Some Tight Bounds for Genetic Programming on MAJORITY and ORDER

Timo Kötzing

University of Jena, Germany

September 14, 2013

Known Results

What can we do so far?

- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...

Known Results

What can we do so far?

- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...

Known Results

What can we do so far?

- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...

Known Results

What can we do so far?

- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...

Known Results

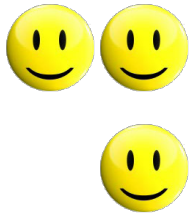
What can we do so far?

- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...

Known Results

What can we do so far?

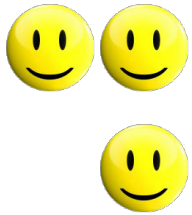
- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...



Known Results

What can we do so far?

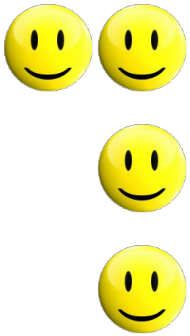
- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...



Known Results

What can we do so far?

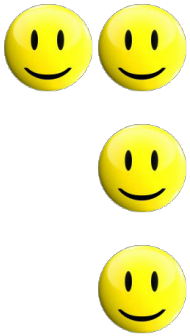
- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...



Known Results

What can we do so far?

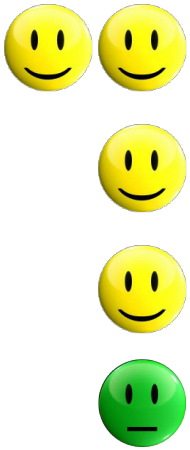
- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...



Known Results

What can we do so far?

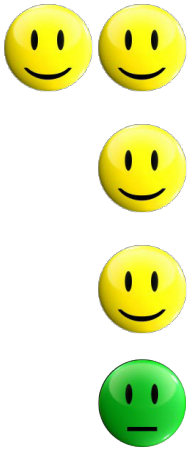
- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...



Known Results

What can we do so far?

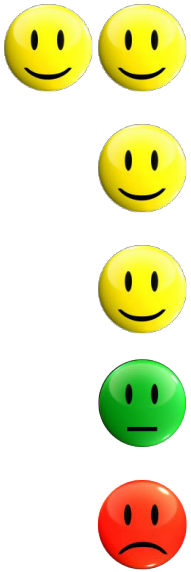
- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- . . .



Known Results

What can we do so far?

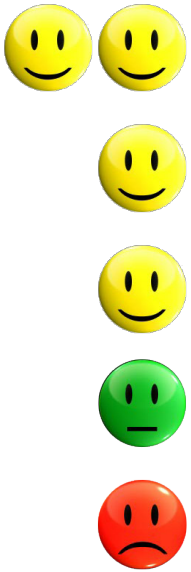
- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- . . .



Known Results

What can we do so far?

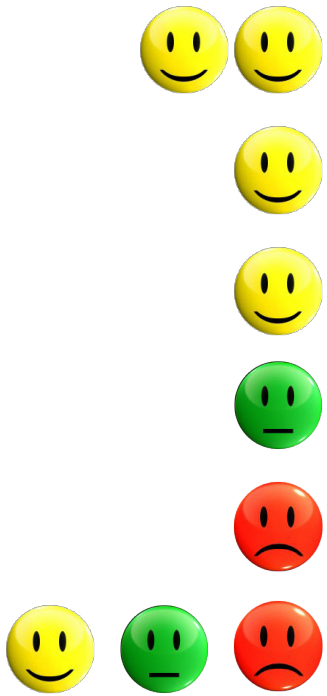
- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...



Known Results

What can we do so far?

- 1+1 EA and RLS on OneMax.
- 1+1 EA and RLS on linear functions.
- ACO on OneMax.
- 1+1 EA and RLS on MST and SSSP.
- ACO on linear functions.
- ...



Majority and Order

- Imagine a list of variable length with entires from $1, \ldots, n$ and $\overline{1}, \ldots, \overline{n}$.

3	$\overline{6}$	1	3	$\overline{3}$	...	2	$\overline{2}$	5

- MAJORITY and ORDER assign a fitness to such lists.

MAJORITY

One fitness point per $i \leq n$ such that at least as many i are listed as $\overline{i}$ (and at least one i is listed).

ORDER

One fitness point per $i \leq n$ such there is an i listed before any $\overline{i}$.

Majority and Order

- Imagine a list of variable length with entires from $1, \ldots, n$ and $\overline{1}, \ldots, \overline{n}$.

3	$\overline{6}$	1	3	$\overline{3}$	$\ldots$	2	$\overline{2}$	5

- MAJORITY and ORDER assign a fitness to such lists.

MAJORITY

One fitness point per $i \leq n$ such that at least as many i are listed as $\overline{i}$ (and at least one i is listed).

ORDER

One fitness point per $i \leq n$ such there is an i listed before any $\overline{i}$.

Majority and Order

- Imagine a list of variable length with entires from $1, \ldots, n$ and $\overline{1}, \ldots, \overline{n}$.

3	$\overline{6}$	1	3	$\overline{3}$	$\ldots$	2	$\overline{2}$	5

- MAJORITY and ORDER assign a fitness to such lists.

MAJORITY

One fitness point per $i \leq n$ such that at least as many i are listed as $\bar{i}$ (and at least one i is listed).

ORDER

One fitness point per $i \leq n$ such there is an i listed before any $\bar{i}$.

Majority and Order

- Imagine a list of variable length with entires from $1, \ldots, n$ and $\overline{1}, \ldots, \overline{n}$.

3	$\overline{6}$	1	3	$\overline{3}$	$\ldots$	2	$\overline{2}$	5

- MAJORITY and ORDER assign a fitness to such lists.

MAJORITY

One fitness point per $i \leq n$ such that at least as many i are listed as $\bar{i}$ (and at least one i is listed).

ORDER

One fitness point per $i \leq n$ such there is an i listed before any $\bar{i}$.

Majority and Order

- Imagine a list of variable length with entires from $1, \ldots, n$ and $\overline{1}, \ldots, \overline{n}$.

3	$\overline{6}$	1	3	$\overline{3}$	$\ldots$	2	$\overline{2}$	5

- MAJORITY and ORDER assign a fitness to such lists.

MAJORITY

One fitness point per $i \leq n$ such that at least as many i are listed as $\overline{i}$ (and at least one i is listed).

ORDER

One fitness point per $i \leq n$ such there is an i listed before any $\overline{i}$.

Majority and Order

- Imagine a list of variable length with entires from $1, \ldots, n$ and $\overline{1}, \ldots, \overline{n}$.

3	$\overline{6}$	1	3	$\overline{3}$	$\ldots$	2	$\overline{2}$	5

- MAJORITY and ORDER assign a fitness to such lists.

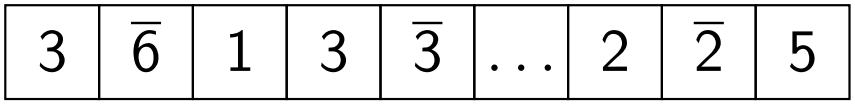
MAJORITY

One fitness point per $i \leq n$ such that at least as many i are listed as $\overline{i}$ (and at least one i is listed).

ORDER

One fitness point per $i \leq n$ such there is an i listed before any $\overline{i}$.

Genetic Programming



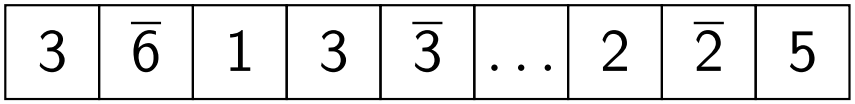
- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming

3	$\overline{6}$	1	3	$\overline{3}$	...	2	$\overline{2}$	5

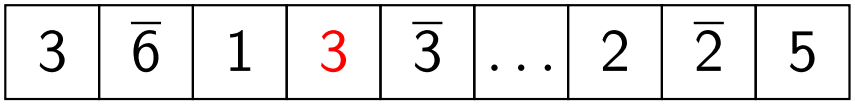
- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming



- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming



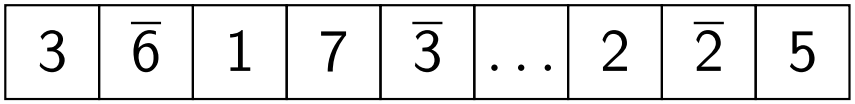
- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming

3	$\overline{6}$	1	7	$\overline{3}$	...	2	$\overline{2}$	5

- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming



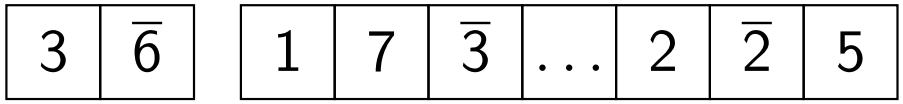
- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming

3	$\overline{6}$	1	7	$\overline{3}$	...	2	$\overline{2}$	5

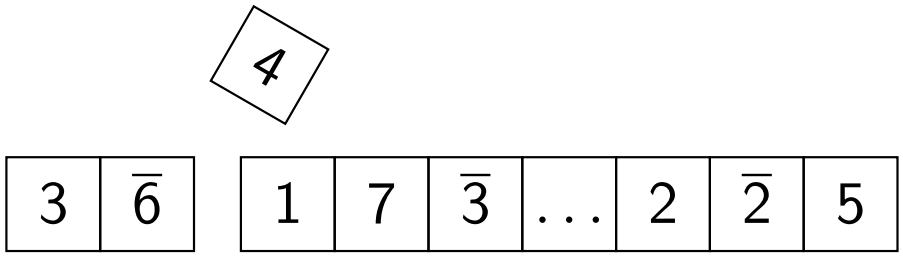
- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming



- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming



- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming

3	$\overline{6}$	4	1	7	$\overline{3}$	...	2	$\overline{2}$	5

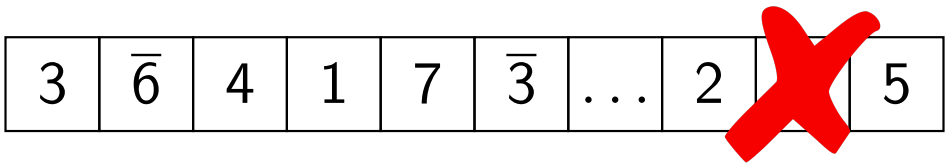
- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming

3	$\overline{6}$	4	1	7	$\overline{3}$	...	2	$\overline{2}$	5

- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming



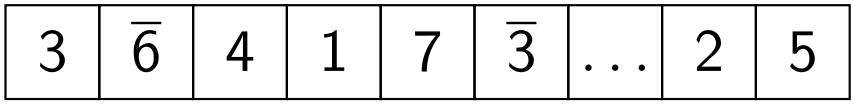
- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming

3	$\overline{6}$	4	1	7	$\overline{3}$	...	2	5

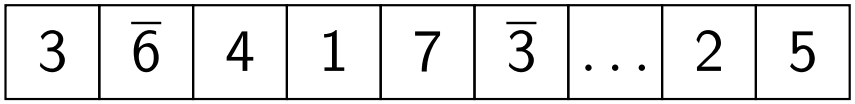
- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming



- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Genetic Programming



- GP algorithms work on these lists with the following randomized operators.
  - Relabel.
  - Insert.
  - Delete.
- Either $k = 1$ uses of operators per offspring or $k = 1+\text{Pois}(1)$.
- Bloat control: In case of equal fitness, favor shorter lists.

Known Results

- What do we know about the performance of GP on MAJORITY and ORDER?

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP on MAJORITY and ORDER?

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP on MAJORITY and ORDER?

F(X)	(1+1)-GP, F(X)	
	k=1	k=1+Pois(1)
ORDER	$O(nT_{max})$ Durrett et al. (2011) $O(T_{init} + n \log n)$ Urli et al. (2012)	$O(nT_{max})$ Durrett et al. (2011) $O(T_{init} + n \log n)$ Urli et al. (2012)
WORDER	$O(nT_{max}) \star$ $O(T_{init} + n \log n)$ Urli et al. (2012)	$O(nT_{max}(\log n + \log w_{max})) \star$ $O(T_{init} + n \log n)$ Urli et al. (2012)
MAJORITY	$O(n^2 T_{max} \log n)$ Durrett et al. (2011) $O(T_{init} + n \log n)$ Urli et al. (2012)	? $O(T_{init} + n \log n)$ Urli et al. (2012)
WMAJORITY	? $O(T_{init} + n \log n)$ Urli et al. (2012)	? $O(T_{init} + n \log n)$ Urli et al. (2012)

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP on MAJORITY and ORDER?

F(X)	(1+1)-GP, F(X)	
	k=1	k=1+Pois(1)
ORDER	$O(nT_{max})$ Durrett et al. (2011) $O(T_{init} + n \log n)$ Urli et al. (2012)	$O(nT_{max})$ Durrett et al. (2011) $O(T_{init} + n \log n)$ Urli et al. (2012)
WORDER	$O(nT_{max}) \star$ $O(T_{init} + n \log n)$ Urli et al. (2012)	$O(nT_{max}(\log n + \log w_{max})) \star$ $O(T_{init} + n \log n)$ Urli et al. (2012)
MAJORITY	$O(n^2 T_{max} \log n)$ Durrett et al. (2011) $O(T_{init} + n \log n)$ Urli et al. (2012)	? $O(T_{init} + n \log n)$ Urli et al. (2
WMAJORITY	? $O(T_{init} + n \log n)$ Urli et al. (2	? $O(T_{init} + n \log n)$ Urli et al. (2012)

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP with bloat control on MAJORITY and ORDER?

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP with bloat control on MAJORITY and ORDER?

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP with bloat control on MAJORITY and ORDER?

F(X)	(1+1)-GP, MO-F(X)	
	k=1	k=1+Pois(1)
ORDER	$O(T_{init} + n \log n)$Neumann (2012) $O(T_{init} + n \log n)$ Urli et al. (2012)	$O(n^2 \log n)$ $\star \dagger$ $O(T_{init} + n \log n)$ Urli et al. (2012)
WORDER	$O(T_{init} + n \log n)$Neumann (2012) $O(T_{init} + n \log n)$ Urli et al. (2012)	? $O(T_{init} + n \log n)$ Urli et al. (2012)
MAJORITY	$O(T_{init} + n \log n)$Neumann (2012) $O(T_{init} + n \log n)$ Urli et al. (2012)	? $O(T_{init} + n \log n)$ Urli et al. (2012)
WMAJORITY	$O(T_{init} + n \log n)$Neumann (2012) $O(T_{init} + n \log n)$ Urli et al. (2012)	? $O(T_{init} + n \log n)$ Urli et al. (2012)

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP with bloat control on MAJORITY and ORDER?

F(X)	(1+1)-GP, MO-F(X)	
	k=1	k=1+Pois(1)
ORDER	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	$O(n^2 \log n)$ $\star$ † $O(T_{init} + n \log n)$ Urli et al. (2012)
WORDER	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (2012)
MAJORITY	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (2012)
WMAJORITY	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (2012)

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP with bloat control on MAJORITY and ORDER?

F(X)	(1+1)-GP, MO-F(X)	
	k=1	k=1+Pois(1)
ORDER	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	$O(n^2 \log n)$ $\star$ † $O(T_{init} + n \log n)$ Urli et al. (2012)
WORDER	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (2012)
MAJORITY	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (2[illegible])
WMAJORITY	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (2012)

- (Source: Nguyen et al. 2013)

Known Results

- What do we know about the performance of GP with bloat control on MAJORITY and ORDER?

F(X)	(1+1)-GP, MO-F(X)	
	k=1	k=1+Pois(1)
ORDER	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	$O(n^2 \log n)$ $\star$ † $O(T_{init} + n \log n)$ Urli et al. (20[illegible])
WORDER	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (2012)
MAJORITY	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (20[illegible])
WMAJORITY	$O(T_{init} + n \log n)$Neumann (20[illegible]) $O(T_{init} + n \log n)$ Urli et al. (20[illegible])	? $O(T_{init} + n \log n)$ Urli et al. (2012)

- (Source: Nguyen et al. 2013)

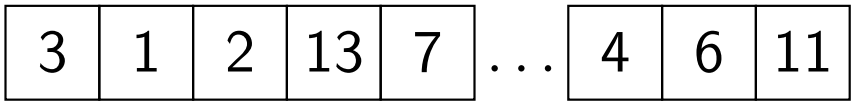
Optimize with Bloat Control

- Optimum (in both cases):

- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

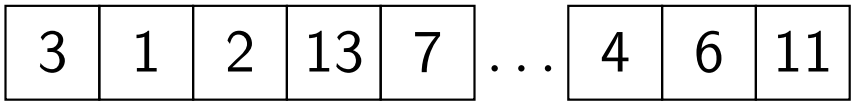
- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

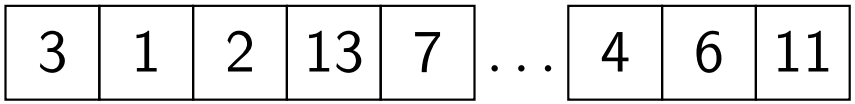
- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

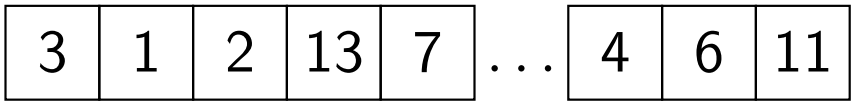
- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

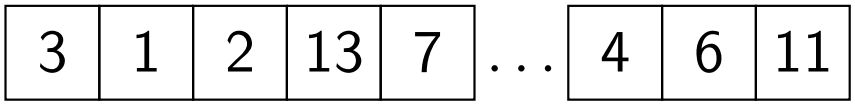
- Optimum (in both cases):

3	1	2	13	7	...	4	6	11

- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

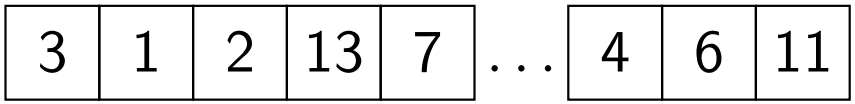
- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

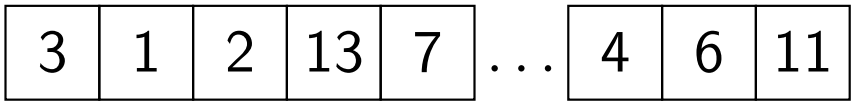
- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

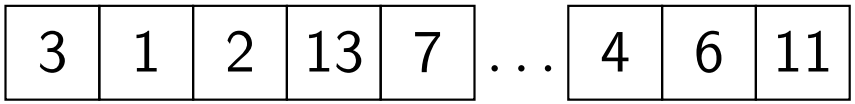
- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

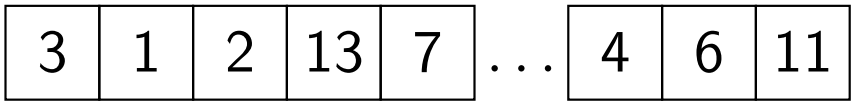
- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

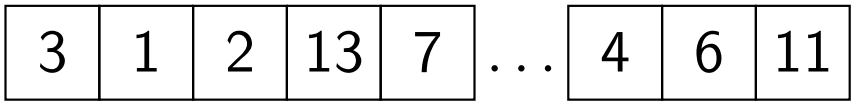
- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

Optimize with Bloat Control

- Optimum (in both cases):



- Let $v(t)$ be the number of i such that i occurs in list t.
- Let $r(t)$ be the number of redundant entries in list t.
- Drift function: $f(t) = (n - v(t))5 + r(t)$.
- All and only optimal lists have value of 0.
- If $r(t) \geq v(t)$: chance of deleting redundant entry is high
- $\Rightarrow$ constant additive drift!
- If $r(t) < v(t)$: either delete redundant or add new entry
- $\Rightarrow$ multiplicative drift!
- Overall: Variable drift, which comes out as $O(T_{\text{init}} + n \log n)$.

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\text{max}} \log T_{\text{init}} + n \log n)$.
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\text{max}} \log T_{\text{init}} + n \log n)$.
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\text{max}} \log T_{\text{init}} + n \log n)$.
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\text{max}} \log T_{\text{init}} + n \log n)$.
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\text{max}} \log T_{\text{init}} + n \log n)$.
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\text{max}} \log T_{\text{init}} + n \log n)$.
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\text{max}} \log T_{\text{init}} + n \log n)$.
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\text{max}} \log T_{\text{init}} + n \log n)$.
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\max} \log T_{\text{init}} + n \log n)$. 
- Bound on maximal list length: Gambler's Ruin?

MAJORITY without Bloat Control

- For each i, let $v_i(t)$ be the number of $\overline{x_i}$ minus the number of x_i (at least 0, plus 1 if there is no x_i).
- Drift function: $f(t) = \sum_{i=1}^{n} v_i(t)$.
- All and only optimal lists have value of 0.
- If, for some i, there are more $\overline{x_i}$ than x_i, then it's more likely to delete a $\overline{x_i}$ than x_i.
- $\overline{x_i}$ and x_i are inserted in a balanced way.
- $\Rightarrow$ multiplicative drift!
- Overall: $O(T_{\max} \log T_{\text{init}} + n \log n)$. 
- Bound on maximal list length: Gambler's Ruin?

ORDER without Bloat Control

- This seems to be harder than MAJORITY.

- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound...
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY ...
- ...for a good bound, it seems to be the reverse.

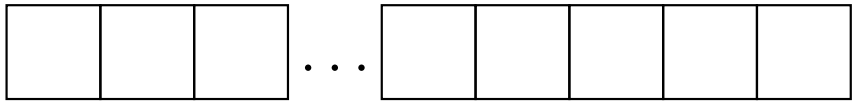
ORDER without Bloat Control

- This seems to be harder than MAJORITY.

- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound...
- Best known: $O(T_{\text{max}} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY ...
- ...for a good bound, it seems to be the reverse.

ORDER without Bloat Control

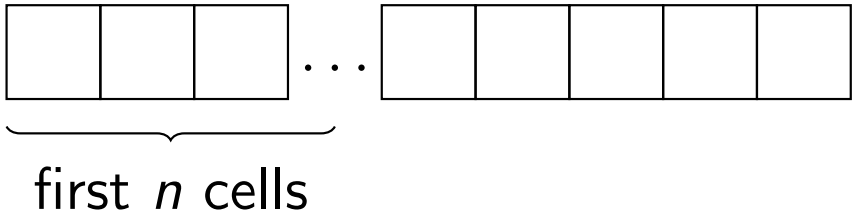
- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound...
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY ...
- ...for a good bound, it seems to be the reverse.

ORDER without Bloat Control

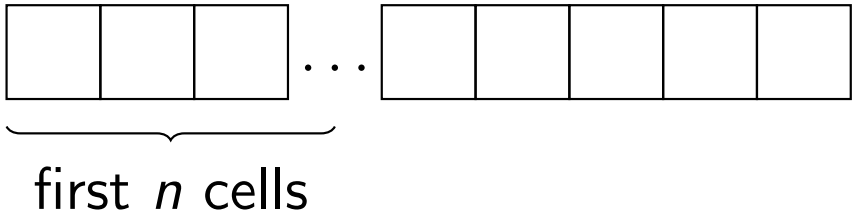
- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound...
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY ...
- ...for a good bound, it seems to be the reverse.

ORDER without Bloat Control

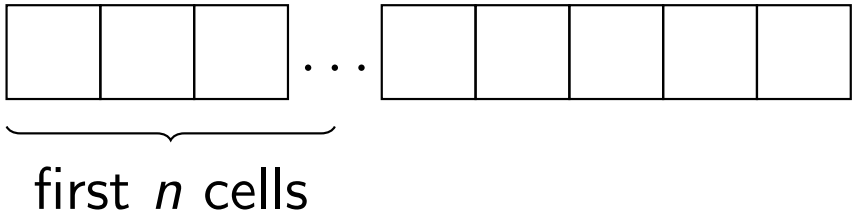
- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound...
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY ...
- ...for a good bound, it seems to be the reverse.

ORDER without Bloat Control

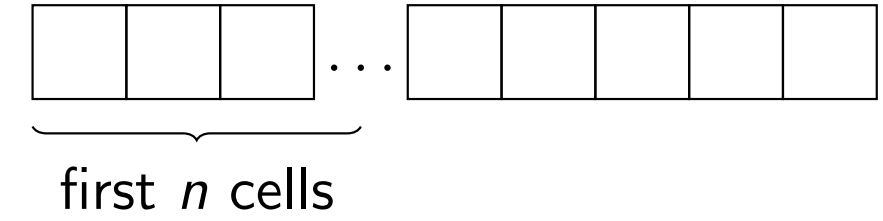
- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound...
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY ...
- ...for a good bound, it seems to be the reverse.

ORDER without Bloat Control

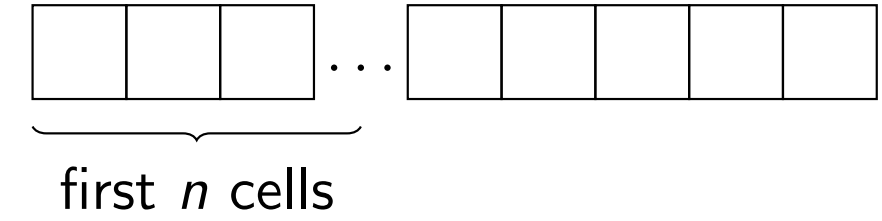
- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound. . .
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY . . .
- . . . for a good bound, it seems to be the reverse.

ORDER without Bloat Control

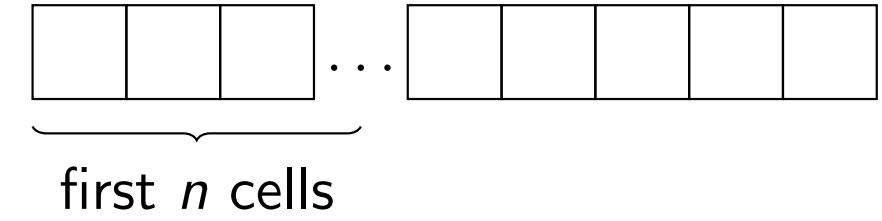
- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound. . .
- Best known: $O(T_{\text{max}} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY . . .
- . . . for a good bound, it seems to be the reverse.

ORDER without Bloat Control

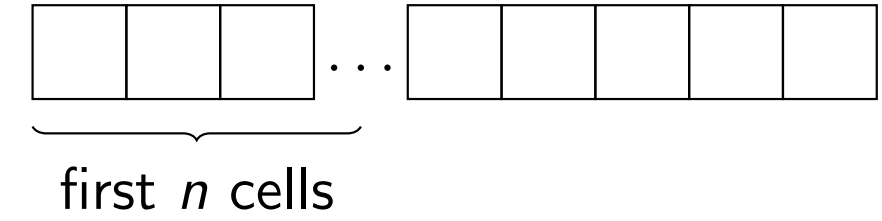
- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound...
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY ...
- ...for a good bound, it seems to be the reverse.

ORDER without Bloat Control

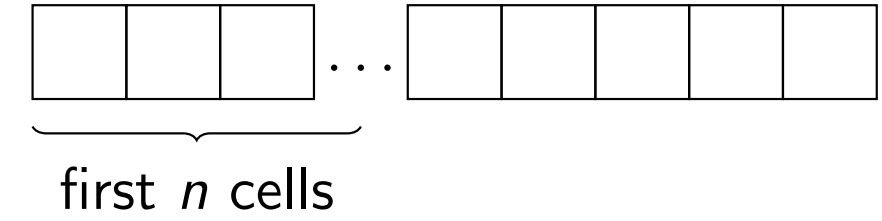
- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound. . .
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY . . .
- . . . for a good bound, it seems to be the reverse.

ORDER without Bloat Control

- This seems to be harder than MAJORITY.



- Bound on maximal list length again?
- $\Rightarrow$ mixing time for first n cells: $O(T_{\text{init}} \log n + n \log n)$.
- Sounds like a good bound...
- Best known: $O(T_{\max} n)$.
- A semi-decent bound on ORDER is much easier than on MAJORITY ...
- ... for a good bound, it seems to be the reverse.

More Work?

- What else is there?
- Random number of mutations with weighted case.
- Some Multi-Objective variants are also still open.
- What can we learn from abstract results?
- $\Rightarrow$ implicit bloat control: Make delete more likely than insert.
- Can we get results as with bloat control?

More Work?

- What else is there?
- Random number of mutations with weighted case.
- Some Multi-Objective variants are also still open.
- What can we learn from abstract results?
- $\Rightarrow$ implicit bloat control: Make delete more likely than insert.
- Can we get results as with bloat control?

More Work?

- What else is there?
- Random number of mutations with weighted case.
- Some Multi-Objective variants are also still open.
- What can we learn from abstract results?
- $\Rightarrow$ implicit bloat control: Make delete more likely than insert.
- Can we get results as with bloat control?

More Work?

- What else is there?
- Random number of mutations with weighted case.
- Some Multi-Objective variants are also still open.
- What can we learn from abstract results?
- $\Rightarrow$ implicit bloat control: Make delete more likely than insert.
- Can we get results as with bloat control?

More Work?

- What else is there?
- Random number of mutations with weighted case.
- Some Multi-Objective variants are also still open.
- What can we learn from abstract results?
- $\Rightarrow$ implicit bloat control: Make delete more likely than insert.
- Can we get results as with bloat control?

More Work?

- What else is there?
- Random number of mutations with weighted case.
- Some Multi-Objective variants are also still open.
- What can we learn from abstract results?
- $\Rightarrow$ implicit bloat control: Make delete more likely than insert.
- Can we get results as with bloat control?

More Work?

- What else is there?
- Random number of mutations with weighted case.
- Some Multi-Objective variants are also still open.
- What can we learn from abstract results?
- $\Rightarrow$ implicit bloat control: Make delete more likely than insert.
- Can we get results as with bloat control?

Thank You!