



A suffix-array based algorithm for expression clustering

Scott Hazelhurst

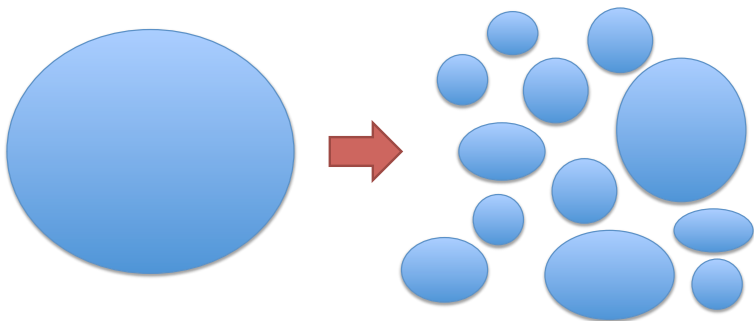
University of the Witwatersrand, Johannesburg (South Africa)

Zsuzsanna Lipták

University of Salerno/University of Verona (Italy)

WAABD, Pisa, 21 Oct. 2011

What is the problem?



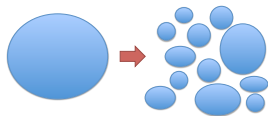
Given:

Set of n sequences (10^5 or more)
of length m

Wanted:

clusters s.t. each cluster
corresponds to a gene

Using pairwise comparison



$n = \# \text{ seq's}$, $m = \text{ave. length}$

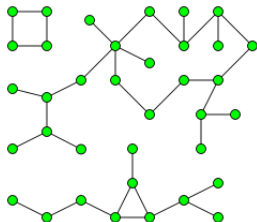
Single linkage

s, t similar $\Rightarrow s, t$ in same cluster

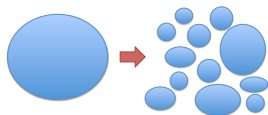
As a graph problem

Find connected components.

Remember: Edges have to be computed!



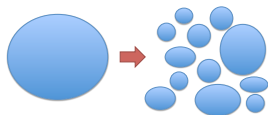
Using pairwise comparison



$n = \# \text{ seq's}$, $m = \text{ave. length}$

- One comparison (e.g. edit dist.) costs m^2 \rightsquigarrow $O(n^2 m^2)$ time

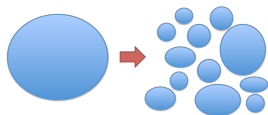
Using pairwise comparison



$n = \#$ seq's, $m =$ ave. length

- One comparison (e.g. edit dist.) costs $m^2 \rightsquigarrow O(n^2 m^2)$ time
Most pairs are not similar: use filters e.g. $O(m)$ time
if x, y pass filter then compute $d(x, y)$

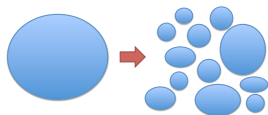
Using pairwise comparison



$n = \#$ seq's, $m =$ ave. length

- One comparison (e.g. edit dist.) costs $m^2 \rightsquigarrow O(n^2 m^2)$ time
Most pairs are not similar: use filters e.g. $O(m)$ time
if x, y pass filter then compute $d(x, y)$
- If one filter test costs $m \rightsquigarrow O(n^2 m)$ time + sg.

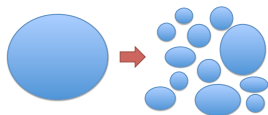
Using pairwise comparison



$n = \#$ seq's, $m =$ ave. length

- One comparison (e.g. edit dist.) costs $m^2 \rightsquigarrow O(n^2 m^2)$ time
Most pairs are not similar: use filters e.g. $O(m)$ time
if x, y pass filter then compute $d(x, y)$
- If one filter test costs $m \rightsquigarrow O(n^2 m)$ time + sg.
- In general:
If one filter test costs $f(m) \rightsquigarrow O(n^2 f(m))$ time + sg.

Using pairwise comparison



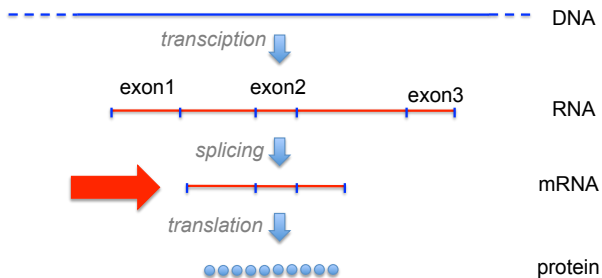
$n = \#$ seq's, $m =$ ave. length

- One comparison (e.g. edit dist.) costs $m^2 \rightsquigarrow O(n^2 m^2)$ time
Most pairs are not similar: use filters e.g. $O(m)$ time
if x, y pass filter then compute $d(x, y)$
- If one filter test costs $m \rightsquigarrow O(n^2 m)$ time + sg.
- In general:
If one filter test costs $f(m) \rightsquigarrow O(n^2 f(m))$ time + sg.

Here: We get rid of the n^2 factor!

Clustering expression data

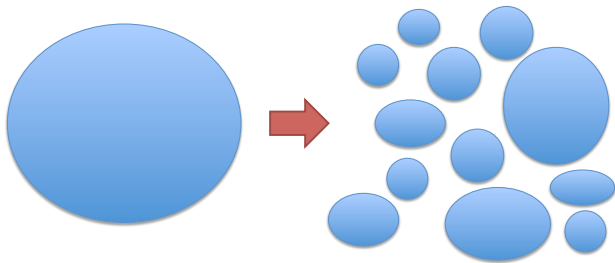
Expression data (formerly ESTs)



Transcriptome/Expression data

(ESTs = expressed sequence tags)
Partial cDNA copies of mRNAs (lab process)
both Sanger-style (ESTs) and NGS

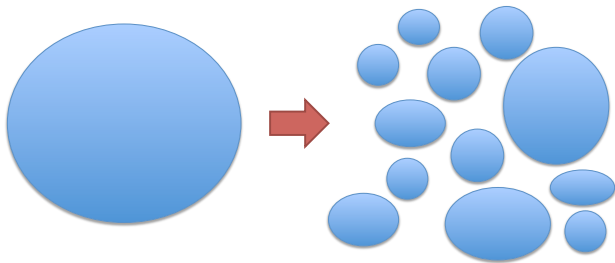
Problem statement



Desideratum

Given a set of sequences S , find a partition of S s.t. if s, t are products of the same gene, then s, t in same cluster.

Problem statement



Desideratum

Given a set of sequences S , find a partition of S s.t. if s, t are products of the same gene, then s, t in same cluster.

Formal Problem

Given a set of sequences S , find a partition of S s.t. if s, t are **similar**, then s, t in same cluster.

Expression clustering

- expression studies (e.g. diseased vs. healthy),
- gene discovery,
- SNP detection,
- discovery of products of **alternative splicing**,
- estimating no. of genes

We are not doing:

- consensus building
- alignment to genome

String similarity

Example

$s = \text{CAAGACAA}$, $t = \text{CAGAGCAC}$

- alignment / edit distance (N/W 1970, S/W 1981)
- q -gram-distance: $\sum_{|w|=q} |\text{freq}(w,s) - \text{freq}(w,t)|$ (Ukkonen 1992)
- d^2 : $\sum_{|w|=q} (\text{freq}(w,s) - \text{freq}(w,t))^2$ (Torney et al. 1990)
- others (fingerprints, information theory based, ...)

String similarity

Example

$s = \text{CAAGACAA}$, $t = \text{CAGAGCAC}$

- alignment / edit distance = 3 (N/W 1970, S/W 1981)
- q -gram-distance: $\sum_{|w|=q} |\text{freq}(w,s) - \text{freq}(w,t)|$ (Ukkonen 1992)
- d^2 : $\sum_{|w|=q} (\text{freq}(w,s) - \text{freq}(w,t))^2$ (Torney et al. 1990)
- others (fingerprints, information theory based, ...)

CAAGA-CAA
CA-GAGCAC

String similarity

Example

$s = \text{CAAGACAA}$, $t = \text{CAGAGCAC}$

- alignment / edit distance = 3 (N/W 1970, S/W 1981)
- q -gram-distance: $\sum_{|w|=q} |\text{freq}(w,s) - \text{freq}(w,t)|$ (Ukkonen 1992)
- d^2 : $\sum_{|w|=q} (\text{freq}(w,s) - \text{freq}(w,t))^2$ (Torney et al. 1990)
- others (fingerprints, information theory based, ...)

CAAGA-CAA
CA-GAGCAC

AA	2 0
AC	1 1
AG	1 2
CA	2 2
GA	1 1
GC	0 1

String similarity

Example

$s = \text{CAAGACAA}$, $t = \text{CAGAGCAC}$

- alignment / edit distance = 3 (N/W 1970, S/W 1981)
- q -gram-distance: $\sum_{|w|=q} |\text{freq}(w,s) - \text{freq}(w,t)| = 4$ (Ukkonen 1992)
- d^2 : $\sum_{|w|=q} (\text{freq}(w,s) - \text{freq}(w,t))^2$ (Torney et al. 1990)
- others (fingerprints, information theory based, ...)

CAAGA-CAA
CA-GAGCAC

AA	2	0	2
AC	1	1	0
AG	1	2	1
CA	2	2	0
GA	1	1	0
GC	0	1	1

4

String similarity

Example

$s = \text{CAAGACAA}$, $t = \text{CAGAGCAC}$

- alignment / edit distance = 3 (N/W 1970, S/W 1981)
- q -gram-distance: $\sum_{|w|=q} |\text{freq}(w,s) - \text{freq}(w,t)| = 4$ (Ukkonen 1992)
- d^2 : $\sum_{|w|=q} (\text{freq}(w,s) - \text{freq}(w,t))^2 = 6$ (Torney et al. 1990)
- others (fingerprints, information theory based, ...)

CAAGA-CAA
CA-GAGCAC

AA	2	0	2	4
AC	1	1	0	0
AG	1	2	1	1
CA	2	2	0	0
GA	1	1	0	0
GC	0	1	1	1

4 6

Similar windows

>1

```
tttattattatgggattaggcttctctctcgcgattctcttacgtt  
ggttatatctttatattactttttcttattgtataattagcttccg  
tcctctaggtcccttctcta
```

>2

```
cccctctctgttggttatatctttatattactttttcttattgtataa  
cttcttattattctccgctctctctcttattcttccggaattctcgagg  
ggatataatattatgcgggcgccctcttataatctctatattctctc  
a
```

Similar windows

>1

```
tttattattatgggattaggcttctctctcgcgattctcttacGTT  
CGTTATATCTTTATATTACTTTTTCTTATTGTATAAAttagcttccg  
tcctctaggtcccttctcta
```

>2

```
cccctctctGTTGGTTATATCTTTATATTACTTCTTCTTATTGTATAA  
cttcttattattctccgctctctctcttattcttccggaattctcgagg  
ggatataatattatgcgggcgcctcttataatctctatattctctc  
a
```

Our string similarity measure

We use d^2 , minimized over all pairs of windows of given size.

$$d(s, t) = \min_{s' \sqsubseteq s, t' \sqsubseteq t, |s|=|t|=\mu} \sum_{|w|=q} (\text{freq}(w, s') - \text{freq}(w, t'))^2$$

Parameters:

wordsize q ($= 6, 8$), window size, threshold (s, t similar iff $d(s, t) < \theta$)

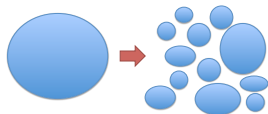
Computation time: $O(m^2)$

Speedup of previous tool

wcd + KABOOM! = wcd-express

The KABOOM-filter

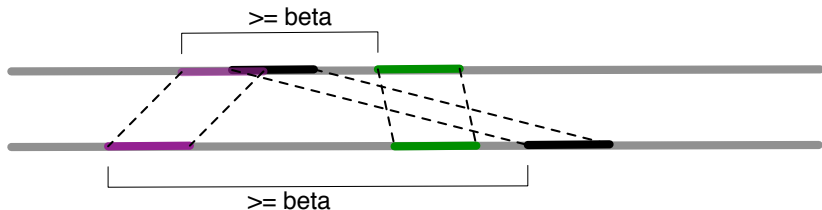
Filters for d^2



$n = \#$ seq's, $m =$ ave. length

- Recall: we want to avoid $O(n^2m^2)$ runtime
- Filters with guarantee that need pairwise comparison only reduce m^2 factor
- **One long exact match** can be implemented with suffix array (linear in nm) – but too fragile!
- KABOOM-filter combines good running time with good filtering qualities

The KABOOM-filter

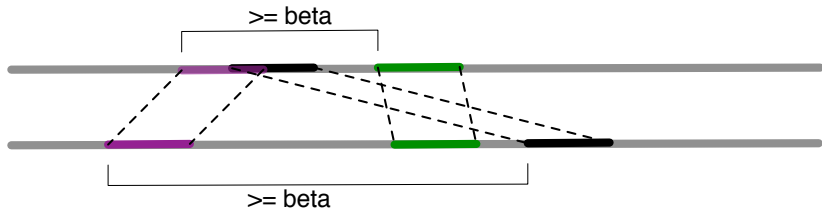


Definition

s, t are (k, α, β) -similar if they have at least α many k -words in common, at least β apart.

- k -words are counted with multiplicities (a.k.a. D^2)
- distance β between leftmost and rightmost occurrence, in both strings
- order of matches not important

The KABOOM-filter



Definition

s, t are (k, α, β) -similar if they have at least α many k -words in common, at least β apart.

- heuristic filter for d^2
- very good in practice: produces practically no FNs and reduces no. of d^2 comparisons dramatically
- $k > q$ (typically k between 12 and 20)

Implementation of the KABOOM-filter

for every sequence s_i
 for every k -word w in s_i
 find all s_j containing w
 $M = M \cup s_j$ (M : candidate sequences for s_i)
 update variables for s_j
 for every s_j in M
 check whether s_i, s_j fulfill (k, α, β) -criterion
 if yes, compute $d(s_i, s_j)$

Implementation of the KABOOM-filter

for every sequence s_i

for every k -word w in s_i

find all s_j containing w

$M = M \cup s_j$

(M : candidate sequences for s_i)

update variables for s_j

for every s_j in M

check whether s_i, s_j fulfill (k, α, β) -criterion

if yes, compute $d(s_i, s_j)$

We need

- suffix array of concatenated string
- inverse of suffix array

Implementation of the KABOOM-filter

```
TTTTAGCTTTTGGGGCCITGGGTTAGGATCTCCCTATATATATATATATCCCT  
AGGGCAGTGGCTGTGGGACTCGGGTCCCTGCCCTGCCATCCTCATCCTGGCAA  
CCCCACTCCCTGGGATGCTACAATCCAATGATGGAAAGATGGCATTAGCTACA  
CGCTTTCCCGAGATGAACATACCACGAACCTNGAGATGCAGAGTCTCCAGAGA  
CCTCCCCGGGAC
```

```
>T27882 g609980 | T27882 CLONE_LIB: Human Eye.  
TGCCTTCAAACGGCTCTGGAATATTTCCAAAGGGTATTGGTGAAGAATAT  
ATAGCTCCAGAGGACCAATCAAATGGGTGAAGAACCTCTGTAAGATGCAC  
GGTGGTGCATTTGGAACATCTCACAGCTCTGGGACTTCTTCCGAAGACCA  
TACCCAGAATCTGCACACTCAAGACATACCTG
```

```
>T27883 g609981 | T27883 CLONE_LIB: Human Eye.  
CTGGGCAAACTAAGCCTTGGCCAGGGCCGAAGTTAGGCCCTGTNTTGT  
CGAGGGCCAGCGGGGGATCTCTACACTCTGCTGTGGGCCAGGGGCTGCTG  
TGCTGCAAAGTCTGGGATCTGCCACTCAACCCGGGAGTGGTGTCCCATG  
AGAGCTTTGACAGATTGTGATTCACAGCTGGGCCCTATGTTTGTGTTCTG  
....  
....
```

```
>T27884 g609982 | T27884 CLONE_LIB: Human Eye.  
CCACGACAGACAGGACTCTCATTAAAGGAAGGTGCTGTGCCCTGACCCCTAC  
AAGAGAAGATGCTCACTTCACTATGTTACCCCAAGAAGGGGACAGGCCAC  
GGG
```

```
>T27885 g609983 | T27885 CLONE_LIB: Human Eye.  
CCCAAACCGATTAAAGTGCATGACGGAAACAATAGGACTTCCCCAGGGGNTGAA  
TCCCGGCCAGCCAGGTGACCCCAAGGTCTGGATGTTNCTGGTCTGTTCTTC
```

```
>T27886 g609984 | T27886 CLONE_LIB: Human Eye.  
ATCCGCCCATCATGGGTTCGATGCATGCTCCCGGAAGGGCTGTGCCAGT  
AATCTNANTAGAGAGCCGGATTCACCCGTTTTGGCTTCGATATATAAGACC  
TCCTCCCTCCCAATGGGAAATATGGAATCATCTTACAGCCCTGCCCCGTG  
TTGTCTGTGTACTCAAGCAATAAAATGGTTGTTAACT
```

```
>T27887 g609985 | T27887 CLONE_LIB: Human Eye.  
GGCACAGTCACTTCCCTGGGGCCGGTGTCACTGTGGGCTCCTGGGGCGGG  
CTGGNATTTACCGCATGCTGCCACCCAGATCCANCTGINTCCACTTTCAC  
TCTGGTGTGATTAATGAGGGCAGGGGT
```

```
>T27888 g609986 | T27888 CLONE_LIB: Human Eye.  
ACACCTGCCATGTGCAGCATGAGGGTTTGCCCAAGCCCTCACCCCTGAGATG
```

```
>T27890 g609988 | T27890 CLONE_LIB: Human Eye.  
CCGGCAGNTTGCAGCAAAAANTGGNGGTGATGCAGGGACATCACTGANTTCA  
TGGTTATGGGGGACAAAAAGACCTTTAGAAGATGGAGATCAACAGATGCT  
TGCTCTCAAAGTACTCTTTTGGAAACAGTATACCACCGATGCATCAGCAG  
TGTAATG
```

2009	AGCTTTATTTTCC...CCAA	661
2010	AGCTTTATTTTCC...CCCA	555
2011	AGCTTTATTTTCC...CCCC	7
2012	AGCTTTATTTTCC...CCCC	1
2013	AGCTTTATTTTCC...CCCC	130
2014	AGCTTTATTTTCC...CCCC	18
2015	AGCTTTATTTTCC...CCCC	400
2016	AGCTTTATTTTCC...CCCT	

Modified suffix array sa_k

aaa@aacggt@gttaaagt@tcggt@gttat@cgg@acggt@

i	sa	Text from $sa[i]$	sa_k
15	16	agt@tcggt@gttat@cgg@acggt@	16
16	29	at@cgg@acggt@	29
17	32	cgg@acggt@	6
18	37	cgg@	21
19	6	cgg@gttaaagt@t...gt@	32
20	21	cgg@gttat@cgg@acggt@	37
21	34	g@acggt@	34
22	33	gg@acggt@	33
23	38	ggt@	7
24	7	ggt@gttaaagt@t...t@	22
25	22	ggt@gttat@cgg@acggt@	38
26	39	gt@	8

Modified suffix array sa_k

aaa@aacggt@gttaaagt@tcggt@gttat@cgg@acggt@

i	sa	Text from $sa[i]$	sa_k
15	16	agt@tcggt@gttat@cgg@acggt@	16
16	29	at@cgg@acggt@	29
17	32	cgg@acggt@	6
18	37	cgg@	21
19	6	cgg@gttaaagt@t...gt@	32
20	21	cgg@gttat@cgg@acggt@	37
21	34	g@acggt@	34
22	33	gg@acggt@	33
23	38	ggt@	7
24	7	ggt@gttaaagt@t...t@	22
25	22	ggt@gttat@cgg@acggt@	38
26	39	gt@	8

Analysis

Time

- $\sum_{|w|=k} \text{freq}(w)^2$, which is between nm and n^2m^2
- in practice: very fast
- WE work efficiency: $1 - (\text{TP} + \text{FP}) / (\text{all pairs})$ above 99.5%

Space

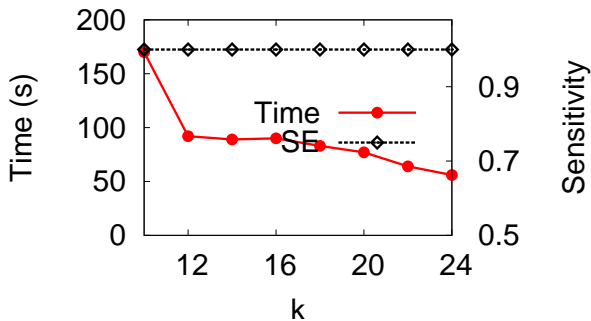
- text nm (nt)
- k -depth suffix array: $nm \log n$ (x2 for RC)
- inverse: $nm \log n$

Experimental results

Speedup w.r.t. wcd

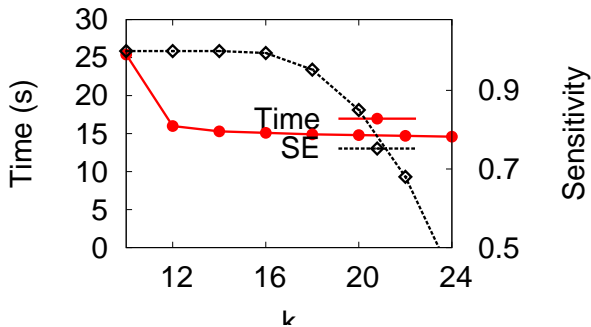
Data set	# seqs (K)	size Mb	γE γE	wcd-exp (s)	wcd (s)	Speed-up factor
A076941	77	32	17	100	578	5.7
A208	484	208	102	1240	23983	19.3
C10	126	56	355	511	4512	8.8
chlamy	190	100	139	1000	5989	6.0
Drosophila	25	86	68	184	1542	8.4
ecHuman	17	11	171	135	496	3.7
pubcot	30	17	34	65	222	3.4
ricinus	58	40	162	840	1518	1.8
xen	233	137	63	855	9298	10.8

Sensitivity to choice of k



Filtered cotton set, $\alpha = 3, \beta = 32, H = 48$.

Sensitivity to choice of k



Synthetic 454 data set, $\alpha = 3, \beta = 16, H = 32$.

Comparison with other tools

Issues

- choice of parameters
- pre-/postprocessing (here: filtering)
- quality: what's the truth?
- synthetic vs. real data

Comparison with other tools

Issues

- choice of parameters
- pre-/postprocessing (here: filtering)
- quality: what's the truth?
- synthetic vs. real data

Citroën/Ferrari comparison problem



FASTER THAN A FERRARI.

Travelling flat out at 71.5mph the Citroën 2CV will easily overtake the Ferrari Mondial travelling at 67mph.



AS MANY WHEELS AS A ROLLS ROYCE.

The £13,410 Rolls-Royce Silver Spirit. How many wheels? Four. The £2,384 Citroën 2CV. How many wheels? Exactly the same.



MORE ROOM THAN A PORSCHE.

With a possible 30cu. ft. boot space there's no need for one of those plastic luggage racks on our little run-about.



THE £2,674 CITROËN 2CV.

All you'll ever need in a car.

PRICES CORRECT AT TIME OF GOING TO PRESS. EXCLUDES TAXES AND LICENSING FEES. ALL PRICES IN POUNDS. SEE THE COMPARTMENT FOR MORE DETAILS OF FINANCING AND LEASING. CONTACT YOUR CITROËN DEALER FOR MORE INFORMATION.

PUBLISHED September 18 1984

Quality and runtime comparison

	Sensitivity		Jaccard Index		Time (s)	
	wcdx	PEACE	wcdx	PEACE	wcdx	PEACE
A076941	0.932	0.930	0.473	0.477	100	951
chlamy	0.949	0.949	0.513	0.513	907	8823
ecHuman	0.996	0.998	0.707	0.630	50	147
metasim454	0.793	0.714	0.765	0.689	16	66
metasimIllum	0.444	0.368	0.398	0.364	19	1975

$$SE = TP/(TP+FN), JI = TP/(TP+FP+FN), PPV = TP/(TP+FP)$$

PEACE: *NAR*, 2010

Quality of clustering using KABOOM-heuristic

“We can drive our car at least as fast as others have driven their car.”

- ESTate
- PaCE
- PEACE
- TIGR

Quality of clustering using KABOOM-heuristic

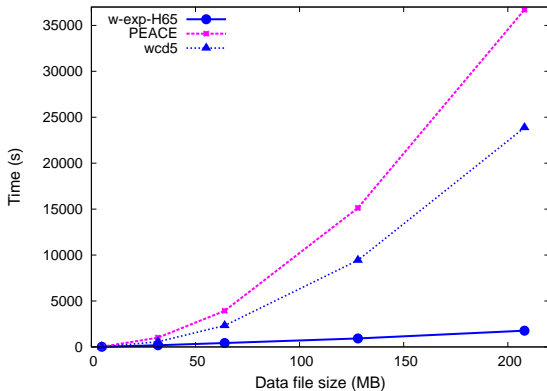
“We can drive our car at least as fast as others have driven their car.”

- ESTate
- PaCE
- PEACE
- TIGR

Performance matters.

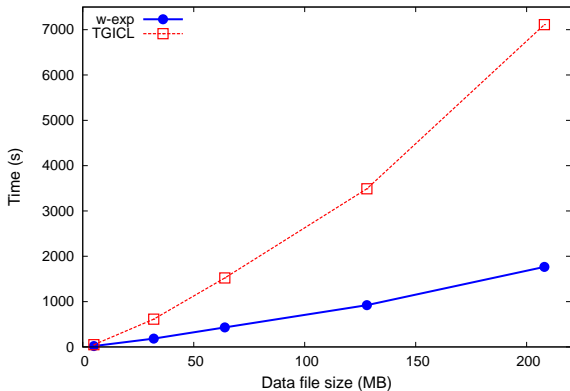
Runtime comparison

Arabidopsis – wcd-express versus PEACE



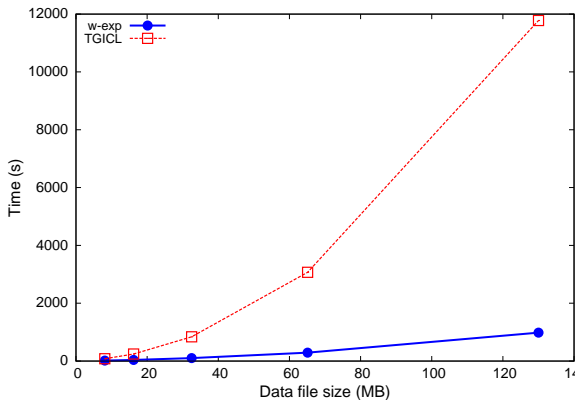
Runtime comparison

Arabidopsis – wcd-express versus TGICL



Runtime comparison

Human 454 ESTs – wcd-express versus TIGR



Conclusion

- $wcd + KABOOM! = wcd\text{-}express$
- Much faster and at least as good as other tools.
- Memory: very heavy memory usage (suffix array + inverse)

Conclusion

- $wcd + KABOOM! = wcd\text{-}express$
- Much faster and at least as good as other tools.
- Memory: very heavy memory usage (suffix array + inverse)

Ongoing/future work

- Memory needs to be reduced!
- Integrate $wcd\text{-}express$ into workflow (filtering/masking, assembly)
- GUI?

Paper to appear in Bioinformatics.



it starts with a
playground.

Thank you.

`zsuzsa@cebitec.uni-bielefeld.de`

`Scott.Hazelhurst@wits.ac.za`