Computationally Hard Problems

Analysis of Randomized Search Heuristics – Foundations

Carsten Witt

Institut for Matematik og Computer Science
Danmarks Tekniske Universitet

Fall 2020

Two Very Simple Search Heuristics

Search space $\{0,1\}^n$ 

, "population" size 1, "offspring population" size 1, selection: "take the better"

Aim: maximize $f\colon \{0,1\}^n \to \mathbb{R}$

Randomized Local Search (RLS)(1+1) Evolutionary Algorithm ((1+1) EA)

1. $t := 0$. Choose $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ uniformly at random.
2. $y := x$
3. Choose one bit in y uniformly and flip it.Independently for each bit in y: flip it with probability $\frac{1}{n}$ (mutation).
4. If $f(y) \geq f(x)$ Then $x := y$ (selection).
5. $t := t + 1$. Continue at line 2.

Extremely simple (good for analysis) and surprisingly efficient.

Focus: smallest t ("runtime") to reach optimal solution

Framework for Analysis

Given

- randomized search heuristic A
- fitness function f

study no. T of f-evaluations (black-box) until A finds optimum.

T is random variable

- ideally study whole distribution $\Pr(T \leq t)$
- less ambitious: expectation $E(T)$
- even less ambitious (but feasible): bounds on $E(T)$

For (1+1) EA: T equals no. of iterations until optimum found.
Call this runtime/optimization time of (1+1) EA on f.

What problems/fitness functions to study?

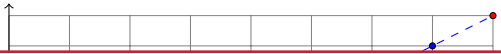
Example Problems/Toy Problems

Most famous example problem: $\textsc{OneMax}(x_1, \ldots, x_n) = x_1 + \cdots + x_n$
(the heuristic does not know it is working on it)

Why should we care about such example problems?

- support analysis, help to develop analytical tools
- are easy to understand, are clearly structured
- make important aspects visible
- act as counterexamples
- help to discover general properties
- are important tools for further analysis $\rightarrow \mathcal{NP}$-complete problems
- positive results on easy examples make us trust the algorithm

$\textsc{OneMax}(x)$



A First Attempt

Theorem (General Upper Bound)

The expected optimization time of the (1+1) EA on an arbitrary function is $O(n^n)$.

Proof:

- Wait for current bitstring to mutate to optimum.
- At most n bits need to flip: probability at least $1/n^n$.
- Waiting time argument (Lemma A2 in lecture notes). □

General upper bound for RLS does not exist (∞).

Upper Bound on OneMax

Theorem

The expected optimization time of RLS and the (1+1) EA on OneMax *is* $O(n \log n)$.

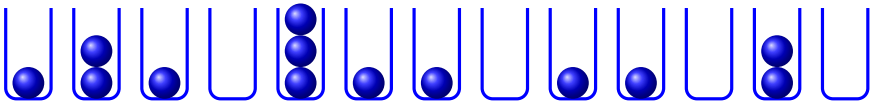
Proof for RLS(1+1) EA:

- Consider $\phi \in \{0, \ldots, n\}$: current no. one-bits
- Divide run: phase i starts when $\phi = i$ and ends when ϕ increases
- Sufficient for increase: flip a zero-bit, do not flip rest
- $\Pr(\text{increase } \phi \mid \phi = i) \geq \binom{n-i}{1} \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{n-i}{n} \frac{n-i}{en}$
- $E(\text{length of phase } i) \leq \frac{n}{n-i} \frac{en}{n-i}$
- Expected duration of all phases $\leq \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{i=1}^{n} \frac{1}{i} = O(n \log n)$ since

$$\sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1. \leq \sum_{i=0}^{n-1} \frac{en}{n-i} = en \sum_{i=1}^{n} \frac{1}{i} = O(n \log n) \text{ since } \sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1.$$

Coupon Collector's Problem

Previous reasoning and proof strategy is well known in a combinatorial game.



Scenario: You have n bins. At each time step, you choose a bin uniformly at random and throw a ball in it.

Question: How long does it take until every bin has at least one ball?

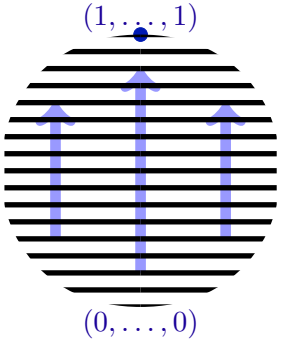
Name: Coupon Collector's Problem (each bin can be considered a "coupon"): there are n types of coupons and in each round you get one coupon uniformly at random with replacement. How long does it take until you have at least coupon of very type?

Known result: Expected time is at least $n \ln n$ and at most $n \ln n + n$.

Note: Arguments for this are in the previous proof (RLS part).

Fitness-Based Partitions

Aim: more general upper-bound method



Definition

$L_0, L_1, \ldots, L_k \subseteq \{0,1\}^n$ is fitness-based partition iff

1. $\forall i \in \{0,1,\ldots,k\}\colon L_i \neq \emptyset$
2. $\forall i \neq j \in \{0,1,\ldots,k\}\colon L_i \cap L_j = \emptyset$
3. $\bigcup_{i=0}^{k} L_i = \{0,1\}^n$
4. $\forall i < j \in \{0,1,\ldots,k\}\colon \forall x \in L_i\colon\ \forall y \in L_j\colon f(x) < f(y)$
5. $L_k = \{x \in \{0,1\}^n \mid f(x) = \max\{f(x') \mid x' \in \{0,1\}^n\}\}$

Example:

$L_i = \{x \mid \textsc{OneMax}(x) = i\}$

Bounds with Fitness-Based Partitions

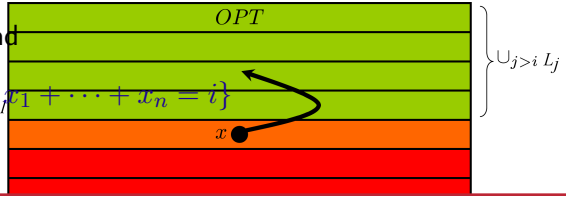
Theorem

Consider (1+1) EA/RLS on $f\colon \{0,1\}^n \to \mathbb{R}$.
Consider f*-based partition* $L_0, L_1, \ldots, L_k$.
Let for all $i \in \{0,1,\ldots,k-1\}$ $s_i := \min_{x \in L_i} \Pr(\textit{mutate } x \textit{ into some } y \in L_{i+1} \cup \cdots \cup L_k)$

Then expected optimization time at most $\leq \sum_{i=0}^{k-1} \frac{1}{s_i}$.

Note: for OneMax we had

- $L_i := \{(x_1,\ldots,x_n) \mid x_1 + \cdots + x_n = i\}$
- $s_i \geq \frac{n-i}{en}$.



Fitness-Based Partitions: Example

Consider $\textsc{BinVal}(x_1, \ldots, x_n) := \sum_{i=1}^{n} 2^{n-i} \cdot x_i$,
another simple function (a so-called separable/linear function).

Theorem

The expected optimization time of the (1+1) EA and RLS on $\textsc{BinVal}$ *is* $O(n^2)$.

Proof idea:

- $L_i := \{x \mid 2^{n-1} + 2^{n-2} + \cdots + 2^{n-i} \leq \textsc{BinVal}(x) < 2^{n-1} + 2^{n-2} + \cdots + 2^{n-i} + 2^{n-i-1}\}$
  (the i most significant bits are 1, bit $i+1$ is 0).
- $s_i \geq \frac{1}{en}$ for all $i < n$
- Optimization time $\leq n \cdot en = O(n^2)$.

Additional O-Notation: "big-Omega"

We know: $f(n) = O(g(n))$ if $f(n) \leq c \cdot g(n)$ for some constant $c > 0$ and all $n \geq 1$ ($f(n)$ and $g(n)$ have to be non-negative functions).

The other way round: $f(n) = \Omega(g(n))$ if $f(n) \geq c \cdot g(n)$ for some constant $c > 0$ and all $n \geq 1$.

Example: $n^2 = \Omega(n \log n)$.

Attention: $\Omega(.)$ will also be used in exponentially small upper bounds: $2^{-\Omega(n)}$ means $\leq 2^{-cn}$ for some $c > 0$.

Deviation/Concentration Inequalities

- Markov's inequality: If X random var. with non-negative outcomes then $\Pr(X \geq t \cdot E(X)) \leq \frac{1}{t}$ for any t

  Example: algorithm with expected runtime $O(n^2)$ finishes in time $O(n^3)$ with probability $\geq 1 - \frac{1}{n}$.

- Chernoff's bound: $X_1, \ldots, X_n$ independent $\{0,1\}$-random vars with $p_i := \Pr(X_i = 1)$, $1 \leq i \leq n$. For $X := \sum_{i=1}^{n} X_i$ and $\mu := E(X) = \sum_{i=1}^{n} p_i$:
  1. $\Pr(X < (1-\delta)\mu) < e^{-\mu\delta^2/2}$ for $0 < \delta < 1$ (where $e = 2.718\ldots$),
  2. $\Pr(X > (1+\delta)\mu) < e^{-\mu\delta^2/3}$ for $0 < \delta < 1$.

  Toy example: number of heads among 100001000 coin tosses is in $[4000, 6000]$ with prob. $\geq 0.99999999999999999999999999992$ $[400, 600]$ with prob. ≥ 0.997.

  "Serious application": number of 1-bits in a uniform bit string of length n is in $[\frac{1}{3}n, \frac{2}{3}n]$ with prob. $1 - 2^{-\Omega(n)}$.

General Lower Bound (1/5)

Theorem

Let $f\colon \{0,1\}^n \to \mathbb{R}$ be a function with unique optimal bit string. The expected optimization time of the (1+1) EA and RLS on f is $\Omega(n \log n)$.

Proof for RLS:

- Initial bit string has at least $\frac{n}{3}$ bits chosen differently from unique optimum due to
  - uniform choice
  - Chernoff bounds
  - with probability $1 - 2^{-\Omega(n)}$
- Assuming this $\to$ play a Coupon Collector on at least $n/3$ bits.
- Expected time $\Omega((n/3)\log(n/3)) = \Omega(n \log n)$

General Lower Bound (2/5)

Proof for (1+1) EA: (Why can't we use coupon collector arguments?)

- Assume again $\geq n/3$ wrong bits after initialization.
- T: optimization time under this assumption
- Observe $\forall t \in \mathbb{N}$:
  $E(T) \geq \sum_{i=t}^{\infty} i \cdot \Pr(T = i) \geq \sum_{i=t}^{\infty} t \cdot \Pr(T = i) = t \cdot \Pr(T \geq t)$
- Later: make a clever choice for t.
- $\Pr(T \geq t)$
  $\geq \Pr(\exists$ after t steps bit from the $\frac{n}{3}$ wrong ones that never flipped$)$

General Lower Bound (3/5)

$\Pr(\text{1 specific bit flips}) = \frac{1}{n}$

$\Pr(\text{1 specific bit does not flip}) = 1 - \frac{1}{n}$

$\Pr(\text{1 specific bit does not flip in } t \text{ steps}) = \left(1 - \frac{1}{n}\right)^t$

$\Pr(\text{1 specific bit flips at least once in } t \text{ steps}) = 1 - \left(1 - \frac{1}{n}\right)^t$

$\Pr(n/3 \text{ specific bits all flip at least once in } t \text{ steps})$
$= \left(1 - \left(1 - \frac{1}{n}\right)^t\right)^{n/3}$

$\Pr(\exists \text{ bit out of } n/3 \text{ specific bits that never flips in } t \text{ steps})$
$= 1 - \left(1 - \left(1 - \frac{1}{n}\right)^t\right)^{n/3}$

General Lower Bound (4/5)

$\Pr(\exists \text{ bit out of } n/3 \text{ specific bits that never flips in } t \text{ steps}) = 1 - \left(1 - \left(1 - \frac{1}{n}\right)^t\right)^{n/3}$

Our "magic" choice: $t := (n-1)\ln n$

$$\begin{aligned}
\implies \Pr\cdots &= 1 - \left(1 - \left(1 - \frac{1}{n}\right)^{(n-1)\ln n}\right)^{n/3} \\
\text{(using } (1-1/n)^{n-1} \geq 1/e\text{)} \quad &\geq 1 - \left(1 - \left(\frac{1}{e}\right)^{\ln n}\right)^{n/3} \\
&= 1 - \left(1 - \frac{1}{n}\right)^{n/3} \\
\text{(using } (1-1/n)^{n} \leq 1/e\text{)} \quad &\geq 1 - \left(\frac{1}{e}\right)^{1/3} = 1 - e^{-1/3} > 0.28
\end{aligned}$$

General Lower Bound (5/5)

Putting things together

$$\Pr(T \geq (n-1)\ln n) \;\geq\; 1 - e^{-1/3}$$

$$E(T) \geq (n-1)\ln n \cdot (1 - e^{-1/3}) = \Omega(n \log n)$$

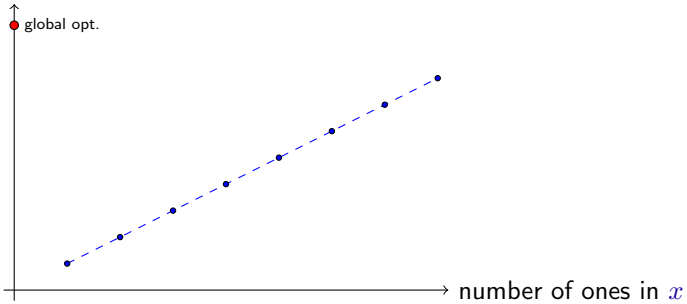
Our condition on $\geq n/3$ initially wrong bits holds with prob. $1 - 2^{-\Omega(n)}$.

Altogether:
Expected optimization time $\Omega(n \log n)$. $\square$

A Worst-Case Example

General upper bound $O(n^n)$ for (1+1) EA is tight.

Consider $\textsc{Trap}(x_1, \ldots, x_n) := \begin{cases} x_1 + \cdots + x_n & \text{if } x_1 + \cdots + x_n \geq 1, \\ n+1 & \text{otherwise.} \end{cases}$



A Worst-Case Example: Proof (1/2)

Theorem

The optimization time of the (1+1) EA on TRAP *is* $\geq n^{n/2}$ *with probability* $1 - 2^{-\Omega(n)}$. *For RLS, it is infinite with probability* $1 - 2^{-\Omega(n)}$.

Proof outline: consider the following typical run

1. Start with $\geq n/3$ ones.
2. Given that $n/3$ bits never flip together: "see" only ONEMAX, reach all-ones string.
3. Final improvement $(1, \ldots, 1) \to (0, \ldots, 0)$ takes $n^{\Omega(n)}$ steps (even ∞ for RLS).

A Worst-Case Example: Proof (2/2)

Filling in details: bound prob. of unwanted events by $2^{-\Omega(n)}$

1. Chernoff bounds yield probability $1 - 2^{-\Omega(n)}$.
2. Optimization time on $\textsc{OneMax}$ is $\leq 2^n n \log n$ with prob. $1 - 2^{-\Omega(n)}$ (Markov's inequality).

   Flipping specific $n/3$ bits has probability $\leq \left(\frac{1}{n}\right)^{n/3} = n^{-\Omega(n)}$.

   Doesn't happen in time $2^n n \log n$ with prob. $1 - (2^n n \log n) \cdot n^{-\Omega(n)} = 1 - n^{-\Omega(n)}$ (union bound, inequality (A5) in notes).
3. All bits flip together with prob. n^{-n}, does not happen in time $n^{n/2}$ with probability $\geq 1 - n^{n/2} \cdot n^{-n} = 1 - n^{-n/2}$.

$\Rightarrow$ The typical run occurs with probability $1 - 2^{-\Omega(n)}$.

Computationally Hard Problems

Analysis of Randomized Search Heuristics – MSTs

Carsten Witt

Institut for Matematik og Computer Science
Danmarks Tekniske Universitet

Fall 2020

Combinatorial Optimization

Most of the problems you know from algorithms classes are combinatorial optimization problems.

- shortest path problems,
- spanning tree problems,
- matching problems,
- covering problems,
- ...

Aims: understand randomized search heuristics on such problems,

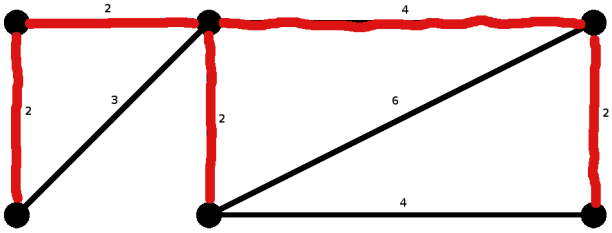
- deriving bound on the optimization time,
- not hoping to be better than the best problem-specific algorithm!

Today: one example, further ones are in the literature.

Minimum Spanning Trees (MSTs)

Problem

Given: Undirected connected graph $G = (V, E)$ with n vertices and m edges with positive integer weights $w\colon E \to \mathbb{N}$.
Find: Edge set $E' \subseteq E$ with minimal weight connecting all vertices.



Solvable in time $O(n \log n + m)$ by textbook algorithms.

The Fitness Function

Model:

- Given an instance to the MST problem,
- reserve for each edge a bit: fitness function $f\colon \{0,1\}^m \to \mathbb{N}$,
- $f(x) =$ weight of the selection if x describes a tree.

To decide: What "fitness" for unconnected graphs?

Problematic: if all unconnected graphs get the same bad fitness
$\Rightarrow$ search heuristic likely to miss connected graphs

Instead: small amount of problem knowledge, e. g.

- start with all edges selected,
- or let the fitness function lead to trees.

The Fitness Function (2/2)

Choice:

$$f(x) := \begin{cases} w_1 x_1 + \cdots + w_m x_m & \text{if } x \text{ encodes a tree} \\ M^2 \cdot (c(x) - 1) \ + & M \cdot (e(x) - (n-1)) \text{ otherwise,} \end{cases}$$

where

- M huge number (e. g., $M = n \cdot (w_1 + \cdots + w_m)$).
- $c(x) =$ no. of connected components in graph encoded by x
- $e(x) = x_1 + \cdots + x_m$ (no. of chosen edges)

Aim: minimize f (replace $\geq$-selection by $\leq$)

Note: unconnected graphs always worse than connected ones, connected non-trees always worse than trees

Upper Bound

Theorem

The expected time until (1+1) EA and RLS$^{\leq 2}$ construct a minimum spanning tree is bounded by $O(m^2(\log n + \log w_{\max})) = O(n^4(\log n + \log w_{\max}))$ where $w_{\max} := \max\{w_1, \ldots, w_m\}$.

RLS$^{\leq 2}$ flips in mutation step either one or two uniformly chosen bits (each of these happening with probablity $1/2$)

Proof outline: three phases

1. From unconnected to connected graphs,
2. from connected graphs to spanning trees,
3. from spanning trees to minimum spanning trees

1. $t := 0$. Choose $x = (x_1, \ldots, x_m) \in \{0,1\}^m$ uniformly at random (u. a. r).

Phase 1

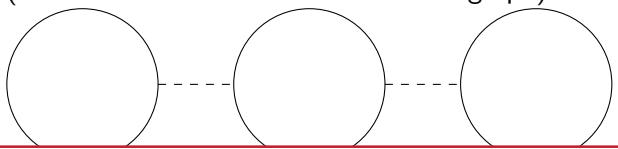
Lemma

The expected time until a connected graph has been reached is $O(m \log n)$.

Proof using fitness-based partitions

- $L_i := \{x \mid (n-i)M^2 > f(x) \geq (n-i-1)M^2\}$,
  i. e., x has $n-i$ CCs, $0 \leq i < n$

  Recall: $f(x) = M^2 \cdot (c(x) - 1) + M \cdot (e(x) - (n-1))$ in Phase 1
- There must be $\geq n-i-1$ edges running between $n-i$ CCs
  (otherwise would not have connected graph).



Phase 2

Lemma

The expected time until a spanning tree has been reached is $O(m \log n)$.

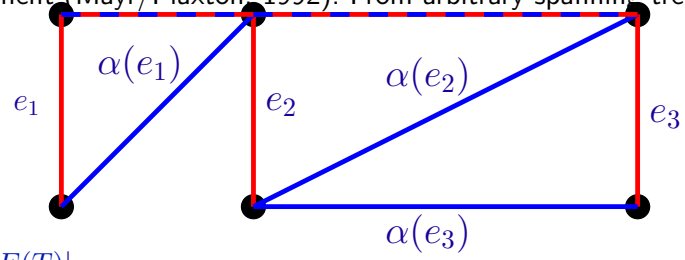
Proof using fitness-based partitions

- $L_i := \{x \mid (m-i)M > f(x) \geq (m-i-1)M\}$,
  i. e., x has $m-i$ edges, $0 \leq i < m-(n-1)$
- At least $m-i-(n-1)$ of these can be flipped out without losing connectedness.
- $s_i = \Omega((m-i-n+1)/m)$ (both for (1+1) EA and RLS).
- Time bound $O(\sum_{i=0}^{m-n+1} m/(m-i-n+1)) = O(m \log m) = O(m \log n)$.

Phase 3

Have reached a tree: How to make progress now?

Combinatorial argument (Mayr/Plaxton, 1992): From arbitrary spanning tree T to MST T^*



- $k := |E(T^*) \setminus E(T)|$
- Bijection $\alpha : E(T^*) \setminus E(T) \to E(T) \setminus E(T^*)$
- $\alpha(e_i)$ on the cycle of $E(T) \cup \{e_i\}$
- $w(e_i) \leq w(\alpha(e_i))$

Phase 3: Analyzing 2-bit Flips

Essence of combinatorial argument:
$\exists\, k \leq n-1$ improving 2-bit flips; doing all of them after another leads from current tree x to optimum x_{OPT}.

- Problem: some 2-bit flips might only yield tiny progress
- But: each of k possibilities improves on average by $\frac{f(x)-f(x_{\mathrm{OPT}})}{k}$.
- A uniformly random 2-bit flip ($\binom{m}{2}$ choices) improves in expectation by $\frac{f(x)-f(x_{\mathrm{OPT}})}{m(m-1)/2}$ (as worsenings never accepted).
- 2-bit flip happens with prob. $\geq \frac{1}{e2^2}$ (with RLS: $1/2$).

Hence: current value $f(x) \rightsquigarrow$ next f-value in expectation $\leq f(x) - \frac{f(x)-f(x_{\mathrm{OPT}})}{2em^2}$, distance to optimum (i. e. $f(x) - f(x_{\mathrm{OPT}})$) decreases by expected factor $\leq 1 - \frac{1}{2em^2}$.

Now: handle the expected distance decrease improvement.

Decrease by Constant Factor: Exercise

You start with 1024 DKK and spend half your money every day. How many days does it take until you have at most 1 DKK left?

10 days.

You start with n DKK and spend a $(1-c)$-fraction (i.e. $(1-c)\cdot 100$ percent) of your money every day. How many days does it take until you have at most 1 DKK left?

Solve $n \cdot c^d \leq 1$

$\Rightarrow d = \frac{-\ln(n)}{\ln(c)} = -\ln_c(n) = \ln_{1/c}(n).$

If you spend a $(1-c)$-fraction only *in expectation* things are similar.

Phase 3: Formalizing Progress

Proof method *multiplicative drift analysis*

Scenario: process with random distances D_t, $t \geq 0$, from optimum

If the D_t are nonnegative integers and $E(D_{t+1} \mid D_t = d) \leq cd$ for $c < 1$

then expected time to reach distance 0
bounded from above $\frac{\ln(D_0)+1}{1-c}$.

Without proof.

Phase 3: Applying the Multiplicative Drift Technique

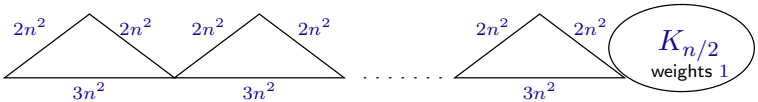
Back to MST problem:

- Distance decrease by factor $c = 1 - \frac{1}{2em^2}$,
- initial distance at most $w_1 + \cdots + w_m \leq m \cdot w_{\max}$,
- expected length of Phase 3: $O\left(\frac{\ln(m \cdot w_{\max})+1}{1-(1-1/(2em^2))}\right) = O(m^2(\log m + \log w_{\max}))$ $= O(m^2(\log n + \log w_{\max}))$.

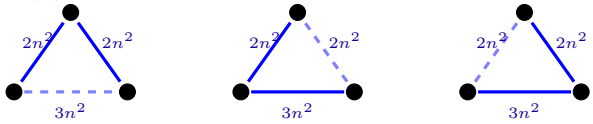
All phases together: still expected time $O(m^2(\log n + \log w_{\max}))$.

Lower Bound: Idea

$\exists$ instance where (1+1) EA and RLS$^{\leq 2}$ take $\Omega(n^4 \log n)$ steps.



- $\Omega(n)$ triangles, each with 2 wrong and 1 right configurations



- Clique on $n/2$ nodes makes the graph have $m = \Omega(n^2)$ edges
- Two edges of triangle flipped with prob. $\Theta(1/m^2) = \Theta(1/n^4)$.
- Coupon collector argument $\rightarrow$ time $\Omega(n^4 \log n)$.

Summary

- Studied (1+1) EA and RLS on a combinatorial optimization problem, the MST problem
- Polynomial bound $O(m^2(\log n + \log w_{\max}))$, which is $O(m^2 \log n) = O(n^4 \log n)$ unless weights very large
- Lower bound $\Omega(n^4 \log n)$ on a specific instance
- Proof techniques:
  - fitness levels
  - multiplicative drift