Implementation of a Near-Optimal Complex Root Clustering Algorithm

ICMS

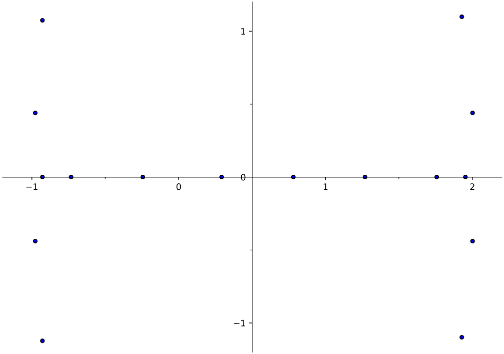
Rémi Imbach[1,3], Victor Y. Pan[2,4] and Chee Yap[1,5]

[1] Courant Institute of Mathematical Sciences, New York University, USA
[2] Lehman College, City University of New York, USA
[3] European Union's H2020 No. 676541 (OpenDreamKit)
[4] NSF Grants # CCF-1116736 and # CCF-1563942 and PSC CUNY Award 698130048.
[5] NSF Grants # CCF-1423228 and # CCF-1564132

Root isolation problem

Input: a polynomial $f \in \mathbb{C}[z]$, $\epsilon > 0$,

Output:

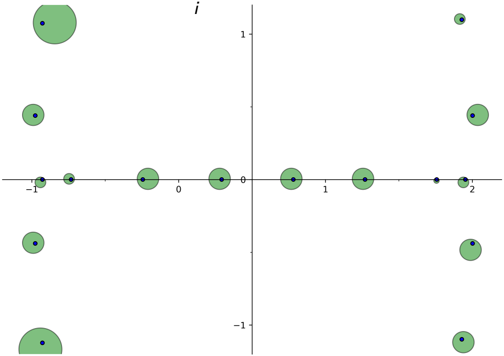


Root isolation problem

Input: a polynomial $f \in \mathbb{C}[z]$, $\epsilon > 0$,

Output: a set $\{\Delta_1, \ldots, \Delta_k\}$ of pairwise-disjoint discs such that:

- the Δ_i's have radius $r(\Delta_i) \leq \epsilon$ and contain a unique root
- Global version: $Z(\mathbb{C}, f) \subseteq \bigcup_i \Delta_i$



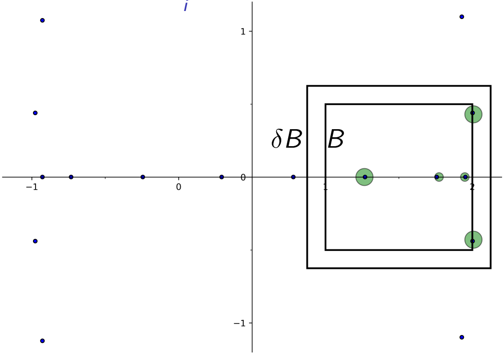
Notations: $Z(S, f)$: roots of f in S

Root isolation problem

Input: a polynomial $f \in \mathbb{C}[z]$, $\epsilon > 0$, a complex box B

Output: a set $\{\Delta_1, \ldots, \Delta_k\}$ of pairwise-disjoint discs such that:

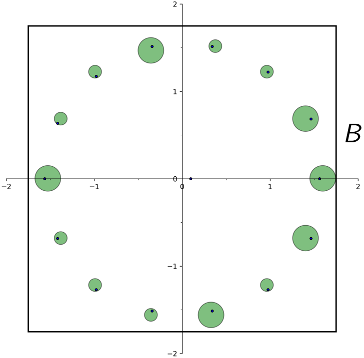
- the Δ_i's have radius $r(\Delta_i) \leq \epsilon$ and contain a unique root
- Local version: $Z(B, f) \subseteq \bigcup_i \Delta_i \subseteq Z(\delta B, f)$, for $\delta > 1$

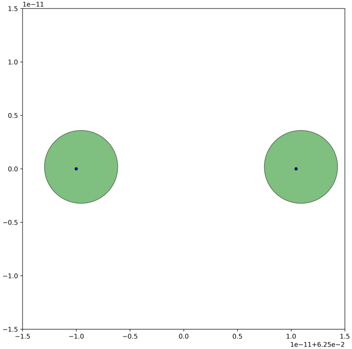


Notations: $Z(S, f)$: roots of f in S

Root isolation problem

Example: Mignotte-like polynomial: $z^d - 2(2^\sigma z - 1)^2$, where $d = 16, \sigma = 4$

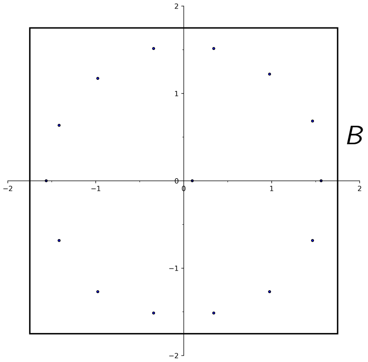




Local root clustering problem

Input: a polynomial $f \in \mathbb{C}[z]$, $\epsilon > 0$, a complex box B

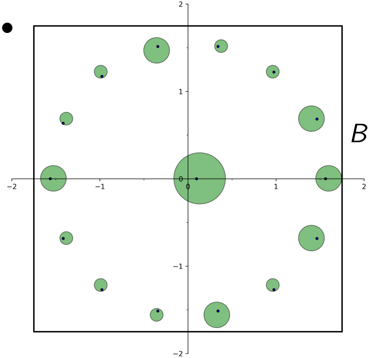
Output:



Local root clustering problem

Input: a polynomial $f \in \mathbb{C}[z]$, $\epsilon > 0$, a complex box B

Output: a set of pairs $\{(\Delta_1, m_1), \ldots, (\Delta_k, m_k)\}$ where

- the Δ_i's are pairwise-disjoint discs of radius $r(\Delta_i) \leq \epsilon$
- $\forall i$, $\#(\Delta_i, f) = m_i$,
- 

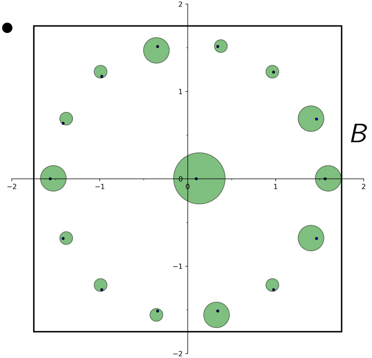
$Z(B, f) \subseteq \bigcup_i \Delta_i \subseteq Z(\delta B, f)$, for $\delta > 1$

Notations: $\#(S, f)$: sum of multiplicities of roots of f in S

Local root clustering problem

Input: a polynomial $f \in \mathbb{C}[z]$, $\epsilon > 0$, a complex box B

Output: a set of pairs $\{(\Delta_1, m_1), \ldots, (\Delta_k, m_k)\}$ where

- the Δ_i's are pairwise-disjoint discs of radius $r(\Delta_i) \leq \epsilon$
- $\forall i$, $\#(\Delta_i, f) = m_i$, and $\#(3\Delta_i, f) = m_i$ (natural clusters)
- 

$Z(B, f) \subseteq \bigcup_i \Delta_i \subseteq Z(\delta B, f)$, for $\delta > 1$

Notations: $\#(S, f)$: sum of multiplicities of roots of f in S

Local root clustering algorithm

[BSS+16] Ruben Becker, Michael Sagraloff, Vikram Sharma, Juan Xu, and Chee Yap.
Complexity analysis of root clustering for a complex polynomial.
In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, pages 71–78. ACM, 2016.

Input polynomial: f given *via* a black-box $[f]$
$[f] : L \mapsto \tilde{f}$ L-bit approx. of (the coeffs. of) f

Near optimal: bit complexity $\widetilde{O}(d^2(\sigma + d))$
for the benchmark problem

Notations: d, σ: degree, bit-size of f

Outline of [BSS+16]

Discarding test: $T_0 : (\Delta, [f]) \mapsto m \in \{-1, 0\}$
$T_0(\Delta, [f]) = 0 \Rightarrow f$ has no root in Δ

Counting test: $T_* : (\Delta, [f]) \mapsto m \in \{-1, 0, \dots, d\}$
$T_*(\Delta, [f]) \geq 0 \Rightarrow \#(\Delta, f) = m$

Subdivision approach:

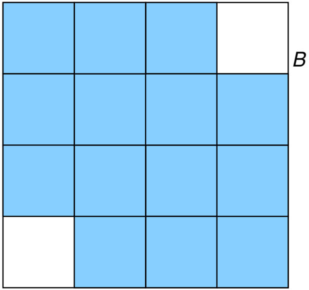
Notations: $\#(S, f)$: sum of multiplicities of roots of f in S
d: degree of f

Outline of [BSS+16]

Discarding test: $T_0 : (\Delta, [f]) \mapsto m \in \{-1, 0\}$
$T_0(\Delta, [f]) = 0 \Rightarrow f$ has no root in Δ

Counting test: $T_* : (\Delta, [f]) \mapsto m \in \{-1, 0, \ldots, d\}$
$T_*(\Delta, [f]) \geq 0 \Rightarrow \#(\Delta, f) = m$

Subdivision approach:



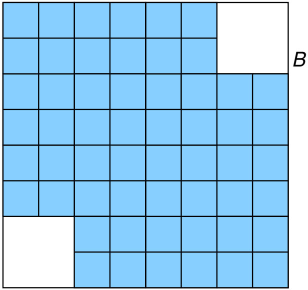
Notations: $\#(S, f)$: sum of multiplicities of roots of f in S
d: degree of f

Outline of [BSS+16]

Discarding test: $T_0 : (\Delta, [f]) \mapsto m \in \{-1, 0\}$
$T_0(\Delta, [f]) = 0 \Rightarrow f$ has no root in Δ

Counting test: $T_* : (\Delta, [f]) \mapsto m \in \{-1, 0, \ldots, d\}$
$T_*(\Delta, [f]) \geq 0 \Rightarrow \#(\Delta, f) = m$

Subdivision approach:



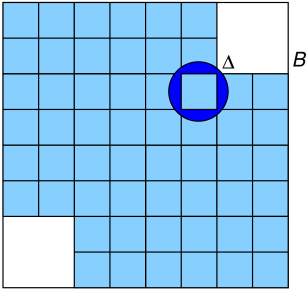
Notations: $\#(S, f)$: sum of multiplicities of roots of f in S
d: degree of f

Outline of [BSS+16]

Discarding test: $T_0 : (\Delta, [f]) \mapsto m \in \{-1, 0\}$
$T_0(\Delta, [f]) = 0 \Rightarrow f$ has no root in Δ

Counting test: $T_* : (\Delta, [f]) \mapsto m \in \{-1, 0, \ldots, d\}$
$T_*(\Delta, [f]) \geq 0 \Rightarrow \#(\Delta, f) = m$

Subdivision approach:



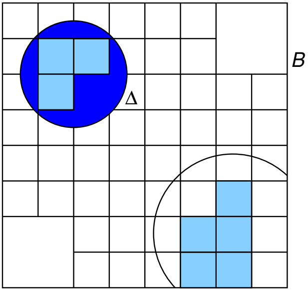
Notations: $\#(S, f)$: sum of multiplicities of roots of f in S
d: degree of f

Outline of [BSS+16]

Discarding test: $T_0 : (\Delta, [f]) \mapsto m \in \{-1, 0\}$
$T_0(\Delta, [f]) = 0 \Rightarrow f$ has no root in Δ

Counting test: $T_* : (\Delta, [f]) \mapsto m \in \{-1, 0, \ldots, d\}$
$T_*(\Delta, [f]) \geq 0 \Rightarrow \#(\Delta, f) = m$

Subdivision approach:



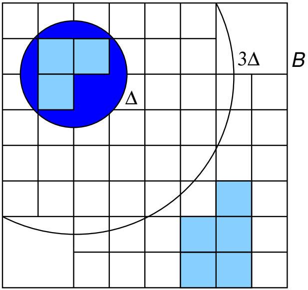
Notations: $\#(S, f)$: sum of multiplicities of roots of f in S
d: degree of f

Outline of [BSS+16]

Discarding test: $T_0 : (\Delta, [f]) \mapsto m \in \{-1, 0\}$
$T_0(\Delta, [f]) = 0 \Rightarrow f$ has no root in Δ

Counting test: $T_* : (\Delta, [f]) \mapsto m \in \{-1, 0, \ldots, d\}$
$T_*(\Delta, [f]) \geq 0 \Rightarrow \#(\Delta, f) = m$

Subdivision approach:



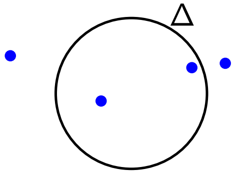
Notations: $\#(S, f)$: sum of multiplicities of roots of f in S
d: degree of f

The Pellet test

Pellet's Theorem: Δ complex disc centered in c and radius r
$f \in \mathbb{C}[z]$, $f_\Delta = f(c + rz)$
If $\exists 0 \leq m \leq d$ s.t.

$$|(f_\Delta)_m| > \sum_{i \neq k} |(f_\Delta)_i| \quad (1)$$

then f has exactly m roots in Δ.



Notations: $(g)_m$: coeff. of the monomial of degree m of g

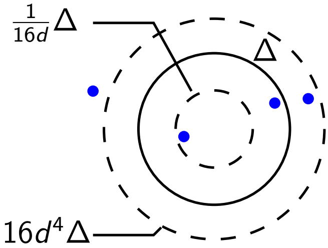
The Pellet test

Pellet's Theorem: Δ complex disc centered in c and radius r
$f \in \mathbb{C}[z]$, $f_\Delta = f(c + rz)$
If $\exists 0 \leq m \leq d$ s.t.

$$|(f_\Delta)_m| > \sum_{i \neq k} |(f_\Delta)_i| \quad (1)$$

then f has exactly m roots in Δ.

If f has no root in this annulus $\rightarrow$
$\exists m$ s.t. eq. 1 holds.



Notations: $(g)_m$: coeff. of the monomial of degree m of g

The Pellet test with Graeffe iterations

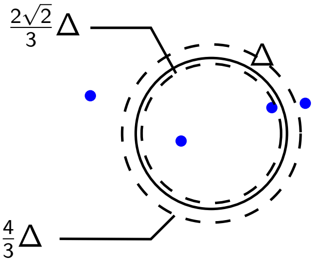
Let $N = 4 + \lceil log(1 + log(d)) \rceil$

Pellet's Theorem: Δ complex disc centered in c and radius r
$f \in \mathbb{C}[z]$, $f_\Delta = f(c + rz)$, $f_\Delta^{[N]}$: N-th Graeffe iterate of f_Δ
If $\exists 0 \leq m \leq d$ s.t.

$$|(f_\Delta^{[N]})_m| > \sum_{i \neq k} |(f_\Delta^{[N]})_i| \quad (1)$$

then f has exactly m roots in Δ.

If f has no root in this annulus $\rightarrow$
$\exists m$ s.t. eq. 1 holds.



Notations: $(g)_m$: coeff. of the monomial of degree m of g

The Pellet test with Graeffe iterations

Let $N = 4 + \lceil log(1 + log(d)) \rceil$

Pellet's Theorem: Δ complex disc centered in c and radius r
$f \in \mathbb{C}[z]$, $f_\Delta = f(c + rz)$, $f_\Delta^{[N]}$: N-th Graeffe iterate of f_Δ
If $\exists 0 \leq m \leq d$ s.t.

$$|(f_\Delta^{[N]})_m| > \sum_{i \neq k} |(f_\Delta^{[N]})_i| \qquad (1)$$

then f has exactly m roots in Δ.

GraeffePelletTest(Δ, k, f) *//Output in* $\{-1, 0, 1, \dots, k\}$

1. compute $f_\Delta^{[N]}$
2. **for** m **from** 0 **to** k **do**
3. **if** $|(f_\Delta^{[N]})_m| > \sum_{i \neq k} |(f_\Delta^{[N]})_i|$
4. **return** m
5. **return** -1

Notations: $(g)_m$: coeff. of the monomial of degree m of g

The soft Pellet test

$\tilde{T}_k^G(\Delta, k, [f])$ *//Output in* $\{-1, 0, 1, \ldots, k\}$

... *//soft version of GraeffePelletTest*(Δ, k, f)

GraeffePelletTest(Δ, k, f) *//Output in* $\{-1, 0, 1, \ldots, k\}$

1. compute $f_\Delta^{[N]}$
2. **for** m **from** 0 **to** k **do**
3. **if** $|(f_\Delta^{[N]})_m| > \sum_{i \neq k} |(f_\Delta^{[N]})_i|$
4. **return** m
5. **return** -1

The soft Pellet test

$\tilde{T}^G_k(\Delta, k, [f])$	//*Output in* $\{-1, 0, 1, \ldots, k\}$
... //*soft version of GraeffePelletTest*(Δ, k, f)	

Discarding test:

$T_0(\Delta, [f])$	//*Output in* $\{-1, 0\}$
1. return $\tilde{T}^G_k(\Delta, 0, [f])$	

Counting test:

$T_*(\Delta, [f])$	//*Output in* $\{-1, 0, 1, \ldots, d\}$
1. return $\tilde{T}^G_k(\Delta, d, [f])$	

Our implementation

Ccluster: library in C based on

- FLINT[1]: arithmetic for the geometric algorithm
- Arb[2]: arbitrary precision floating arithmetic with error bounds

Available at https://github.com/rimbach/Ccluster

Ccluster.jl: interface for julia[3] based on Nemo[4]

Available at https://github.com/rimbach/Ccluster.jl

[1] https://github.com/wbhart/flint2

[2] http://arblib.org/

[3] https://julialang.org/

[4] http://nemocas.org/

Improved soft Pellet test

$\tilde{T}_k^G(\Delta, k, [f])$ *//Output in* $\{-1, 0, 1, \ldots, k\}$

... *//soft version of GraeffePelletTest*(Δ, k, f)

GraeffePelletTest(Δ, k, f) *//Output in* $\{-1, 0, 1, \ldots, k\}$

1. compute $f_\Delta^{[N]}$

2. **for** m **from** 0 **to** k **do**
3. **if** $|(f_\Delta^{[N]})_m| > \sum_{i \neq k} |(f_\Delta^{[N]})_i|$
4. **return** m
5. **return** -1

Improved soft Pellet test

$\tilde{T}^G_k(\Delta, k, [f])$ *//Output in* $\{-1, 0, 1, \ldots, k\}$

... *//soft version of GraeffePelletTest*(Δ, k, f)

GraeffePelletTest(Δ, k, f) *//Output in* $\{-1, 0, 1, \ldots, k\}$

1. compute f_Δ
1.b for n **from** 0 **to** N **do**
1.c compute $f_\Delta^{[n]}$
2. **for** m **from** 0 **to** k **do**
3. **if** $|(f_\Delta^{[n]})_m| > \sum_{i \neq k} |(f_\Delta^{[n]})_i|$
4. **return** m
5. **return** -1

Improved soft Pellet test: results

Benchmark:

V1: `Ccluster`: original version

V2: `Ccluster`: with improved soft Pellet test

Table: $\epsilon = 2^{-53}$, $B = [-50, 50]^2$

	V1		V2	
	(n1, n3)	tV1	(n1, n3)	tV2/tV1
Bern., $d = 64$	(2308, 20440)	10.6	(2308, 6031)	2.96
Mign., $d = 64$, $\sigma = 14$	(2060, 18018)	9.42	(2060, 5326)	3.03
Bern., $d = 128$	(4676, 42077)	86.1	(4676, 12049)	3.46
Mign., $d = 128$, $\sigma = 14$	(3900, 36281)	75.3	(3900, 10007)	3.55
Bern., $d = 256$	(9572, 98152)	1024	(9572, 27059)	3.75
Mign., $d = 256$, $\sigma = 14$	(8756, 89864)	945	(8756, 24309)	3.81

Notations:

n1: number of discarding tests

n3: number of Graeffe iterations

Counting instead of discarding

$\tilde{T}_k^G(\Delta, k, [f])$ *//Output in* $\{-1, 0, 1, \ldots, k\}$

... *//soft version of GraeffePelletTest*(Δ, k, f)

Discarding test:

$T_0(\Delta, [f])$ *//Output in* $\{-1, 0\}$

1. return $\tilde{T}_k^G(\Delta, 0, [f])$

Counting test:

$T_*(\Delta, [f])$ *//Output in* $\{-1, 0, 1, \ldots, d\}$

1. return $\tilde{T}_k^G(\Delta, d, [f])$

Counting instead of discarding

$\tilde{T}^G_k(\Delta, k, [f])$ *//Output in* $\{-1, 0, 1, \ldots, k\}$

... *//soft version of GraeffePelletTest*(Δ, k, f)

GraeffePelletTest(Δ, k, f) *//Output in* $\{-1, 0, 1, \ldots, k\}$

1. compute f_Δ
1.b **for** n **from** 0 **to** N **do**
1.c compute $f_\Delta^{[n]}$
2. **for** m **from** 0 **to** k **do**
3. **if** $|(f_\Delta^{[n]})_m| > \sum_{i \neq k} |(f_\Delta^{[n]})_i|$
4. **return** m
5. **return** -1

Counting instead of discarding: results

Benchmark:

V1: `Ccluster`: original version

V2: `Ccluster`: with improved soft Pellet test

V3: V2 with counting instead of discarding

Table: $\epsilon = 2^{-53}$, $B = [-50, 50]^2$

	V1		V3	
	(n1, n3)	tV1	(n1, alert¡1¿n3)	tV3/tV1
Bern., $d = 64$	(2308, 20440)	10.6	(2308, 2292)	7.39
Mign., $d = 64$, $\sigma = 14$	(2060, 18018)	9.42	(2060, 2080)	7.65
Bern., $d = 128$	(4676, 42077)	86.1	(4676, 4496)	11.2
Mign., $d = 128$, $\sigma = 14$	(3900, 36281)	75.3	(3900, 3899)	11.6
Bern., $d = 256$	(9572, 98152)	1024	(9572, 8847)	20.5
Mign., $d = 256$, $\sigma = 14$	(8756, 89864)	945	(8756, 7605)	20.6

Notations:

n1: number of discarding tests

n3: number of Graeffe iterations

Local vs Global comparison

Benchmark: Bernoulli polynomials
Ccluster local: $B = [-1, 1]^2$, $\epsilon = 2^{-53}$
Ccluster global: $B = [-150, 150]^2$, $\epsilon = 2^{-53}$

Table:

	Ccluster local		Ccluster global			
d	(#Sols:#Clus)	t (s)	(#Sols:#Clus)	t (s)		
64	(4:4)	0.12	(64:64)	2.10		
128	(4:4)	0.34	(128:128)	9.90		
191	(5:5)	0.69	(191:191)	32.5		
256	(4:4)	0.96	(256:256)	60.6		
383	(5:5)	2.06	(383:383)	181		
512	(4:4)	2.87	(512:512)	456		
767	(5:5)	6.09	(767:767)	1413		

External comparison

Benchmark: Bernoulli polynomials
Ccluster local: $B = [-1, 1]^2$, $\epsilon = 2^{-53}$
Ccluster global: $B = [-150, 150]^2$, $\epsilon = 2^{-53}$
secsolve: secular algorithm of mpsolve
fsolve: Maple univariate solver

Table:

	Ccluster local		Ccluster global		secsolve	fsolve
d	(#Sols:#Clus)	t (s)	(#Sols:#Clus)	t (s)	t (s)	t (s)
64	(4:4)	0.12	(64:64)	2.10	0.01	0.1
128	(4:4)	0.34	(128:128)	9.90	0.05	6.84
191	(5:5)	0.69	(191:191)	32.5	0.16	50.0
256	(4:4)	0.96	(256:256)	60.6	0.37	$>$ 1000
383	(5:5)	2.06	(383:383)	181	1.17	$>$ 1000
512	(4:4)	2.87	(512:512)	456	3.63	$>$ 1000
767	(5:5)	6.09	(767:767)	1413	10.38	$>$ 1000

Clustering ability

Polynomial with nested clusters of roots: $\texttt{NestClus}_{(D)}(z)$

- has degree $d = 3^D$
- is defined by induction on D:
  - $\texttt{NestClus}_{(1)}(z) = z^3 - 1$ with roots $\omega, \omega^2, \omega^3 = 1$
  - Suppose $\texttt{NestClus}_{(D)}(z)$ has roots $\{r_j | j = 1, \ldots, 3^D\}$, then we define

$$\texttt{NestClus}_{(D+1)}(z) = \prod_{j=1}^{3^D} (z - r_j - \frac{\omega}{16^D})(z - r_j - \frac{\omega^2}{16^D})(z - r_j - \frac{1}{16^D})$$

Notations: $\omega = e^{2\pi i/3}$

Conclusion

`Ccluster`:

- is still a work in progress
- robust to roots with multiplicity
- takes as input any polynomial
- works locally
- is competitive

Thank you!

`https://github.com/rimbach/Ccluster`
`https://github.com/rimbach/Ccluster.jl`