-Dash = Cap3; vertical tics = 95% confidence intervals on estimates. Intervals are not presented for Type 1 error due to the effective lack of variance.

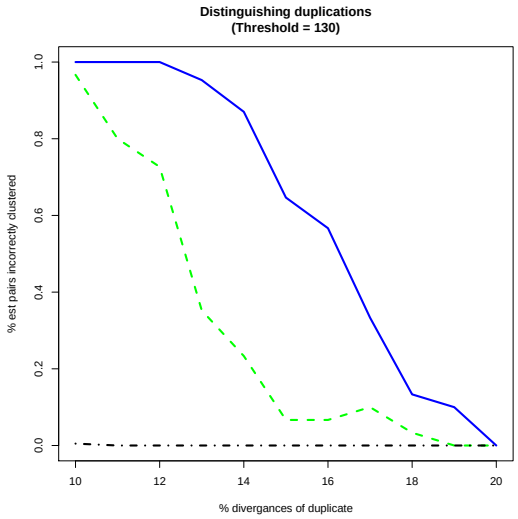

Figure S5: Ability to distinguish duplicates as a function of divergence. Estimates averaged over 30 trials; one trial consists of taking a random zebra fish gene, copying it and stochastically changing bases at the specified rate, then using the two genes as the bases for generating a simulated set. Blue/Solid = PEACE, Green/Dashed = WCD, Black/Dot-Dashed = CAP3; variance was too small for visible plotting of confidence intervals.

E Real Data

E.1 Runtimes

In Table S1 we show the runtime of PEACE and WCD on five different benchmark sets obtained from the WCD and EasyCluster studies [3, 10].

E.2 Higher Threshold

In Table S2 we see the effect on the application of PEACE to the data sets from Table 1 when raising the threshold T to 130 (see previous discussion in Section D.5). While raising T is effective in simulations, it did not hold up in practice. We see essentially no change in the sensitivity, but significant changes in the Jaccard Index (and a corresponding decrease in the total number of clusters). We suspect that this is due to a higher number of more closely related genes than are present in the simulated data. Regardless, without considerably more investigation, these results indicated we should be leaving T at a default rate of 40.

Benchmark	Number of Sequences	Number of Bases	Peace Runtme (s)	WCD Runtime (s)
Human Cluster	17,733	11.7 Mb	293	804
Cotton	29,995	17.1 Mb	222	339
Ricinus	57,690	40.9 Mb	1344	2356
A076941	76,941	32.8 Mb	1166	966
A686903	686,904	294.7 Mb	9449	2481

Table S1: Runtime for PEACE and WCD on benchmarks from the WCD and EasyCluster studies [3, 10]. Runtimes for the first four sets are sequential, while that last was run over thirty processors.

		Sensitivity	Jaccard	Type 1 error	Type 2 error	Number of Clusters	Number of Singletons	Single processor runtime (s)
EasyCluster Human Benchmark (111 Genes)	PEACE	0.999	0.106	0.027	0.030	67	2	295
	WCD	0.998	0.672	0.144	0.044	113	16	804
	Cap3	0.657	0.643	1.000	0.001	2269	1827	NA
WCD A076941 Benchmark (13240 genes)	PEACE	0.937	0.345	0.337	0.0342	18256	8135	1293
	WCD	0.933	0.476	0.350	0.027	18787	8553	966
	Cap3	0.826	0.802	0.486	0.014	25042	14916	NA

Table S2: Comparisons of runs on the EasyCluster human Benchmark Dataset and the WCD A076941 Arabidopsis thaliana dataset using the standard quality measurements with $T = 130$.

F Short Read Analysis

In Table S3 we present the results of application of the tools to MetaSim generated simulated short read data, as discussed in the results section of the paper.

		Sensitivity	Jaccard	Type 1 error	Type 2 error	Number of Clusters	Number of Singletons	Single processor runtime (s)
454 Simulation	PEACE	0.871	0.809	0.186	0.049	129	4	129
	WCD(1)	0.459	0.448	0.946	0.003	599	235	76
	TGICL	0.297	0.288	0.984	0.005	886	0	68
Illumina Simulation	PEACE	0.384	0.343	0.230	0.018	673	299	3463
	TGICL	0.131	0.127	0.977	0.008	1522	0	373

Table S3: Comparisons of runs on on short read data, as generated by MetaSim[11]; each figure is an average over 30 runs. For WCDwe use parameters suggested by the authors ("-H 1 -T 160"). WCD was unable to return any reasonable results for Illumina data, while Cap3 was unable to return results for either sumulation. Each 454 simulation produced 25,000 reads from the 100 zebra-fish transcripts, ranging in size from 200bp to 270bp (aveage = 250bp, standard deviation = 17bp). Each Illumina simulation produce 150K reads from the 100 zebra-fish transcript, each exactly 62bp.

References

[1] Winston Hide, John Burke, and Daniel B Davison. Biological evaluation of d2, an algorithm for high-performance sequence comparison. *Journal of Computational Biology*, 1(3):199–215, Aug 1994. [PubMed:8790465].

[2] Scott Hazelhurst. An efficient implementation of the d2 distance function for est clutering: preliminary investigations. *Proceedings of the SAICSAT*, pages 1–5, Aug 2004.

[3] Scott Hazelhurst, Winston Hide, Zsuzsanna Lipták, Ramon Nogueira, and Richard Starfield. An overview of the wcd est clustering tool. *Bioinformatics*, 24(13):1542–6, Jul 2008. [PubMed:18480101] [PubMed Central:PMC2718666] [doi:10.1093/bioinformatics/btn203].

[4] A Jain, M Murty, and P Flynn. Data clustering: a review. *Computing Surveys (CSUR*, 31(3), Sep 1999.

[5] Xiu-Feng Wan, Mufit Ozden, and Guohui Lin. Ubiquitous reassortments in influenza a viruses. *Journal of bioinformatics and computational biology*, 6(5):981–99, Oct 2008. [PubMed:18942162].

[6] R Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, Jan 1957.

[7] X Huang and A Madan. Cap3: A dna sequence assembly program. *Genome Res*, 9(9):868–877, 1999. [PubMed:10508846] [PubMed Central:PMC310812].

[8] S Hazelhurst and A Bergheim. Estsim: A tool for creating benchmarks for est clustering algorithms. *Dept. of Computer Science, Univ. of Witwatersrand (South Africa), Tech. Rep. CS-2003-1*, 2003.

[9] Ji-Ping Z Wang, Bruce G Lindsay, James Leebens-Mack, Liying Cui, Kerr Wall, Webb C Miller, and Claude W dePamphilis. Est clustering error evaluation and correction. *Bioinformatics*, 20(17):2973–84, Nov 2004. [PubMed:15189818] [doi:10.1093/bioinformatics/bth342].

[10] E Picardi, F Mignone, and G Pesole. Easycluster: a fast and efficient gene-oriented clustering tool for large-scale transcriptome .... *BMC Bioinformatics*, Jan 2009.

[11] Daniel C Richter, Felix Ott, Alexander F Auch, Ramona Schmid, and Daniel H Huson. Metasim: a sequencing simulator for genomics and metagenomics. *PLoS ONE*, 3(10):e3373, Jan 2008. [PubMed:18493042] [PubMed Central:PMC2556396].