Computational Complexity

and

Evolutionary Computation

Ingo Wegener, Univ. Dortmund, Germany

More precisely:

How to apply methods from

complexity theory

and

classical algorithm analysis

to evolutionary computation

Aims: The EC community should know:

there are powerful methods from complexity theory

and analysis of (randomized) algorithms

which can be applied to

evolutionary computation

But why?

These methods lead to

- theorems without any assumptions
- theorems on the algorithm and

  not on a model of the algorithm
- theorems for arbitrary problem dimension

1. Introduction (survey later)

We discuss search heuristics
(= randomized algorithms)

including EA, ES, GA, GP, Sim. Ann., tabu search

for some kind of optimization

$\longrightarrow$ Restriction: discrete search spaces

Different types of problems:

one-shot scenario:	one function $\longrightarrow$ no theory
problem-specific scenario:	TSP, scheduling, ...
structural scenario:	pseudo-boolean polynomials
	degree $\leq d$, $\leq N$ terms,
	positive weights, ...

The scenario

Problem: Class of functions

– all linear functions $f\colon \{0,1\}^n \rightarrow \mathbb{R}$

– all TSP-functions

$f_D(\pi) =$ cost of tour π

w.r.t. distance matrix D

Instance: one specific of these functions

instance is known (cost matrix for TSP)
and can be used by the algorithm

Important → / ↘

instance is not known
(black-box optimization)

trivial problem

Needle in the haystack → / ↘

difficult problem

Given a problem and an algorithm –

what do we want to know?

The probability distribution of the "state"

of the algorithm depending on t and the instance

$\longrightarrow$ impossible in non-trivial situations

$\longrightarrow$ expected time until good event

(optimum found) happens

variance, moments, ...

success probabilities

$\longrightarrow$ only good estimates are possible

DON'T TRY TO BE TOO EXACT!

YOU WILL FAIL

Typical EA-theory approaches:

$\longrightarrow$ reasonable model, calculation in the model, experiments to "verify" the model

$\longrightarrow$ no result for large problem dimension n

$\longrightarrow$ infinite populations

$\longrightarrow$ how to control the error?

$\longrightarrow$ studying the dynamics of the stochastic process

$\longrightarrow$ what is the meaning of the results?

$\longrightarrow$ studying the one-step behavior
(schema theory, quality gain, progress rate, . . .)

$\longrightarrow$ what happens in many steps?

$\longrightarrow$ building block hypothesis

$\longrightarrow$ just a nice hypothesis (royal roads)

$\longrightarrow$ convergence results

$\longrightarrow$ I do not have enough time!

DON'T TRY TO BE TOO GENERAL!

RESULTS ARE NECESSARILY BAD

Methods from complexity theory and

classical algorithm analysis:

- no assumptions
- results about the algorithm
- only (good) estimates
- error can be controlled

  (upper and lower bounds)

$\longrightarrow$ theorems (!), mathematically proven, for all problem dimensions n and instances

$\longrightarrow$ useful in 10 or 100 years

$\longrightarrow$ no verification by experiments

$\longrightarrow$ experiments are useful: what happens between the lower and the upper bound?

2. Survey on the rest of the talk

I Complexity Theory

3. NFL scenario vs. realistic scenarios

4. Yao's minimax principle –

   lower bounds in the black-box scenario

II Algorithm analysis (with concrete examples)

5. The coupon collector's theorem

6. Chernoff bounds

7. Random walks on plateaus

8. Potential functions

9. Typical runs

III Applications to classical problems

10. Sorting

11. Shortest paths

12. Minimum spanning trees

13. Maximum matchings

IV

14. Conclusions

3. The NFL scenario vs. realistic scenarios

NFL-Theorem: A, B finite. Each randomized search strategy sampling no point twice has on the average of all $f: A \to B$ the same behavior (expected optimization time, success probability, ...)

Holds iff class of functions is closed under permutations

The proof is simple – the result is fundamental

– the scenario is not realistic

We never optimize a function without

– a polynomial-time evaluation algorithm $(a, f) \to f(a)$

– a short description

– structure on the search space

E.g., $A = \{0,1\}^{100}$ and $B = \{1,\ldots,10000\}$

$\#\{f \mid f\colon A \to B\} = 10000^{2^{100}}$

Almost all f have a shortest description length of $\geq 2^{100} \log 10000 - 100$

(Kolmogorov complexity $\rightarrow$ all types of description)

$\longrightarrow$ almost all functions will never be considered

(the same for permutations on A)

Realistic scenarios are resource bounded

$\longrightarrow$ no NFL theorem (DJW GECCO'99)

Almost NFL theorem (DJW TCS'02)

Each rand. search heuristic efficient on f (easy to describe) is bad for many g which are easy to describe and closely related to f

The NFL theorem is fundamental
and everything has been said on it

Essential arguments were known before in complexity theory

It is time to stop the discussion on NFL

Lessons learned

Each rand. search heuristic realizes a certain idea
about the structure of the considered problem type and fails
if the problem does not have this structure

Knowing $f(a_1), \dots, f(a_t)$ (t not too large)
has to imply some knowledge where to look for good search points

4. Yao's minimax principle – lower bounds in the black-box scenario

The black-box scenario:

Given a class of functions $F \subseteq \{f\colon A \to B\}$.

The function $f \in F$ to be optimized is unknown

(is chosen by an adversary or "the real world")

$\rightarrow$ Search by sampling

Step t:

we know $a_1, f(a_1), \ldots, a_{t-1}, f(a_{t-1})$,

we choose a_t (the prob. distribution to choose a_t)

$\longrightarrow$ we obtain $f(a_t)$

Note that

EA, ES, GA, Sim Ann, ... fit into this scenario

- We can analyse what is not possible in this setting

- Lower bounds show the limits of all randomized search heuristics

- How can we obtain such lower bounds?

Yao's Minimax Principle (1978)

(Andy Yao, Turing Award Winner 2001)

Consider black-box optimization as zero-sum game between

Player 1: the algorithm designer
Player 2: the adversary choosing the instance f

Player 1 has to pay 1 \$ for each f-evaluation

Condition

– Number of problem instances is finite

– Number of deterministic search strategies

  is finite (forget repeated tests)

The miracle:

Lower bounds for deterministic algorithms
imply lower bounds for randomized algorithms

Theorem

The minimal (w.r.t. **randomized** algorithms A)
maximal or worst-case (w.r.t. problem instances f)
expected optimization time $T(A, f)$)

$\geq$ maximal (w.r.t. prob. dist. p on instances f)
minimal (w.r.t. **deterministic** algorithms A)
average optimization time $T_p(A, f)$)

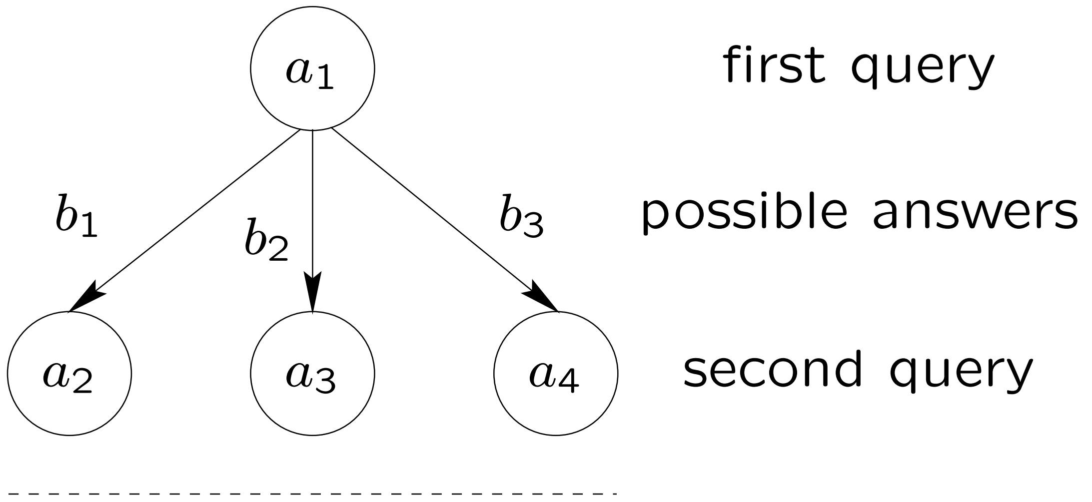
$\geq \min_A E(T_p(A, f))$ for each p

This theorem for two-persons zero-sum games
is 50 years old (von Neumann)

The new idea is to consider algorithm
design as such a game

Note: We can choose p and have to
investigate deterministic algorithms only

Deterministic search strategies are decision trees.



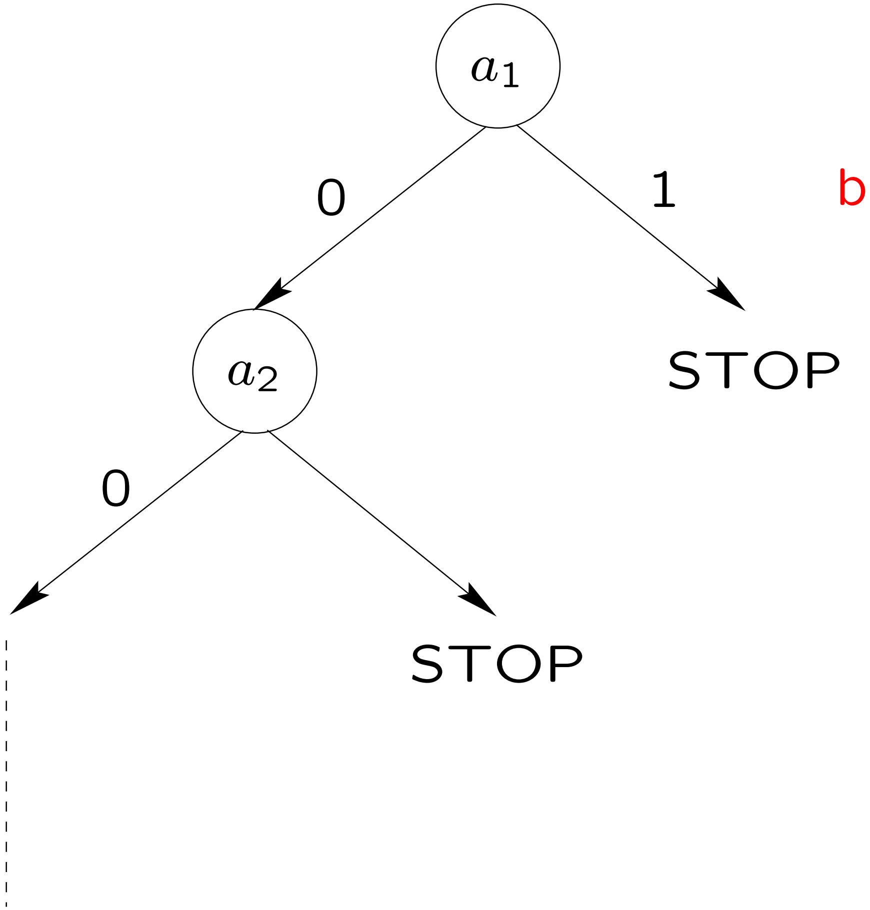
for each f:

optimization time = # nodes on query path until query point is optimal.

Applications (DJTW – FOGA '02) and new

Needle in the haystack

all $f_a(x) = \begin{cases} 1 & x = a \\ 0 & \text{otherwise} \end{cases}$ uniform distribution



black box complexity $2^{n-1} + \frac{1}{2}$

EAs are efficient,
they are slow but close to
lower bound

Trap

all $f_a(x) = \begin{cases} 2n & x = a \\ \text{ONEMAX}(x) & \text{otherwise} \end{cases}$

lower bound: $2^{n-1} + \frac{1}{2}$

random search: $2^{n-1} + \frac{1}{2} \leftarrow$ optimal

typical EAs: $\Theta(n^n) = \Theta(2^{n \log n}) \leftarrow$ inefficient

Unimodal functions

$f\colon \{0,1\}^n \to \mathbb{R}$ is unimodal iff for all a:
a is optimal or has a better Hamming neighbor

Easy: $\mathrm{Im}(f)$ image set $\Rightarrow$
expected optimization time of $(1+1)$EA: $O(n \cdot |\mathrm{Im}(f)|)$

(common belief: unimodal $\Rightarrow$ easy for EAs)

But:

Each randomized search heuristic needs for many unimodal functions on average

$$\Omega(|\mathrm{Im}(f)|/n^{\varepsilon}) \text{ steps, } \varepsilon > 0.$$

The result ist counterintuitive!?

No, the common belief is based on a too general statement.

Consider randomized long path functions:

- $p_0 = 1^n$
- p_i random Hamming neighbor of p_{i-1}
- eliminate loops

$$\longrightarrow f_P(a) = \begin{cases} n+i & a = p_i \\ \text{ONEMAX}(a) & \end{cases}$$

$p_0, \ldots, p_i$ and some points outside P known:
no chance to guess p_{i+j} for some j not too small

Now: Algorithm analysis

5. The Coupon Collector's Theorem

The best-known analysis of an EA:
expected optimization time of $(1+1)$EA on ONEMAX:

$$\Theta(n \log n)$$

Can we break the $n \log n$ barrier
(for functions with a unique global optimum)?

Children's problem:

With each bar of chocolat you get a picture of one of 20 players of one of 18 teams.

How many bars do you expect to buy until you have a complete collection of pictures?

Expected value

$$360\left(1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{360}\right)\approx 2300$$

Better: swap pictures with your friends

In general

$$n\left(1+\frac{1}{2}+\cdots+\frac{1}{n}\right)\approx n\ln n+0.58\ldots n$$

The Coupon Collector's Theorem says
this is a sharp threshold result, i.e.,

prob. that $(1-\varepsilon)n \ln n$ pictures are enough
$\longrightarrow$ 0 exponentially fast
prob. that $(1+\varepsilon)n \ln n$ pictures are not enough
$\longrightarrow$ 0 exponentially fast

expected value is close to be correct (almost always)

Pick the incorrect bits of a random search point ($\sim n/2$), mutation probability $1/n$

$\rightarrow$ time $n \ln n \pm \Theta(n)$ until all wrong bits have flipped once

One-point crossover:

If you need a crossover at εn given positions:
$\longrightarrow$ time $n \ln n \pm \Theta(n)$ until this has happened

$\longrightarrow$ there is an $n \log n$ barrier

6. Chernoff bounds

$X_1, \ldots, X_n$ independent $0 - 1$ random variables
$X = X_1 + \cdots + X_n$ (number of successes)
$\text{Prob}(X_i = 1) = p_i$ for some $0 < p_i < 1$
$\Rightarrow$
$E(X) = p_1 + \cdots + p_n$
$0 < \delta < 1$: $\text{Prob}\,(X \le (1 - \delta) \cdot E(X)) \le \mathrm{e}^{-E(X)\delta^2/2}$

The bounds are close to optimal

Choose $a \in \{0,1\}^n$ randomly

exp. number of ones:	$n/2$
Prob(#ones $\leq 0.4n$)	expo. small
Prob(#ones $\leq n/2 - n^{3/4}$)	weakly expo. small
Prob(#ones $\leq n/2 - n^{1/2}$)	a positive constant

Applications

Probability of fitness increasing step $\frac{1}{n}$

$\rightarrow$ almost surely $\Theta(n^2)$ steps to increase fitness n times

$\longrightarrow$

DO NOT INVESTIGATE SINGLE STEPS –
INVESTIGATE PHASES OF MODERATE LENGTH

We can estimate the prob. of bad events

Mutation prob. $1/n$, phase length n^2

Prob(x_i has flipped less than $0.9n$ times
or more than $1.1n$ times) = expo. small

$$
\begin{aligned}
\text{Prob}(\exists x_i : x_i \ldots) &\leq n \cdot \text{ expo. small} \\
&= \text{expo. small}
\end{aligned}
$$

7. Random walks on plateaus

$f : \{0,1\}^n \rightarrow \{0,1,\ldots,N\}$
$n = 100$ $N = 10^6$, 2^{100} search points $\rightarrow$
many have the same fitness

Plateau $i = \{a | f(a) = i\}$

Populations sitting on a plateau search
for the exit to a higher plateau

Such a search is a random walk –
fitness gives no hints

Example 1 (JW - IEEE.Trans on EC, 2000)

$$f(a) = \begin{cases} 2n & a = 1^n \\ n & a = 0^i 1^{n-i} \\ n - \text{ONEMAX}(a) & \text{otherwise} \end{cases}$$

Plateau on level n: a path with n points

00000—00001—00011—00111—01111 ⌒ 11111

It is easy to find the path –
then $(1+1)$ EA with mutation probability $1/n$:

prob(child on the path) $= \Theta\left(\frac{1}{n}\right)$
(Chernoff $\Rightarrow n \cdot \#$ successful steps)

Random walk needs n more steps in the
good direction (if starting in 0^n)

Steps of length ≥ 2 are "fair"

Prob(among cn^2 steps of length 1
are $\geq \frac{1}{2}cn^2 + \frac{1}{2}n$ in the good direction) $= \delta > 0$

Expected number of phases $\leq 1/\delta$

$\rightarrow$ Expected optimization time: $\Theta(n^3)$

Example 2 (FW - GECCO'2004)

Ising model (Naudts, von Hoyweghen, Goldberg, . . .
difficult because of symmetry)

$$a_1 \quad - \quad a_2 \quad - \quad a_3 \quad - \quad a_4 \quad - \quad a_5 \quad - \quad a_6 \quad - \quad a_7 \quad - \quad a_8 \quad - \quad a_9 \quad - \quad a_{10}$$

$f(a) = n -$ number of 2-colored edges

Likely: $0^i 1^j 0^{n-i-j}$

The 0-1-walls take a random walk
– until they meet

GAs need niching

$(1+1)$ EA $O(n^3)$

8. Potential functions

The selection steps of the EA are based
on the fitness –
may be difficult to analyse –
in particular, if we analyse classes of functions,
e.g., all linear functions

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

Idea from classical algorithm analysis:

– find artificial “fitness” (called potential) to measure the progress of the search according to the potential function (the EA uses still the real fitness)

Difficult: the right intuition to define a suitable potential function

First application in EC theory (DJW - WCCI'98, TCS'02)

Linear functions, w.l.o.g. $w_1 \geq w_2 \geq \cdots \geq w_n > 0$

potential function $2x_1 + \cdots + 2x_{n/2} + x_{n/2+1} + \cdots + x_n$

$\longrightarrow$ a drift anaysis is possible

$\longrightarrow$ $\Theta(n \log n)$

Also maximum matchings

$G = (V, E)$ undirected graph

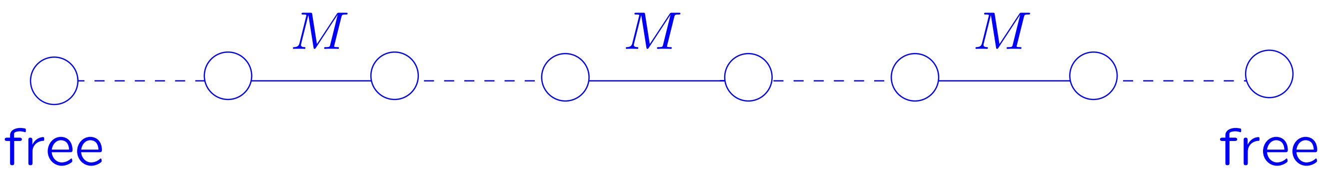
$E' \subseteq E$ matching $\Leftrightarrow$

edges in E' have no vertex in common

$$\text{Fitness} = \begin{cases} |E'| \text{ for matchings} \\ - \text{ number of forbidden edge pairs} \end{cases}$$

$\rightarrow$ one of the classical optimization problems in P

Theory of augmenting paths



potential function =

$$n \cdot \text{ fitness} - \text{length of shortest augm. path}$$

(results later)

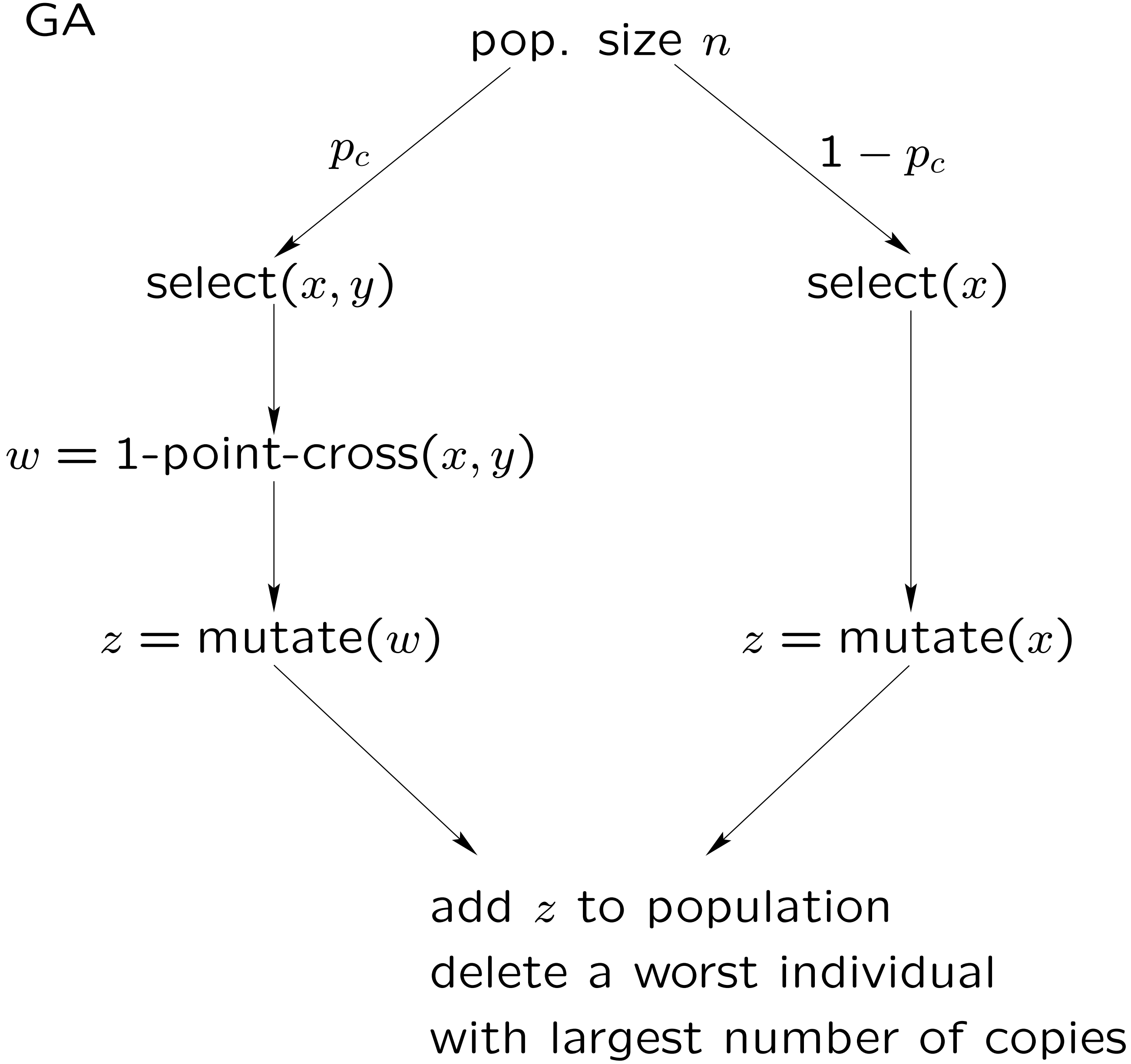
9. The analysis of typical runs

Use intuition to describe what
typically happens,
define phases with well-defined subgoals,
estimate the probability that something
goes wrong

Example JW - GECCO'01

the first example where provably
mutation-based EAs need exponential time
and a generic steady-state GA has a
polynomial expected optimization time

GA


pop. size n
p_c
$1 - p_c$
select(x, y)
select(x)
$w =$ 1-point-cross(x, y)
$z =$ mutate(w)
$z =$ mutate(x)
add z to population
delete a worst individual
with largest number of copies

Condition: $f(x) \geq f(y) \Rightarrow \text{Prob}(\text{select}(x)) \geq \text{Prob}(\text{select}(y))$

Real royal roads

block length $b(a)$ = length of longest 1-block

$$1100010111001 \rightarrow b(a) = 4$$

$$f(a) = \begin{cases} 2n^2 & a = 1^n \\ n \cdot \text{ONEMAX}(a) + b(a) & \text{ONEMAX}(a) \leq (2/3)n \\ 0 & \text{otherwise} \end{cases}$$

Phase 1: all individuals have positive fitness

(Chernoff) $1 + o(1)$

Phase 2: optimal individual or

all individuals have $(2/3)n$ ones

(success probability $\geq \varepsilon$ for

potential # ones in population) $O(n^2)$

Phase 3: optimal individual or all

individuals have block length $(2/3)n$

(duplicates and 2-bit mutations help

for potential sum of block lengths) $O(n^2 \log n)$

Phase 4: optimal individual or population

contains all different

second-best individuals

(2-bit mutations and potential

number of diff. second-best ind.) $O(n^4)$

Phase 5: successful search

$1\ldots1$	$1\ldots1$	$0\ldots0$
$0\ldots0$	$1\ldots1$	$1\ldots1$

↑
good cuts

Choose these individuals for crossover,

choose a good cut position and do

not flip any bit afterwards $O(n^2)$

III Applications to classical problems

Does this all work only for toy examples?

No,
we investigate well-known problems with
polynomial-time problem-specific algorithms

10. Sorting (STW – PPSN ’02 and new)

– Nobody tries to beat quicksort!

– Here sorting is the maximization of

sortedness in a sequence and

the scenario is the black-box scenario

– Well-known measures of sortedness:

– INV(π) (inversions) =

number of pairs in incorrect order $\longrightarrow$ minimization

– HAM(π) (Hamming distance) =

number of objects at incorrect position $\longrightarrow$ minimization

– RUN(π) (runs) =

number of maximal sorted blocks $\longrightarrow$ minimization

– REM(π) (removals) $=$

minimal number of removals to obtain a sorted subsequence

2 3 7 1 4 5 6 9 8 $\rightarrow$ REM=3

– EXC(π) (exchanges) $=$

minimal number of exchanges to sort the sequence

$\rightarrow$ minimization

$\longrightarrow$ In black-box scenario five different problems

Mutation-based $(1+1)$EA

- s (Poisson distributed $\lambda = 1$)
  $\rightarrow s$ local changes

- exchange (i, j)

  6 8 1 2 4 7 5 3

  6 4 1 2 8 7 5 3

  jump (i, j)

  6 8 1 2 4 7 5 3

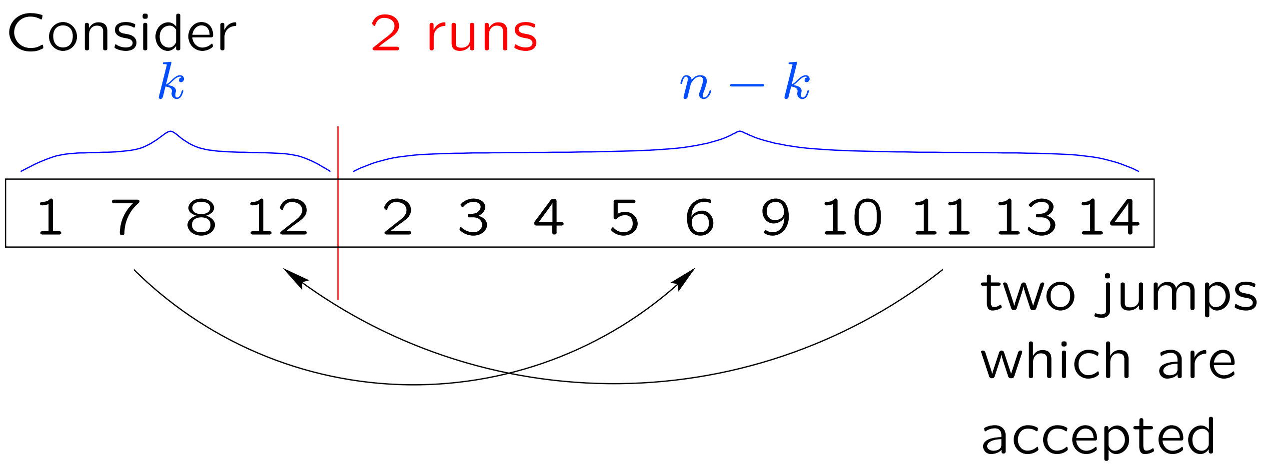
  6 4 8 1 2 7 5 3

  6 4 8 2 7 5 3 1

INV	$O(n^2 \log n)$	$\Omega(n^2)$	exchanges, jumps
REM	$O(n^2 \log n)$	$\Omega(n^2 \log n)$	jumps
HAM	$O(n^2 \log n)$	$\Omega(n^2)$	exchanges
EXC	$O(n^2 \log n)$	$\Omega(n^2)$	exchanges

typical runs, subgoals, Chernoff bounds, . . .

What about RUN?



We search on the plateau with fitness 2

Exchanges are almost useless

Jumps can change the lengths of the runs

$k < n - k$

k	jumps	shorten shorter run
$n-k$	jumps	lengthen shorter run

Random walk is "unfair" $\longrightarrow$ exponential time

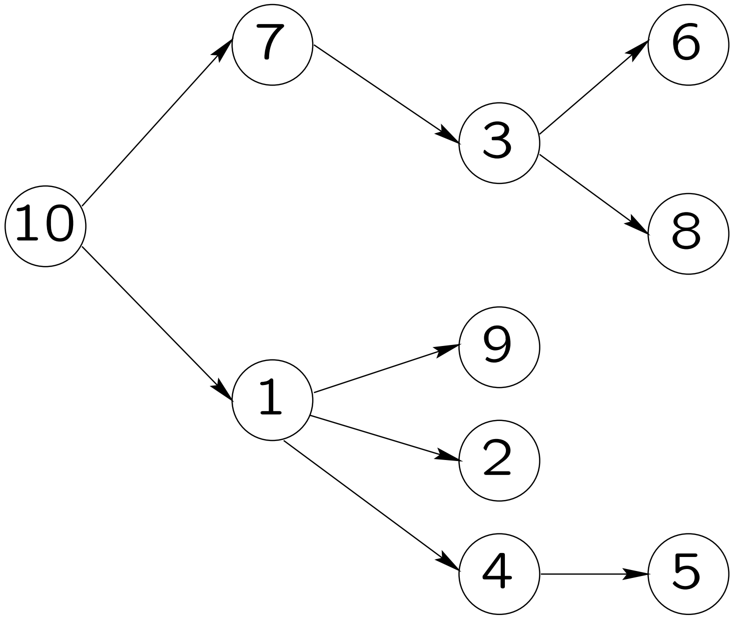
11. Shortest paths (STW – PPSN '02)

Single source shortest paths (Dijkstra problem)

Distance matrix

Shortest paths from $s = n$ to all other places i —

how to encode the individuals?



(10, 1, 7, 1, 4, 3, 10 , 3, 1) –

vector of direct predecessors

fitness = sum of path lengths

Yao's minimax principle

$\longrightarrow$

no polynomial-time black-box search heuristic

The problem is a multi-objective

optimization problem

fitness = vector of path lengths

search for Pareto optima w.r.t. to "$\leq$"

$(l_1, \ldots, l_{n-1}) \leq (l'_1, \ldots, l'_{n-1})$ iff $\forall i\colon l_i \leq l'_i$

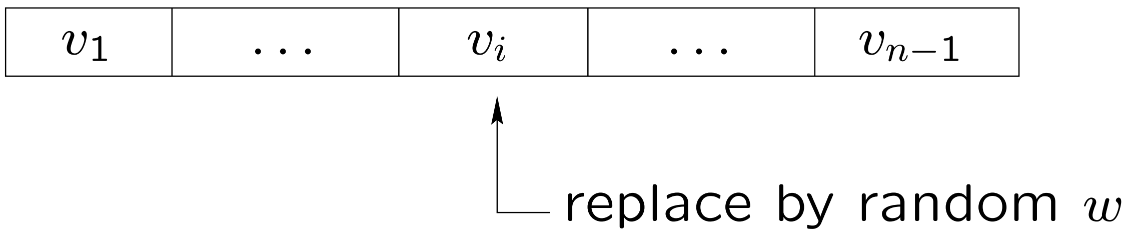
Pareto optimum is unique in this case

Analysis of mutation-based EA

– again number of local changes s

where s is Poisson distributed $\lambda = 1$

– local change



$\longrightarrow$ $O(n^3)$ with our standard techniques

12. Minimum Spanning Trees

(NW – GECCO'2004)

Graphs $G = (V, E)$ on n vertices with m edges.

$w\colon E \to \mathbb{N}$ weight function.

Find an edge set describing a minimum spanning tree.

Search space $S = \{0, 1\}^m$, i. e.,
x describes the choice of the edges e_i where $x_i = 1$.

$f(x) := n \cdot$ number of connected components
$+$ weight of chosen edges.

Standard: $O(m \log n)$ until we have search points describing connected graphs.

Edges in cycles can be eliminated.

Aim: Add a cheap edge which creates a cycle and eliminate a more expensive edge from a cycle.

There can be many of these steps
leading to a small improvement

or

there can be few of these steps
leading to a large improvement.

$\longrightarrow$

A bound for the expected multiplicative weight decrease.

Time bound: $O(n^2m(\log n + \log w_{\max}))$.

This bound is best possible for the (1+1) EA.

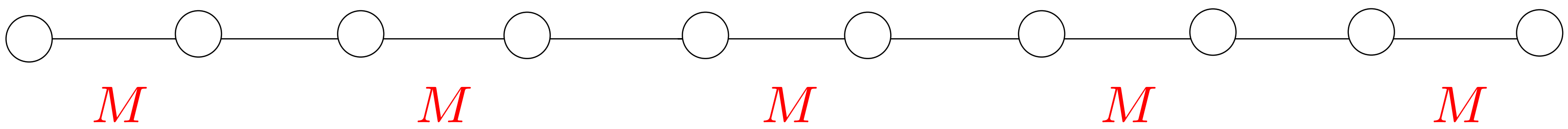
This is much worse than Kruskal's algorithm
– but polynomial.

However, the algorithm does not apply
any knowledge about the problem.

13. Maximum matchings

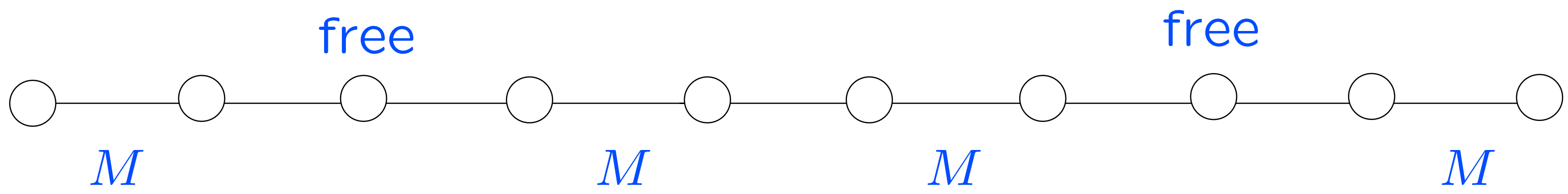
(GW – STACS '03 and new)

A simple case – a path



optimal solution

perhaps algorithm finds a matching of size 4

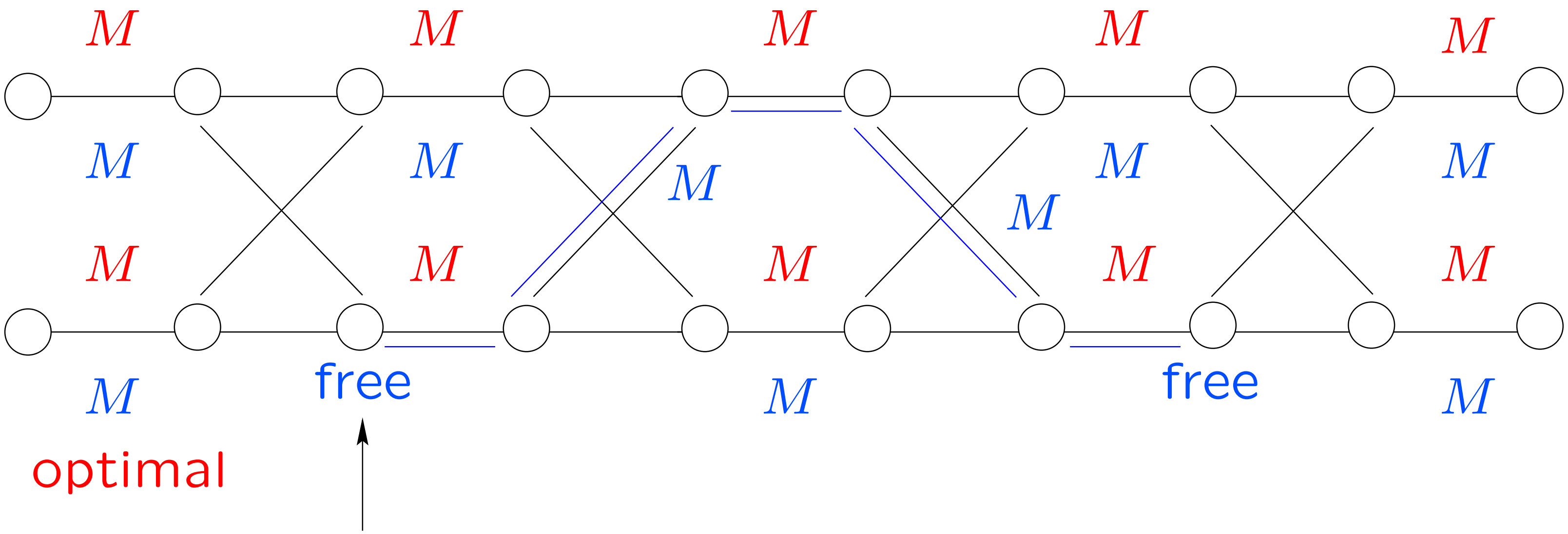


length of augmenting path: 5

2-bit mutations can shorten or lengthen the

augmenting path

almost fair random walk on a plateau: $O(n^4)$



One 2-bit mutation shortens the augmenting path
Two 2-bit mutations lengthen the augmenting path

→ unfair random walk on a plateau
(analysed with potential function) → expo. time

However, the aim of search heuristics is approximation and not exact optimization

For graphs on m edges, a mutation-based hill climber finds a matching of size $\geq (1-\varepsilon)$ opt. size in expected time $O(m^{2/\varepsilon})$

(polynomial-time randomized approximation scheme)

14. Conclusions

– EAs are algorithms and should be analysed as other algorithms

– Algorithm analysis has a long history, is a fundamental discipline of computer science, deep results and clever methods are known

- The EA community has adopted methods from physics, engineering, experimental disciplines but not from theoretical computer science

- EAs are considered as black sheeps in the family of algorithms if you ask the algorithm community

- Results like those presented here have started to change this

- Theoretical results on EAs should be published also in journals / conferences of theoretical computer science

- This happened:
  Journals: TCS, Algorithmica, Journal of Discrete Algorithms, Combinatorics, Probability and Computing, Discrete Applied Mathematics
  Conferences: ICALP, WG, MFCS, EMS (invited)
  STACS, ESA

I hope that you and others from the EA community will apply the strong methods from classical algorithm analysis (and sometimes also complexity theory) from now on.