

CS4618 Artificial Intelligence I

Today: Parametrisation:
Offspring Population Size
& Mutation Probability

Thomas Jansen

December 5th

Plans for Today

- 1 In-Class Test
Outcome
Problems and Solutions
- 2 ONEMAX
Function and Intuition
Result and Proof
- 3 Example for a Speed-Up
Idea and Function
- 4 Setting the Mutation Probability
Motivation
ONEMAX
- 5 Summary
Summary & Take Home Message



Results for the In-Class Test




Results for the In-Class Test

more personal details available from me on request

Results for the In-Class Test

more personal details available from me on request

Result Distribution



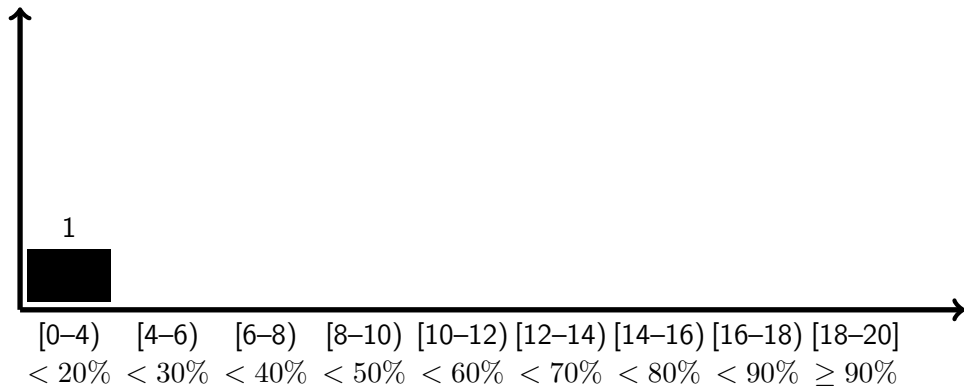
Score Range	Percentage
[0-4)	< 20%
[4-6)	< 30%
[6-8)	< 40%
[8-10)	< 50%
[10-12)	< 60%
[12-14)	< 70%
[14-16)	< 80%
[16-18)	< 90%
[18-20]	≥ 90%



Results for the In-Class Test

more personal details available from me on request

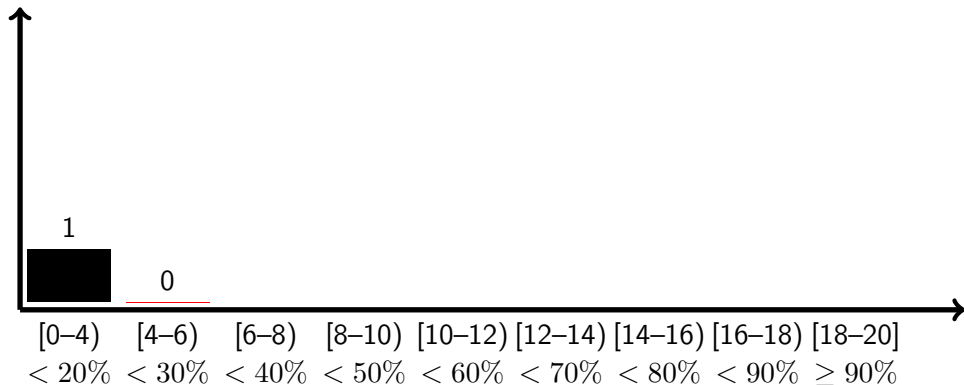
Result Distribution



Results for the In-Class Test

more personal details available from me on request

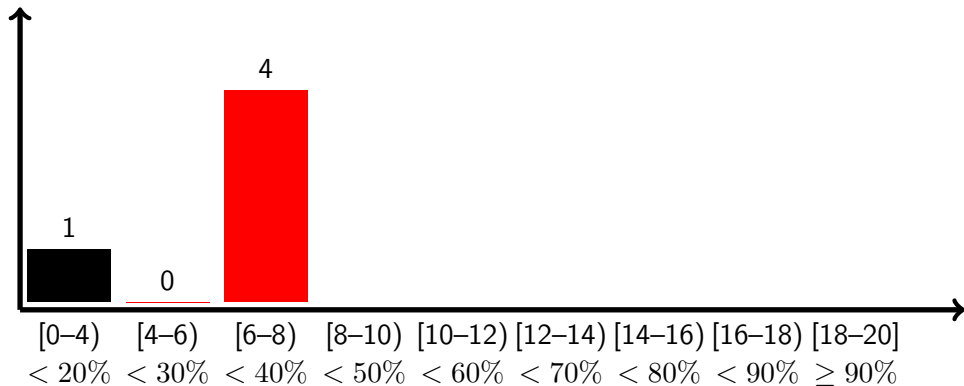
Result Distribution



Results for the In-Class Test

more personal details available from me on request

Result Distribution

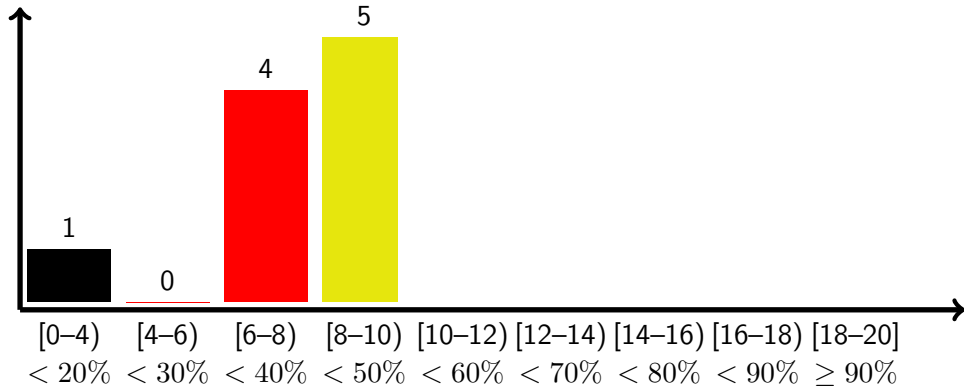




Results for the In-Class Test

more personal details available from me on request

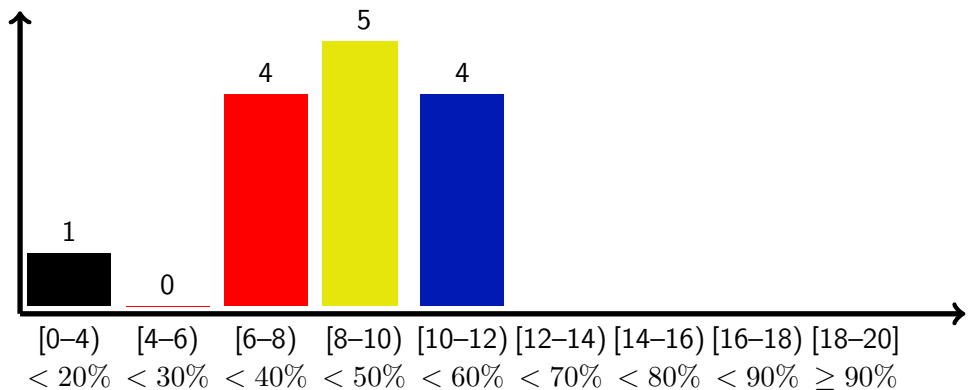
Result Distribution



Results for the In-Class Test

more personal details available from me on request

Result Distribution

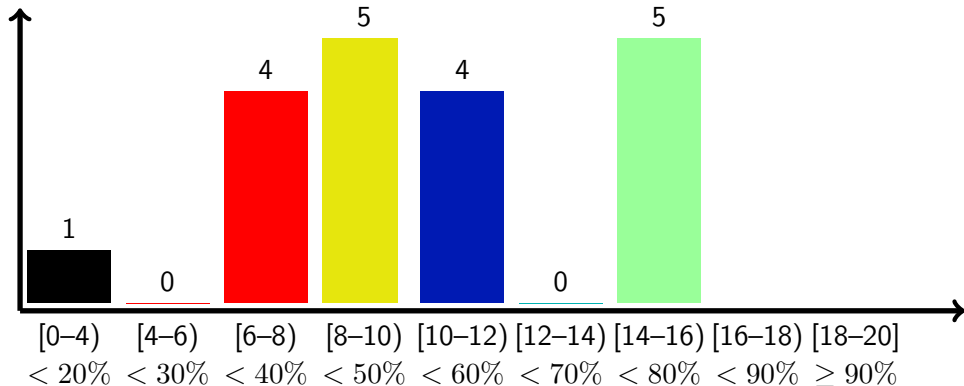




Results for the In-Class Test

more personal details available from me on request

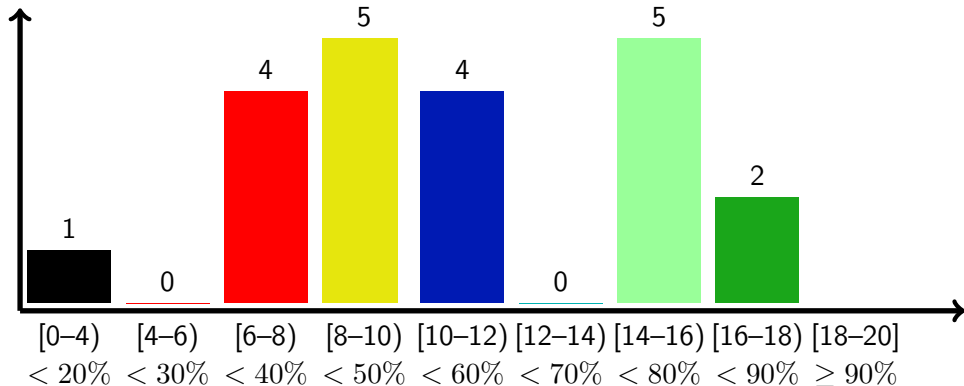
Result Distribution



Results for the In-Class Test

more personal details available from me on request

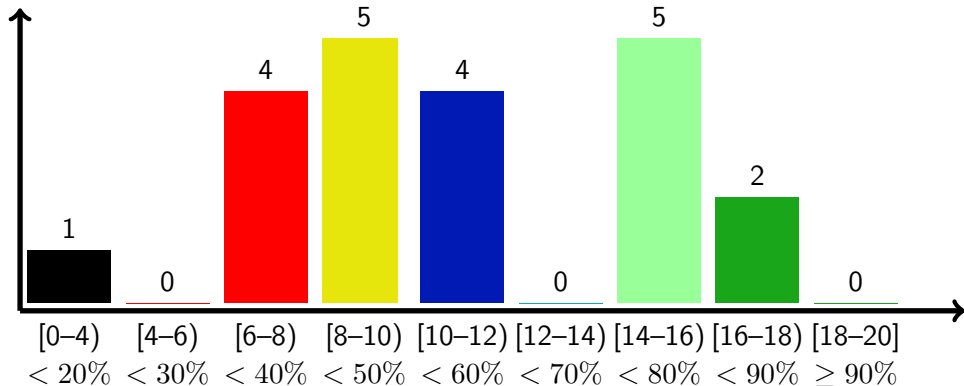
Result Distribution



Results for the In-Class Test

more personal details available from me on request

Result Distribution



Statistics

median 11 $\hat{=}$ 55%
 minimum 4 $\hat{=}$ 20%

average 11 $\hat{=}$ 55%
 maximum 17 $\hat{=}$ 85%

Topic I) Task Environments

Give a PEAS description for a sorting machine at a recycling facility. The machine is supposed to sort glass bottles with respect to the colour of the glass: clear, brown, green. Classify the corresponding task environment using the list of seven 'dimensions' from the lecture.

Topic I) Task Environments

Give a PEAS description for a sorting machine at a recycling facility. The machine is supposed to sort glass bottles with respect to the colour of the glass: clear, brown, green. Classify the corresponding task environment using the list of seven 'dimensions' from the lecture.

- **performance measure** number of correctly sorted bottles per minute
- **environment** bottles and parts of the machine
- **actuators** robot arm or strong air blow or ...
- **sensors** camera (or possibly only a light sensor)

Topic I) Task Environments

Give a PEAS description for a sorting machine at a recycling facility. The machine is supposed to sort glass bottles with respect to the colour of the glass: clear, brown, green. Classify the corresponding task environment using the list of seven 'dimensions' from the lecture.

- **performance measure** number of correctly sorted bottles per minute
- **environment** bottles and parts of the machine
- **actuators** robot arm or strong air blow or ...
- **sensors** camera (or possibly only a light sensor)
- **properties** fully observable, single agent, stochastic, episodic, dynamic, continuous, known

Topic I) Task Environments

Give a PEAS description for a sorting machine at a recycling facility. The machine is supposed to sort glass bottles with respect to the colour of the glass: clear, brown, green. Classify the corresponding task environment using the list of seven 'dimensions' from the lecture.

- **performance measure** number of correctly sorted bottles per minute
- **environment** bottles and parts of the machine
- **actuators** robot arm or strong air blow or ...
- **sensors** camera (or possibly only a light sensor)
- **properties** fully observable, single agent, stochastic, episodic, dynamic, continuous, known

min 1 median 2 average 1.76 max 2

Topic II) Rational Agents

Compare a table-driven agent to a simple-reflex agent.

Topic II) Rational Agents

Compare a table-driven agent to a simple-reflex agent.

table-driven agent

- stores list of percepts ($\hat{=}$ complete history)
- has table mapping lists of percepts to actions
- + able to solve every task
- table usually huge, possibly infinite \rightsquigarrow not practical

Topic II) Rational Agents

Compare a table-driven agent to a simple-reflex agent.

table-driven agent

- stores list of percepts ($\hat{=}$ complete history)
- has table mapping lists of percepts to actions
- + able to solve every task
- table usually huge, possibly infinite \rightsquigarrow not practical

simple-reflex agent

- has table mapping single percepts to actions
- uses no/little memory
- + simple, efficient
- restricted, not able to solve all problems

Topic II) Rational Agents

Compare a table-driven agent to a simple-reflex agent.

table-driven agent

- stores list of percepts ($\hat{=}$ complete history)
- has table mapping lists of percepts to actions
- + able to solve every task
- table usually huge, possibly infinite \rightsquigarrow not practical

simple-reflex agent

- has table mapping single percepts to actions
- uses no/little memory
- + simple, efficient
- restricted, not able to solve all problems

min 0 median 2 average 1.62 max 2

Topic III) Uninformed Search

Remember Knuth's number game. Consider using BFS or DFS for solving this puzzle and compare the two algorithms.

Topic III) Uninformed Search

Remember Knuth's number game. Consider using BFS or DFS for solving this puzzle and compare the two algorithms.

BFS

- + complete ($\hat{=}$ guaranteed to find a solution)
- + optimal (since all actions have equal cost)
- may still take a long time, run out of memory, ...

Topic III) Uninformed Search

Remember Knuth's number game. Consider using BFS or DFS for solving this puzzle and compare the two algorithms.

BFS

- + complete ($\hat{=}$ guaranteed to find a solution)
- + optimal (since all actions have equal cost)
- may still take a long time, run out of memory, ...

DFS

- + may be lucky and more efficient than BFS
- not complete, **likely** to run indefinitely without success

Topic III) Uninformed Search

Remember Knuth's number game. Consider using BFS or DFS for solving this puzzle and compare the two algorithms.

BFS

- + complete ($\hat{=}$ guaranteed to find a solution)
- + optimal (since all actions have equal cost)
- may still take a long time, run out of memory, ...

DFS

- + may be lucky and more efficient than BFS
- not complete, **likely** to run indefinitely without success

Consequence prefer BFS over DFS

Topic III) Uninformed Search

Remember Knuth's number game. Consider using BFS or DFS for solving this puzzle and compare the two algorithms.

BFS

- + complete ($\hat{=}$ guaranteed to find a solution)
- + optimal (since all actions have equal cost)
- may still take a long time, run out of memory, ...

DFS

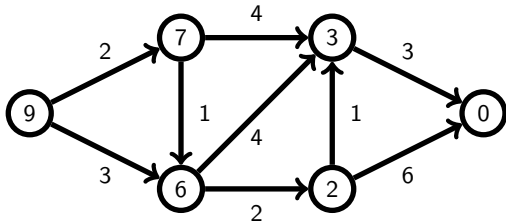
- + may be lucky and more efficient than BFS
- not complete, **likely** to run indefinitely without success

Consequence prefer BFS over DFS

min 0 median 2 average 1.69 max 2

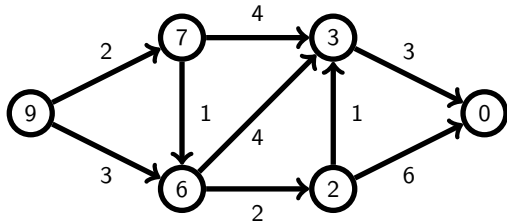
Topic IV) Informed Search

The graph below specifies a search problem. States are nodes, the start state is 9, the only goal state is 0. Edges describe actions and their outcome, the number on an edge gives its cost. We use the number in the nodes as heuristic. For each node write next to it the minimal cost to reach the goal state from there. Decide if this heuristic is admissible. Decide if this heuristic is consistent.



Topic IV) Informed Search

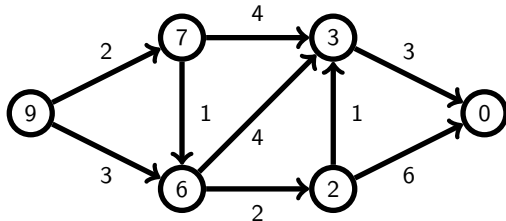
The graph below specifies a search problem. States are nodes, the start state is 9, the only goal state is 0. Edges describe actions and their outcome, the number on an edge gives its cost. We use the number in the nodes as heuristic. For each node write next to it the minimal cost to reach the goal state from there. Decide if this heuristic is admissible. Decide if this heuristic is consistent.



- cost equal to heuristic, except for 2 with cost 4

Topic IV) Informed Search

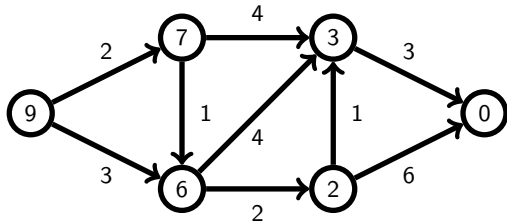
The graph below specifies a search problem. States are nodes, the start state is 9, the only goal state is 0. Edges describe actions and their outcome, the number on an edge gives its cost. We use the number in the nodes as heuristic. For each node write next to it the minimal cost to reach the goal state from there. Decide if this heuristic is admissible. Decide if this heuristic is consistent.



- cost equal to heuristic, except for 2 with cost 4
- **admissible** since no overestimation

Topic IV) Informed Search

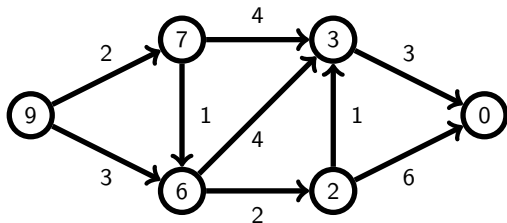
The graph below specifies a search problem. States are nodes, the start state is 9, the only goal state is 0. Edges describe actions and their outcome, the number on an edge gives its cost. We use the number in the nodes as heuristic. For each node write next to it the minimal cost to reach the goal state from there. Decide if this heuristic is admissible. Decide if this heuristic is consistent.



- cost equal to heuristic, except for 2 with cost 4
- **admissible** since no overestimation
- **not consistent** since $h(6) > 2 + h(2)$

Topic IV) Informed Search

The graph below specifies a search problem. States are nodes, the start state is 9, the only goal state is 0. Edges describe actions and their outcome, the number on an edge gives its cost. We use the number in the nodes as heuristic. For each node write next to it the minimal cost to reach the goal state from there. Decide if this heuristic is admissible. Decide if this heuristic is consistent.

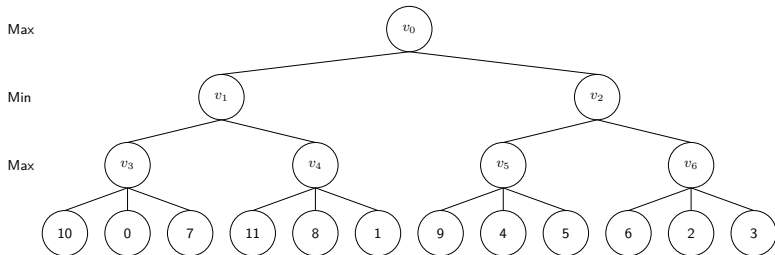


- cost equal to heuristic, except for 2 with cost 4
- **admissible** since no overestimation
- **not consistent** since $h(6) > 2 + h(2)$

min 0 median 1 average 1.05 max 2

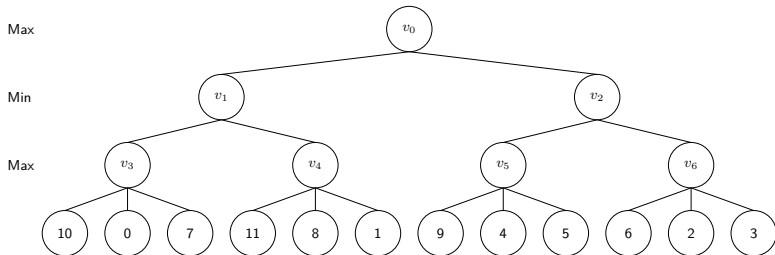
Topic V) Alpha-Beta Pruning

Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



Topic V) Alpha-Beta Pruning

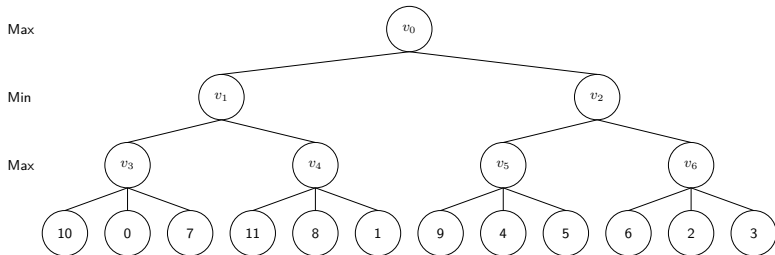
Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



- values v_3 : 10

Topic V) Alpha-Beta Pruning

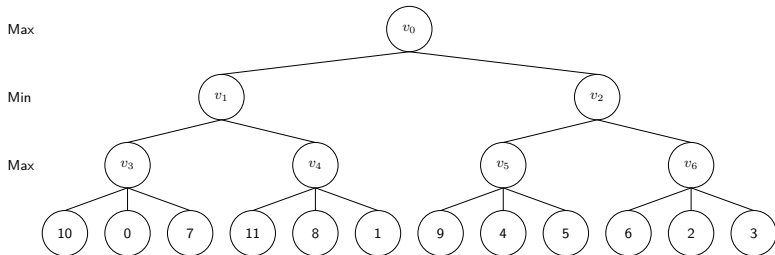
Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



- values v_3 : 10, v_4 : 11

Topic V) Alpha-Beta Pruning

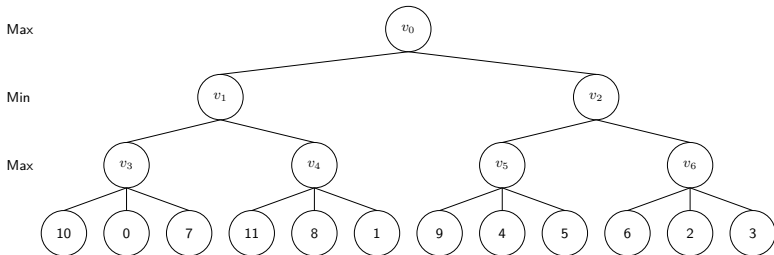
Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



- values v_3 : 10, v_4 : 11, v_1 : 10

Topic V) Alpha-Beta Pruning

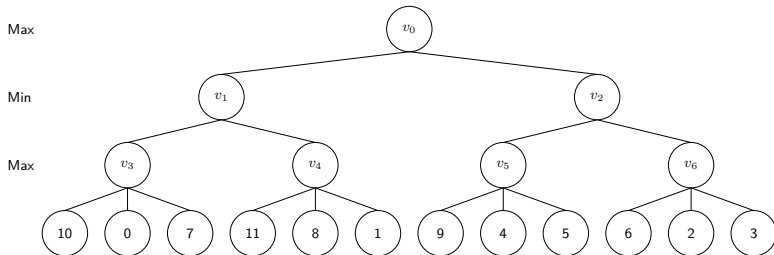
Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



- values $v_3: 10, v_4: 11, v_1: 10, v_5: 9$

Topic V) Alpha-Beta Pruning

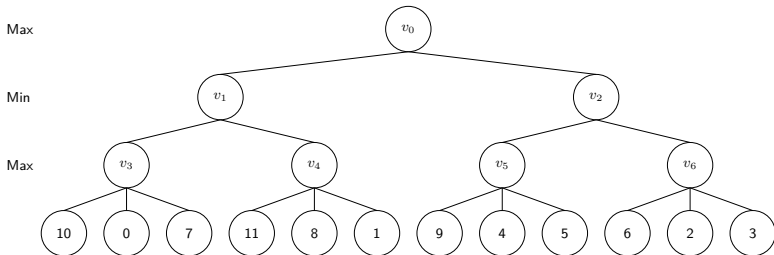
Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



- values $v_3: 10, v_4: 11, v_1: 10, v_5: 9, v_2: 9$

Topic V) Alpha-Beta Pruning

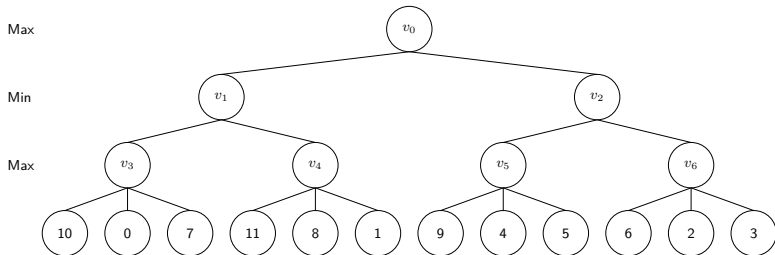
Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



- values $v_3: 10, v_4: 11, v_1: 10, v_5: 9, v_2: 9, v_0: 10$

Topic V) Alpha-Beta Pruning

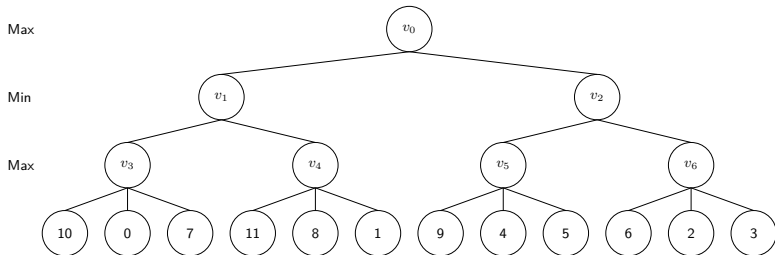
Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



- **values** v_3 : 10, v_4 : 11, v_1 : 10, v_5 : 9, v_2 : 9, v_0 : 10
- **not explored** 8, 1, v_6 , 6, 2, 3

Topic V) Alpha-Beta Pruning

Evaluate the game tree given below using alpha-beta pruning. Consider the moves from left to right. Write down a list of the states that are not considered. Write next to each state that is evaluated the value that is computed by alpha-beta pruning.



- values $v_3: 10, v_4: 11, v_1: 10, v_5: 9, v_2: 9, v_0: 10$
- not explored 8, 1, $v_6, 6, 2, 3$

min 0 median 1 average 1.26 max 2

Topic VI) Adversarial Search

What is a transposition table and why is it useful?

Topic VI) Adversarial Search

What is a transposition table and why is it useful?

- hash table for states

Topic VI) Adversarial Search

What is a transposition table and why is it useful?

- hash table for states
- helps to avoid re-evaluation of states

Topic VI) Adversarial Search

What is a transposition table and why is it useful?

- hash table for states
- helps to avoid re-evaluation of states

min 0 median 1 average 1 max 2

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Remember $\text{Prob}(\text{get next new type when having } i) = (n - i)/n$

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Remember $\text{Prob}(\text{get next new type when having } i) = (n - i)/n$

Thus
$$E(T) = \sum_{i=0}^{(9/10)n-1} \frac{n}{n-i}$$

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Remember $\text{Prob}(\text{get next new type when having } i) = (n - i)/n$

Thus
$$E(T) = \sum_{i=0}^{(9/10)n-1} \frac{n}{n-i} = n \sum_{i=(n/10)+1}^n \frac{1}{i}$$

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Remember $\text{Prob}(\text{get next new type when having } i) = (n - i)/n$

Thus

$$E(T) = \sum_{i=0}^{(9/10)n-1} \frac{n}{n-i} = n \sum_{i=(n/10)+1}^n \frac{1}{i}$$

$$= n \left(\left(\sum_{i=1}^n \frac{1}{i} \right) - \left(\sum_{i=1}^{n/10} \frac{1}{i} \right) \right)$$

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Remember $\text{Prob}(\text{get next new type when having } i) = (n - i)/n$

Thus

$$E(T) = \sum_{i=0}^{(9/10)n-1} \frac{n}{n-i} = n \sum_{i=(n/10)+1}^n \frac{1}{i}$$

$$= n \left(\left(\sum_{i=1}^n \frac{1}{i} \right) - \left(\sum_{i=1}^{n/10} \frac{1}{i} \right) \right) = n (\ln(n) - \ln(n/10) + O(1))$$

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Remember $\text{Prob}(\text{get next new type when having } i) = (n - i)/n$

Thus

$$\begin{aligned}
 E(T) &= \sum_{i=0}^{(9/10)n-1} \frac{n}{n-i} = n \sum_{i=(n/10)+1}^n \frac{1}{i} \\
 &= n \left(\left(\sum_{i=1}^n \frac{1}{i} \right) - \left(\sum_{i=1}^{n/10} \frac{1}{i} \right) \right) = n (\ln(n) - \ln(n/10) + O(1)) \\
 &= n (\ln(n) - \ln(n) + \ln(10) + O(1))
 \end{aligned}$$

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Remember $\text{Prob}(\text{get next new type when having } i) = (n - i)/n$

Thus

$$\begin{aligned}
 E(T) &= \sum_{i=0}^{(9/10)n-1} \frac{n}{n-i} = n \sum_{i=(n/10)+1}^n \frac{1}{i} \\
 &= n \left(\left(\sum_{i=1}^n \frac{1}{i} \right) - \left(\sum_{i=1}^{n/10} \frac{1}{i} \right) \right) = n (\ln(n) - \ln(n/10) + O(1)) \\
 &= n (\ln(n) - \ln(n) + \ln(10) + O(1)) = O(n)
 \end{aligned}$$

Topic VII) Random Processes

Consider the coupon collector's scenario with n different types of coupons. Assume that you are satisfied with your collection when you have 90% of all different coupons. Let T be the number of coupons collected until your collection contains at least 90% of all different types of coupons. Prove an upper bound on $E(T)$.

Remember $\text{Prob}(\text{get next new type when having } i) = (n - i)/n$

Thus

$$\begin{aligned}
 E(T) &= \sum_{i=0}^{(9/10)n-1} \frac{n}{n-i} = n \sum_{i=(n/10)+1}^n \frac{1}{i} \\
 &= n \left(\left(\sum_{i=1}^n \frac{1}{i} \right) - \left(\sum_{i=1}^{n/10} \frac{1}{i} \right) \right) = n (\ln(n) - \ln(n/10) + O(1)) \\
 &= n (\ln(n) - \ln(n) + \ln(10) + O(1)) = O(n)
 \end{aligned}$$

min 0 **median** 1 **average** 0.76 **max** 2

Topic VIII) Black-Box Complexity

We call a function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ a function of unitation if its function value for x only depends on the number of 1-bits in x and not x itself. Let U denote the set of all functions of unitation. Prove an upper bound on the black-box complexity of U .

Topic VIII) Black-Box Complexity

We call a function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ a function of unitation if its function value for x only depends on the number of 1-bits in x and not x itself. Let U denote the set of all functions of unitation. Prove an upper bound on the black-box complexity of U .

Observation There are $\leq n + 1$ different function values

Topic VIII) Black-Box Complexity

We call a function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ a function of unitation if its function value for x only depends on the number of 1-bits in x and not x itself. Let U denote the set of all functions of unitation. Prove an upper bound on the black-box complexity of U .

Observation There are $\leq n + 1$ different function values and we get all by sampling $0^n, 10^{n-1}, 110^{n-2}, \dots, 1^n$.

Topic VIII) Black-Box Complexity

We call a function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ a function of unitation if its function value for x only depends on the number of 1-bits in x and not x itself. Let U denote the set of all functions of unitation. Prove an upper bound on the black-box complexity of U .

Observation There are $\leq n + 1$ different function values and we get all by sampling $0^n, 10^{n-1}, 110^{n-2}, \dots, 1^n$.

Consequence $\mathcal{B}_U \leq n + 1$

Topic VIII) Black-Box Complexity

We call a function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ a function of unitation if its function value for x only depends on the number of 1-bits in x and not x itself. Let U denote the set of all functions of unitation. Prove an upper bound on the black-box complexity of U .

Observation There are $\leq n + 1$ different function values and we get all by sampling $0^n, 10^{n-1}, 110^{n-2}, \dots, 1^n$.

Consequence $\mathcal{B}_U \leq n + 1$

min 0 **median** 0 **average** 0.1 **max** 2

Topic IX) Applying Randomised Search Heuristics

Consider attempting to solve a difficult optimisation problem with random local search. Discuss advantages and disadvantages of increasing the neighbourhood from the set of all Hamming neighbours to all points with a Hamming distance of at most 2.

Topic IX) Applying Randomised Search Heuristics

Consider attempting to solve a difficult optimisation problem with random local search. Discuss advantages and disadvantages of increasing the neighbourhood from the set of all Hamming neighbours to all points with a Hamming distance of at most 2.

- + the increased neighbourhood can help to escape from local optima

Topic IX) Applying Randomised Search Heuristics

Consider attempting to solve a difficult optimisation problem with random local search. Discuss advantages and disadvantages of increasing the neighbourhood from the set of all Hamming neighbours to all points with a Hamming distance of at most 2.

- + the increased neighbourhood can help to escape from local optima
 - one can still get stuck in some local optima

Topic IX) Applying Randomised Search Heuristics

Consider attempting to solve a difficult optimisation problem with random local search. Discuss advantages and disadvantages of increasing the neighbourhood from the set of all Hamming neighbours to all points with a Hamming distance of at most 2.

- + the increased neighbourhood can help to escape from local optima
 - one can still get stuck in some local optima
- on average good Hamming neighbours found much later
↪ slower on simple problems/in easy parts of the search space

Topic IX) Applying Randomised Search Heuristics

Consider attempting to solve a difficult optimisation problem with random local search. Discuss advantages and disadvantages of increasing the neighbourhood from the set of all Hamming neighbours to all points with a Hamming distance of at most 2.

- + the increased neighbourhood can help to escape from local optima
 - one can still get stuck in some local optima
- on average good Hamming neighbours found much later
 ↪ **slower** on simple problems/in easy parts of the search space

min 0 **median** 1 **average** 1.05 **max** 2

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value
- for $f(x) = 0$ it suffices to flip any bit

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value
- for $f(x) = 0$ it suffices to flip any bit

$$\rightsquigarrow s_0 \geq n \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq 1/e$$

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } \text{ONEMAX}(x) \text{ is even or } \text{ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value
- for $f(x) = 0$ it suffices to flip any bit
 $\rightsquigarrow s_0 \geq n \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq 1/e$
- for $f(x) = v$ it suffices to flip two of the $n - v$ 1-bits

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $E(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value
- for $f(x) = 0$ it suffices to flip any bit
 $\rightsquigarrow s_0 \geq n \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq 1/e$
- for $f(x) = v$ it suffices to flip two of the $n - v$ 1-bits
 \rightsquigarrow
 $s_v \geq \binom{n-v}{2} \cdot (1/n)^2 \cdot (1 - 1/n)^{n-2} \geq (n-v)(n-v-1)/(2en^2)$

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value
- for $f(x) = 0$ it suffices to flip any bit
 $\rightsquigarrow s_0 \geq n \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq 1/e$
- for $f(x) = v$ it suffices to flip two of the $n - v$ 1-bits
 \rightsquigarrow
 $s_v \geq \binom{n-v}{2} \cdot (1/n)^2 \cdot (1 - 1/n)^{n-2} \geq (n-v)(n-v-1)/(2en^2)$

Thus
$$\mathbb{E}(T_{(1+1) \text{ EA}, f}) \leq e + \sum_{i=1}^{n/2} \frac{2en^2}{(2i)(2i-1)}$$

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value
- for $f(x) = 0$ it suffices to flip any bit
- for $f(x) = v$ it suffices to flip two of the $n - v$ 1-bits

\rightsquigarrow

$$s_v \geq \binom{n-v}{2} \cdot (1/n)^2 \cdot (1 - 1/n)^{n-2} \geq (n-v)(n-v-1)/(2en^2)$$

Thus
$$\mathbb{E}(T_{(1+1) \text{ EA}, f}) \leq e + \sum_{i=1}^{n/2} \frac{2en^2}{(2i)(2i-1)} < e + 2en^2 \sum_{i=1}^{n/2} \frac{1}{i^2}$$

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value
- for $f(x) = 0$ it suffices to flip any bit
 $\rightsquigarrow s_0 \geq n \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq 1/e$
- for $f(x) = v$ it suffices to flip two of the $n - v$ 1-bits

\rightsquigarrow

$$s_v \geq \binom{n-v}{2} \cdot (1/n)^2 \cdot (1 - 1/n)^{n-2} \geq (n-v)(n-v-1)/(2en^2)$$

Thus
$$\mathbb{E}(T_{(1+1) \text{ EA}, f}) \leq e + \sum_{i=1}^{n/2} \frac{2en^2}{(2i)(2i-1)} < e + 2en^2 \sum_{i=1}^{n/2} \frac{1}{i^2} = O(n^2)$$

Topic X) Analysing Evolutionary Algorithms

Consider the (1+1) EA with mutation probability $p_m = 1/n$ on the function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ below. Prove an upper bound on the expected optimisation time $\mathbb{E}(T_{(1+1) \text{ EA}, f})$.

$$f(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \text{ is even or ONEMAX}(x) = n \\ 0 & \text{otherwise} \end{cases}$$

- use f -based partitions, one level per function value
- for $f(x) = 0$ it suffices to flip any bit
 $\rightsquigarrow s_0 \geq n \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq 1/e$
- for $f(x) = v$ it suffices to flip two of the $n - v$ 1-bits

\rightsquigarrow

$$s_v \geq \binom{n-v}{2} \cdot (1/n)^2 \cdot (1 - 1/n)^{n-2} \geq (n-v)(n-v-1)/(2en^2)$$

Thus $\mathbb{E}(T_{(1+1) \text{ EA}, f}) \leq e + \sum_{i=1}^{n/2} \frac{2en^2}{(2i)(2i-1)} < e + 2en^2 \sum_{i=1}^{n/2} \frac{1}{i^2} = O(n^2)$

min 0 median 1 average 0.71 max 2

Remember Results for LEADINGONES

Theorem

$$T_{(1+\lambda) \text{ EA, LEADINGONES}} = \Theta\left(n^2 + \lambda \cdot \frac{n}{\log(1+\lambda/n)}\right) \text{ for any } \lambda = n^{O(1)}$$

$$G_{(1+\lambda) \text{ EA, LEADINGONES}} = \Theta\left(\frac{n^2}{\lambda} + \frac{n}{\log(1+\lambda/n)}\right) \text{ for any } \lambda = n^{O(1)}$$

Remember Results for LEADINGONES

Theorem

$$T_{(1+\lambda)} \text{ EA, LEADINGONES} = \Theta\left(n^2 + \lambda \cdot \frac{n}{\log(1+\lambda/n)}\right) \text{ for any } \lambda = n^{O(1)}$$
$$G_{(1+\lambda)} \text{ EA, LEADINGONES} = \Theta\left(\frac{n^2}{\lambda} + \frac{n}{\log(1+\lambda/n)}\right) \text{ for any } \lambda = n^{O(1)}$$

Observations

- $G_{(1+\lambda)} \text{ EA, LEADINGONES} = \Omega(n/\log n)$ for all λ
- $T_{(1+\lambda)} \text{ EA, LEADINGONES} = \Omega(n^2)$ for all λ
- $T_{(1+\lambda)} \text{ EA, LEADINGONES} = O(n^2)$ for $\lambda = O(n)$
- $G_{(1+\lambda)} \text{ EA, LEADINGONES} = \Theta(n)$ $\lambda = \Theta(n)$
- $T_{(1+\lambda)} \text{ EA, LEADINGONES} = \Theta(n^2)$ $\lambda = \Theta(n)$
- with $\lambda = \Theta(n)$ **linear speed-up** without increased workload

Remember Results for LEADINGONES

Theorem

$$T_{(1+\lambda)} \text{ EA, LEADINGONES} = \Theta\left(n^2 + \lambda \cdot \frac{n}{\log(1+\lambda/n)}\right) \text{ for any } \lambda = n^{O(1)}$$

$$G_{(1+\lambda)} \text{ EA, LEADINGONES} = \Theta\left(\frac{n^2}{\lambda} + \frac{n}{\log(1+\lambda/n)}\right) \text{ for any } \lambda = n^{O(1)}$$

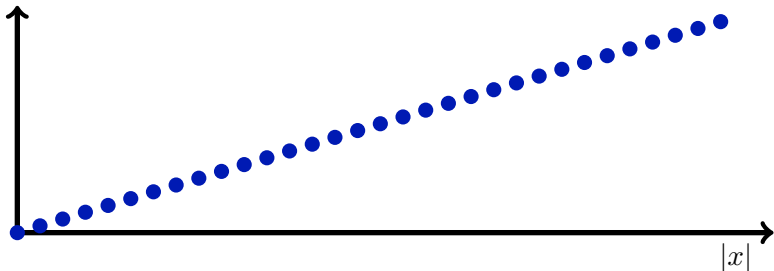
Observations

- $G_{(1+\lambda)} \text{ EA, LEADINGONES} = \Omega(n/\log n)$ for all λ
- $T_{(1+\lambda)} \text{ EA, LEADINGONES} = \Omega(n^2)$ for all λ
- $T_{(1+\lambda)} \text{ EA, LEADINGONES} = O(n^2)$ for $\lambda = O(n)$
- $G_{(1+\lambda)} \text{ EA, LEADINGONES} = \Theta(n)$ $\lambda = \Theta(n)$
- $T_{(1+\lambda)} \text{ EA, LEADINGONES} = \Theta(n^2)$ $\lambda = \Theta(n)$
- with $\lambda = \Theta(n)$ **linear speed-up** without increased workload

Observation $\text{Prob}(\text{single mutation is improving}) = \Theta(1/n)$ **always**
 $\hat{=}$ reciprocal of $\Theta(n)$

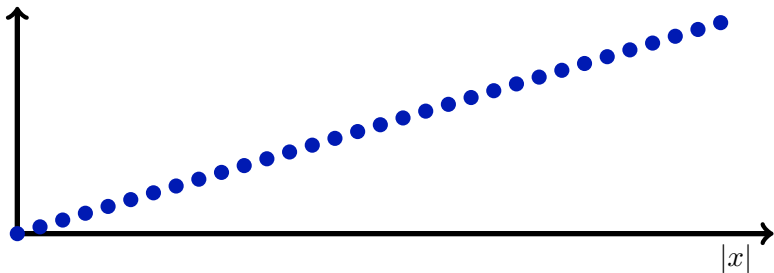
ONEMAX

Remember $\text{ONEMAX}(x) = \sum_{i=1}^n x[i]$



ONEMAX

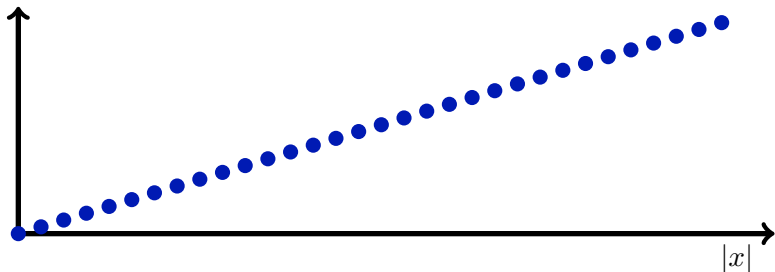
Remember $\text{ONEMAX}(x) = \sum_{i=1}^n x[i]$



Prob (single mutation is improving)?

ONEMAX

Remember $\text{ONEMAX}(x) = \sum_{i=1}^n x[i]$



Prob (single mutation is improving)?

Observation $\text{Prob}(\text{single mutation is improving}) = \Theta\left(\frac{n - \text{ONEMAX}(x)}{n}\right)$
 depends on search point, changes during search

$(1+\lambda)$ EA on ONEMAX

$(1+\lambda)$ EA on ONEMAX

Theorem

$$\left. \begin{aligned} T_{(1+\lambda) \text{ EA, ONEMAX}} &= O(n \log n) \\ G_{(1+\lambda) \text{ EA, ONEMAX}} &= O\left(\frac{n \log n}{\lambda}\right) \end{aligned} \right\} \text{for } \lambda = O\left(\frac{\log(n) \log \log n}{\log \log \log n}\right)$$

$(1+\lambda)$ EA on ONEMAX

Theorem

$$\left. \begin{aligned} T_{(1+\lambda) \text{ EA, ONEMAX}} &= O(n \log n) \\ G_{(1+\lambda) \text{ EA, ONEMAX}} &= O\left(\frac{n \log n}{\lambda}\right) \end{aligned} \right\} \text{for } \lambda = O\left(\frac{\log(n) \log \log n}{\log \log \log n}\right)$$

$$\left. \begin{aligned} T_{(1+\lambda) \text{ EA, ONEMAX}} &= \omega(n \log n) \\ G_{(1+\lambda) \text{ EA, ONEMAX}} &= \omega\left(\frac{n \log n}{\lambda}\right) \end{aligned} \right\} \text{for } \lambda = \omega\left(\frac{\log(n) \log \log n}{\log \log \log n}\right)$$

$(1+\lambda)$ EA on ONEMAX

Theorem

$$\left. \begin{aligned} T_{(1+\lambda) \text{ EA, ONEMAX}} &= O(n \log n) \\ G_{(1+\lambda) \text{ EA, ONEMAX}} &= O\left(\frac{n \log n}{\lambda}\right) \end{aligned} \right\} \text{for } \lambda = O\left(\frac{\log(n) \log \log n}{\log \log \log n}\right)$$

$$\left. \begin{aligned} T_{(1+\lambda) \text{ EA, ONEMAX}} &= \omega(n \log n) \\ G_{(1+\lambda) \text{ EA, ONEMAX}} &= \omega\left(\frac{n \log n}{\lambda}\right) \end{aligned} \right\} \text{for } \lambda = \omega\left(\frac{\log(n) \log \log n}{\log \log \log n}\right)$$

Observation again **linear speed-up** without increased workload if λ is sufficiently large but not too large

$(1+\lambda)$ EA on ONEMAX

Theorem

$$\left. \begin{aligned} T_{(1+\lambda) \text{ EA, ONEMAX}} &= O(n \log n) \\ G_{(1+\lambda) \text{ EA, ONEMAX}} &= O\left(\frac{n \log n}{\lambda}\right) \end{aligned} \right\} \text{for } \lambda = O\left(\frac{\log(n) \log \log n}{\log \log \log n}\right)$$

$$\left. \begin{aligned} T_{(1+\lambda) \text{ EA, ONEMAX}} &= \omega(n \log n) \\ G_{(1+\lambda) \text{ EA, ONEMAX}} &= \omega\left(\frac{n \log n}{\lambda}\right) \end{aligned} \right\} \text{for } \lambda = \omega\left(\frac{\log(n) \log \log n}{\log \log \log n}\right)$$

Observation again **linear speed-up** without increased workload if λ is sufficiently large but not too large

$\hat{=}$ with appropriate λ
parallel $(1+\lambda)$ EA outperforms $(1+1)$ EA
even for very simple fitness function

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA
- $T_{(1+\lambda) \text{ EA}, f} \hat{=} \text{number of function evaluations}$
 $\hat{=} \text{computational effort}$

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA
- $T_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of function evaluations
 $\hat{=}$ computational effort
- $G_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of generations
 $\hat{=}$ parallel computation time using $\geq \lambda$ processors

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA
- $T_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of function evaluations
 $\hat{=}$ computational effort
- $G_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of generations
 $\hat{=}$ parallel computation time using $\geq \lambda$ processors
- **result** on LEADINGONES: $G_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ decreases
without increasing $T_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ for $\lambda = O(n)$

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA
- $T_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of function evaluations
 $\hat{=}$ computational effort
- $G_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of generations
 $\hat{=}$ parallel computation time using $\geq \lambda$ processors
- **result** on LEADINGONES: $G_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ for $\lambda = O(n)$
 $\hat{=}$ reciprocal of success probability

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA
- $T_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of function evaluations
 $\hat{=}$ computational effort
- $G_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of generations
 $\hat{=}$ parallel computation time using $\geq \lambda$ processors
- **result** on LEADINGONES: $G_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ for $\lambda = O(n)$
 $\hat{=}$ reciprocal of success probability
- **result** on ONEMAX: $G_{(1+\lambda) \text{ EA}, \text{ONEMAX}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{ONEMAX}}$ for $\lambda = O(\log(n) \log(\log(n)) / \log(\log(\log(n))))$

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA
- $T_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of function evaluations
 $\hat{=}$ computational effort
- $G_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of generations
 $\hat{=}$ parallel computation time using $\geq \lambda$ processors
- **result** on LEADINGONES: $G_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ for $\lambda = O(n)$
 $\hat{=}$ reciprocal of success probability
- **result** on ONEMAX: $G_{(1+\lambda) \text{ EA}, \text{ONEMAX}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{ONEMAX}}$ for $\lambda = O(\log(n) \log(\log(n)) / \log(\log(\log(n))))$
no 'fixed' success probability on ONEMAX

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA
- $T_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of function evaluations
 $\hat{=}$ computational effort
- $G_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of generations
 $\hat{=}$ parallel computation time using $\geq \lambda$ processors
- **result** on LEADINGONES: $G_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ for $\lambda = O(n)$
 $\hat{=}$ reciprocal of success probability
- **result** on ONEMAX: $G_{(1+\lambda) \text{ EA}, \text{ONEMAX}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{ONEMAX}}$ for $\lambda = O(\log(n) \log(\log(n)) / \log(\log(\log(n))))$
no 'fixed' success probability on ONEMAX
- **seen** parallelisation simple and effective $\hat{=}$ linear speed-up

Setting the Offspring Population Size

Remember

- $(1+\lambda)$ EA
- $T_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of function evaluations
 $\hat{=}$ computational effort
- $G_{(1+\lambda) \text{ EA}, f} \hat{=}$ number of generations
 $\hat{=}$ parallel computation time using $\geq \lambda$ processors
- **result** on LEADINGONES: $G_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{LEADINGONES}}$ for $\lambda = O(n)$
 $\hat{=}$ reciprocal of success probability
- **result** on ONEMAX: $G_{(1+\lambda) \text{ EA}, \text{ONEMAX}}$ decreases without increasing $T_{(1+\lambda) \text{ EA}, \text{ONEMAX}}$ for $\lambda = O(\log(n) \log(\log(n)) / \log(\log(\log(n))))$
no 'fixed' success probability on ONEMAX
- **seen** parallelisation simple and effective $\hat{=}$ linear speed-up
- **Can parallelisation achieve even more?**

$(1+\lambda)$ EA and $(1+1)$ EA

$(1+\lambda)$ EA and $(1+1)$ EA

Observation $(1+\lambda)$ EA with $\lambda > 1$
is **less fast** in adopting and exploiting improvements
in comparison to $(1+1)$ EA

$(1+\lambda)$ EA and $(1+1)$ EA

Observation $(1+\lambda)$ EA with $\lambda > 1$
is **less fast** in adopting and exploiting improvements
in comparison to $(1+1)$ EA

Can this wastefulness be beneficial even without parallelisation?

$(1+\lambda)$ EA and $(1+1)$ EA

Observation $(1+\lambda)$ EA with $\lambda > 1$
is **less fast** in adopting and exploiting improvements
in comparison to $(1+1)$ EA

Can this wastefulness be beneficial even without parallelisation?

Idea may be beneficial if some improvements are better than others

$(1+\lambda)$ EA and $(1+1)$ EA

Observation $(1+\lambda)$ EA with $\lambda > 1$
is **less fast** in adopting and exploiting improvements
in comparison to $(1+1)$ EA

Can this wastefulness be beneficial even without parallelisation?

Idea may be beneficial if some improvements are better than others
necessary function values give correct hints



An Example Function

An Example Function

Definition with $k := \lfloor \sqrt{n} \rfloor$

$$f(x) = \begin{cases} (i+3)n + |x| & \text{if } (x = 0^{n-i}1^i \text{ with } i \in \{0, 1, \dots, n\}) \text{ or} \\ & x = y0^{n-i-k}1^i \text{ with } y \in \{0, 1\}^k \text{ and} \\ & i \in \{k, 2k, \dots, (k-2)k\} \\ 0 & \text{otherwise} \end{cases}$$

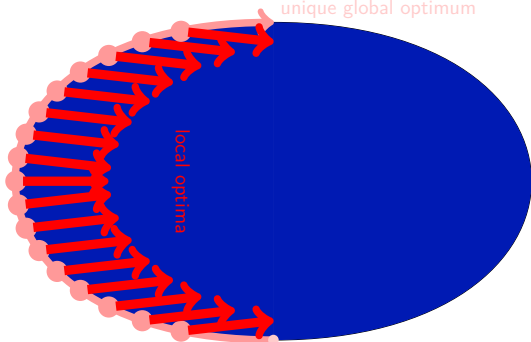
suitable for **deterministic** start in 0^n

An Example Function

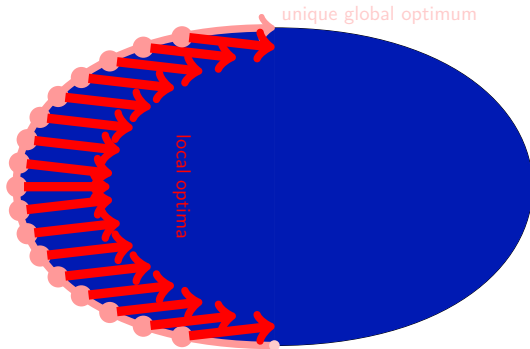
Definition with $k := \lfloor \sqrt{n} \rfloor$

$$f(x) = \begin{cases} |x| & \text{if } (x = 0^{n-i}1^i \text{ with } i \in \{0, 1, \dots, n\}) \text{ or} \\ (i + 3)n + |x| & x = y0^{n-i-k}1^i \text{ with } y \in \{0, 1\}^k \text{ and} \\ & i \in \{k, 2k, \dots, (k-2)k\} \\ 0 & \text{otherwise} \end{cases}$$

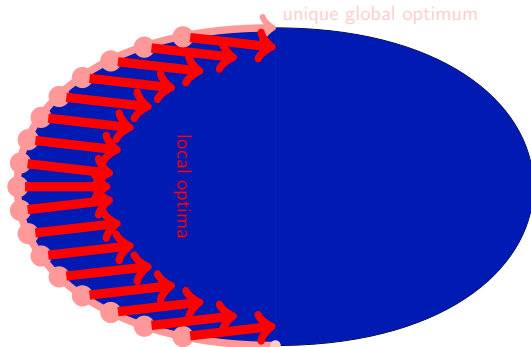
suitable for **deterministic** start in 0^n
unique global optimum



About the Example Function



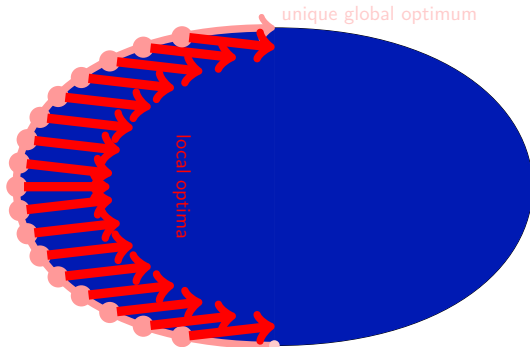
About the Example Function



Facts

- can be imbedded into a complete function g

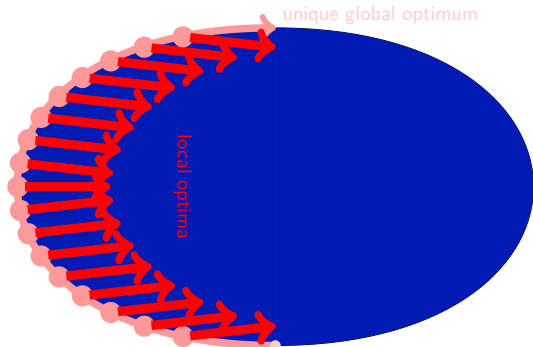
About the Example Function



Facts

- can be imbedded into a complete function g
- $(1+1)$ EA on g **very inefficient** (exponential time with probability very close to 1)

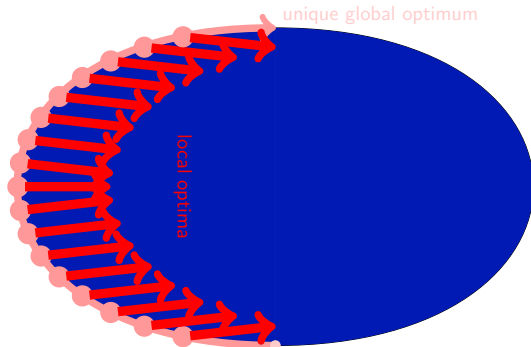
About the Example Function



Facts

- can be imbedded into a complete function g
- $(1+1)$ EA on g **very inefficient** (exponential time with probability very close to 1)
- $(1+\lambda)$ EA with $\lambda \geq n \log n$ **efficient**, expected time $O(n\lambda)$

About the Example Function

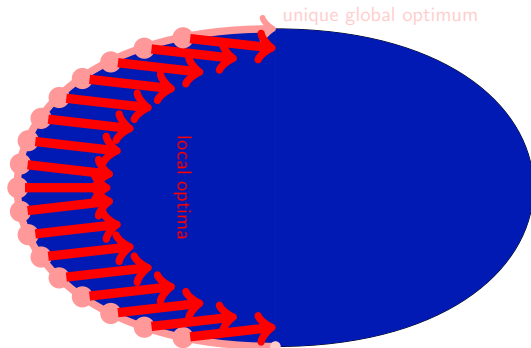


Facts

- can be imbedded into a complete function g
- $(1+1)$ EA on g **very inefficient** (exponential time with probability very close to 1)
- $(1+\lambda)$ EA with $\lambda \geq n \log n$ **efficient**, expected time $O(n\lambda)$

Remark roles of local and global optima can be **reversed**

About the Example Function



Facts

- can be imbedded into a complete function g
- $(1+1)$ EA on g **very inefficient** (exponential time with probability very close to 1)
- $(1+\lambda)$ EA with $\lambda \geq n \log n$ **efficient**, expected time $O(n\lambda)$

Remark roles of local and global optima can be **reversed**
 \Rightarrow reverse performance characteristics

The (1+1) EA

Remember

The (1+1) EA

Remember

(1+1) EA

1. $t := 1$; Choose $x_t \in \{0, 1\}^n$ uniformly at random.

The (1+1) EA

Remember

(1+1) EA

1. $t := 1$; Choose $x_t \in \{0, 1\}^n$ uniformly at random.
2. Repeat
3. $y := \text{standard bit mutation}(x_t)$ with mutation prob. p_m

The (1+1) EA

Remember

(1+1) EA

1. $t := 1$; Choose $x_t \in \{0, 1\}^n$ uniformly at random.
2. Repeat
3. $y := \text{standard bit mutation}(x_t)$ with mutation prob. p_m
4. $t := t + 1$; If $f(y) \geq f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.

The (1+1) EA

Remember

(1+1) EA

1. $t := 1$; Choose $x_t \in \{0, 1\}^n$ uniformly at random.
2. Repeat
3. $y := \text{standard bit mutation}(x_t)$ with mutation prob. p_m
4. $t := t + 1$; If $f(y) \geq f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t .

The (1+1) EA

Remember

(1+1) EA

1. $t := 1$; Choose $x_t \in \{0, 1\}^n$ uniformly at random.
2. Repeat
3. $y := \text{standard bit mutation}(x_t)$ with mutation prob. p_m
4. $t := t + 1$; If $f(y) \geq f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t .

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

The (1+1) EA

Remember

(1+1) EA

1. $t := 1$; Choose $x_t \in \{0, 1\}^n$ uniformly at random.
2. Repeat
3. $y :=$ standard bit mutation(x_t) with mutation prob. p_m
4. $t := t + 1$; If $f(y) \geq f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t .

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

Remember $E(\# \text{flipping bits}) = p_m \cdot n$

The (1+1) EA

Remember

(1+1) EA

1. $t := 1$; Choose $x_t \in \{0, 1\}^n$ uniformly at random.
2. Repeat
3. $y :=$ standard bit mutation(x_t) with mutation prob. p_m
4. $t := t + 1$; If $f(y) \geq f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t .

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

Remember $E(\# \text{flipping bits}) = p_m \cdot n = 1$ with $p_m = 1/n$

The (1+1) EA

Remember

(1+1) EA

1. $t := 1$; Choose $x_t \in \{0, 1\}^n$ uniformly at random.
2. Repeat
3. $y :=$ standard bit mutation(x_t) with mutation prob. p_m
4. $t := t + 1$; If $f(y) \geq f(x_{t-1})$ then $x_t := y$ else $x_t := x_{t-1}$.
5. Until 'decide to stop'
6. Output x_t .

Parameter mutation probability $p_m \in (0, 1/2]$, usually $p_m = 1/n$

Remember $E(\# \text{flipping bits}) = p_m \cdot n = 1$ with $p_m = 1/n$

Why should $p_m = 1/n$ be a good universal choice?

About a Universally Optimal Mutation Probability

Why should $p_m = 1/n$ be a good universal choice?

About a Universally Optimal Mutation Probability

Why should $p_m = 1/n$ be a good universal choice?

Remember NFL 'Averaged over all functions all RSH perform equal.'
 $\hat{=}$ There is no universally best RSH.

About a Universally Optimal Mutation Probability

Why should $p_m = 1/n$ be a good universal choice?

Remember NFL 'Averaged over all functions all RSH perform equal.'

- ≡ There is no universally best RSH.
- ≡ There is no universally best p_m .

About a Universally Optimal Mutation Probability

Why should $p_m = 1/n$ be a good universal choice?

Remember NFL 'Averaged over all functions all RSH perform equal.'
 $\hat{=}$ There is no universally best RSH.
 $\hat{=}$ There is no universally best p_m .

Careful! NFL consider non-repeating algorithms
not actual optimisation times.

About a Universally Optimal Mutation Probability

Why should $p_m = 1/n$ be a good universal choice?

Remember NFL 'Averaged over all functions all RSH perform equal.'
 $\hat{=}$ There is no universally best RSH.
 $\hat{=}$ There is no universally best p_m .

Careful! NFL consider non-repeating algorithms
not actual optimisation times.

Observation for 'normal' RSH not too important

About a Universally Optimal Mutation Probability

Why should $p_m = 1/n$ be a good universal choice?

Remember NFL 'Averaged over all functions all RSH perform equal.'
≡ There is no universally best RSH.
≡ There is no universally best p_m .

Careful! NFL consider non-repeating algorithms
not actual optimisation times.

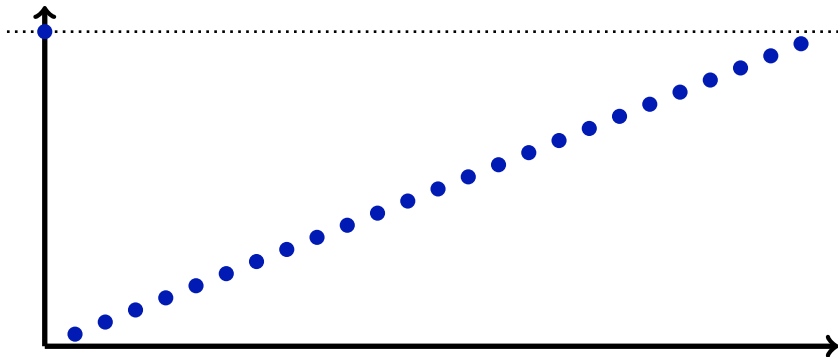
Observation for 'normal' RSH not too important
still: Can we find a simple example to prove the point?

A Simple Example

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

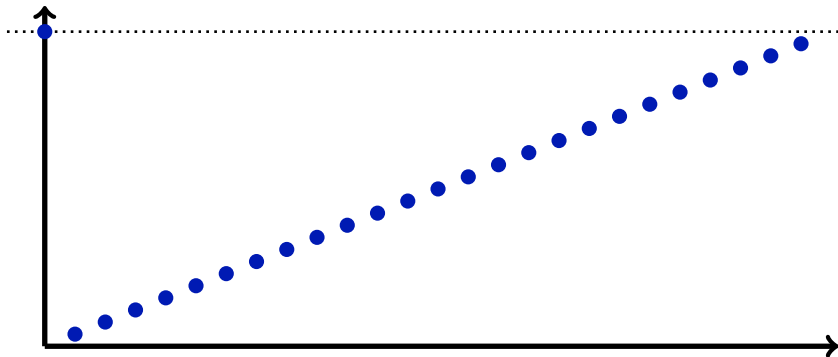
A Simple Example

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$



A Simple Example

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$



Theorem

$$\mathbb{E} \left(T_{(1+1) \text{ EA}, f_1} \right) = \Theta(n^n) \text{ with } p_m = 1/n$$

$$\mathbb{E} \left(T_{(1+1) \text{ EA}, f_1} \right) = \Theta(2^n) \text{ with } p_m = 1/2$$

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

- $\text{Prob}(\text{ONEMAX}(x_0) \geq n/3) = 1 - 2^{-\Omega(n)}$

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

- $\text{Prob}(\text{ONEMAX}(x_0) \geq n/3) = 1 - 2^{-\Omega(n)}$
- given that only direct mutation can lead to optimum

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

- $\text{Prob}(\text{ONEMAX}(x_0) \geq n/3) = 1 - 2^{-\Omega(n)}$
- given that only direct mutation can lead to optimum
- $\text{Prob}(\text{do this before reaching } 1^n) = O((n \log n)/n^{n/3})$

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

- $\text{Prob}(\text{ONEMAX}(x_0) \geq n/3) = 1 - 2^{-\Omega(n)}$
- given that only direct mutation can lead to optimum
- $\text{Prob}(\text{do this before reaching } 1^n) = O\left((n \log n)/n^{n/3}\right)$
- $E(\text{time } 1^n \rightsquigarrow 0^n) = n^n$

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

- $\text{Prob}(\text{ONEMAX}(x_0) \geq n/3) = 1 - 2^{-\Omega(n)}$
- given that only direct mutation can lead to optimum
- $\text{Prob}(\text{do this before reaching } 1^n) = O((n \log n)/n^{n/3})$
- $E(\text{time } 1^n \rightsquigarrow 0^n) = n^n$
- $\Rightarrow E(T_{(1+1) \text{ EA}, f_1}) = \Theta(n^n)$ with $p_m = 1/n$ ✓

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

- $\text{Prob}(\text{ONEMAX}(x_0) \geq n/3) = 1 - 2^{-\Omega(n)}$
- given that only direct mutation can lead to optimum
- $\text{Prob}(\text{do this before reaching } 1^n) = O((n \log n)/n^{n/3})$
- $E(\text{time } 1^n \rightsquigarrow 0^n) = n^n$
- $\Rightarrow E(T_{(1+1) \text{ EA}, f_1}) = \Theta(n^n)$ with $p_m = 1/n$ ✓

(1+1) EA with $p_m = 1/2$

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

- $\text{Prob}(\text{ONEMAX}(x_0) \geq n/3) = 1 - 2^{-\Omega(n)}$
- given that only direct mutation can lead to optimum
- $\text{Prob}(\text{do this before reaching } 1^n) = O((n \log n)/n^{n/3})$
- $E(\text{time } 1^n \rightsquigarrow 0^n) = n^n$
- $\Rightarrow E(T_{(1+1) \text{ EA}, f_1}) = \Theta(n^n)$ with $p_m = 1/n$ ✓

(1+1) EA with $p_m = 1/2$

- $\forall x: \text{Prob}(x \rightsquigarrow 0^n) = 2^{-n}$

Analysis for the Example Function

$$f_1(x) = \begin{cases} \text{ONEMAX}(x) & \text{if } x \neq 0^n \\ n + 1 & \text{otherwise} \end{cases}$$

(1+1) EA with $p_m = 1/n$

- $\text{Prob}(\text{ONEMAX}(x_0) \geq n/3) = 1 - 2^{-\Omega(n)}$
- given that only direct mutation can lead to optimum
- $\text{Prob}(\text{do this before reaching } 1^n) = O((n \log n)/n^{n/3})$
- $E(\text{time } 1^n \rightsquigarrow 0^n) = n^n$
- $\Rightarrow E(T_{(1+1) \text{ EA}, f_1}) = \Theta(n^n)$ with $p_m = 1/n$ ✓

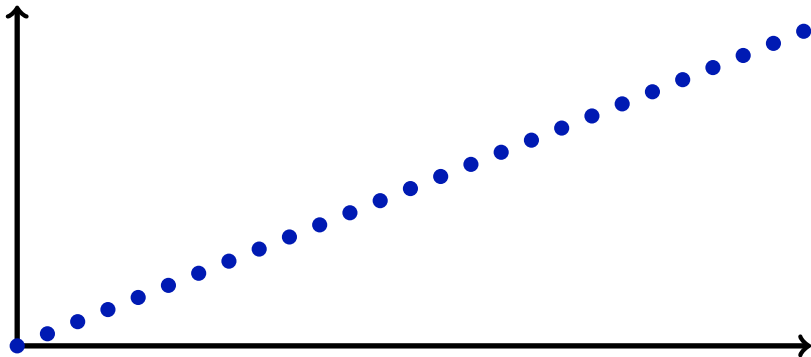
(1+1) EA with $p_m = 1/2$

- $\forall x: \text{Prob}(x \rightsquigarrow 0^n) = 2^{-n}$
- $\Rightarrow E(T_{(1+1) \text{ EA}, f_1}) = \Theta(2^n)$ with $p_m = 1/2$ ✓



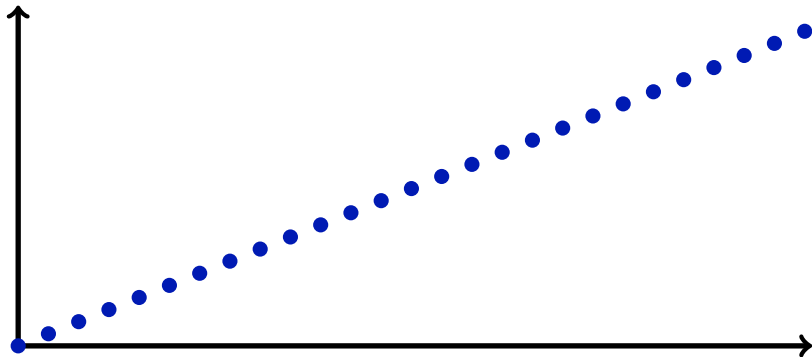
ONEMAX

$$\text{ONEMAX}(x) = \sum_{i=1}^n x[i]$$



ONEMAX

$$\text{ONEMAX}(x) = \sum_{i=1}^n x[i]$$



Goal derive **simple** upper and lower bounds
on $E\left(T_{(1+1)\text{EA}, \text{ONEMAX}}\right)$ depending on p_m



Summary & Take Home Message

Things to remember



Summary & Take Home Message

Things to remember

- $(1+\lambda)$ EA on ONEMAX

Summary & Take Home Message

Things to remember

- $(1+\lambda)$ EA on ONEMAX
- super-linear speed-up by parallelisation of EAs



Summary & Take Home Message

Things to remember

- $(1+\lambda)$ EA on ONEMAX
- super-linear speed-up by parallelisation of EAs
- ideas and example functions

Summary & Take Home Message

Things to remember

- $(1+\lambda)$ EA on ONEMAX
- super-linear speed-up by parallelisation of EAs
- ideas and example functions
- mutation probability crucial

Summary & Take Home Message

Things to remember

- $(1+\lambda)$ EA on ONEMAX
- super-linear speed-up by parallelisation of EAs
- ideas and example functions
- mutation probability crucial

Take Home Message

Summary & Take Home Message

Things to remember

- $(1+\lambda)$ EA on ONEMAX
- super-linear speed-up by parallelisation of EAs
- ideas and example functions
- mutation probability crucial

Take Home Message

- Linear speed-ups only work up to some cut off point that depends on the objective function.

Summary & Take Home Message

Things to remember

- $(1+\lambda)$ EA on ONEMAX
- super-linear speed-up by parallelisation of EAs
- ideas and example functions
- mutation probability crucial

Take Home Message

- Linear speed-ups only work up to some cut off point that depends on the objective function.
- More extreme positive and negative effects can occur.