Algorithms & Complexity

Lecture 6: Randomized Algorithms

October 31, 2019

CentraleSupélec / ESSEC Business School

Dimo Brockhoff

Inria Saclay – Ile-de-France

Discussion Home Exercise

Exercise 1: Greedy Algorithm vs. Dynamic Programming

Correct statements:

a) In a dynamic programming approach, we make at each step a decision considering the current problem and the solution(s) to previously solved sub-problem(s).

b) It is guaranteed that a dynamic programming approach will generate an optimal solution as it generally considers all possible cases and then choose the best.

c) A greedy algorithm follows the problem solving heuristic of making the locally optimal choice at each stage.

d) A problem should possess the property of overlapping subproblems to make a dynamic programming approach an efficient alternative.

e) A greedy algorithm is more efficient in terms of memory than a dynamic programming approach as it never looks back or revises previous choices.

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

$$A_1 \cdot A_2 \cdots A_n$$

1) Conditions on matrix sizes (A is a_i times b_i matrix):

$$\forall 1 \leq i < n\text{: } b_i = a_{i+1}$$

2) Example: 4x3 (matrix) times 3x1 times 1x3 times 3x4
   number of calculations:
   i) (4x3 times (3x1 times 1x3)) times 3x4 [greedy]
   → $3 \cdot 1 \cdot 3 + 4 \cdot 3 \cdot 3 + 4 \cdot 3 \cdot 4 = 9 + 36 + 48 = 93$
   ii) (4x3 times 3x1) times (1x3 times 3x4) [better than greedy]
   → $4 \cdot 3 \cdot 1 + 1 \cdot 3 \cdot 4 + 4 \cdot 1 \cdot 4 = 12 + 12 + 16 = 40$

Definition: $C(i,j)$:= number of calculations to calculate $A_i \cdots A_j$

3) Easy to compute: $C(i,i) = 0$ and $C(i,i+1) = a_i \cdot \mathrm{b}_i \cdot b_{i+1}$
4) Sought value (optimum): $C(1,n)$

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

5) Assumption: $A_i \cdots A_j$ optimally computed as $(A_i \cdots A_k) \cdot (A_{k+1} \cdots A_j)$
   then: $C(i,j) = C(i,k) + C(k+1,j) + a_i \cdot b_k \cdot b_j$
6) In general:

$$C(i,j) = \begin{cases} 0 & \text{if } i = j \\ a_i \cdot b_i \cdot b_j & \text{if } j = i+1 \\ \min_{i \le k < j} C(i,k) + C(k+1,j) + a_i \cdot b_k \cdot b_j & \text{else} \end{cases}$$

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

7) Matrices: A1 (5-by-2), A2 (2-by-10), A3 (10-by-1), A4 (1-by-10), and A5 (10-by-2)

i/j	1	2	3	4	5
1					
2	-				
3	-	-			
4	-	-	-		
5	-	-	-	-	

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

7) Matrices: A1 (5-by-2), A2 (2-by-10), A3 (10-by-1), A4 (1-by-10), and A5 (10-by-2)

i/j	1	2	3	4	5
1	0	100			
2	-	0	20		
3	-	-	0	100	
4	-	-	-	0	20
5	-	-	-	-	0

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

7) Matrices: A1 (5-by-2), A2 (2-by-10), A3 (10-by-1), A4 (1-by-10), and A5 (10-by-2)

i/j	1	2	3	4	5
1	0	100			
2	-	0	20		
3	-	-	0	100	
4	-	-	-	0	20
5	-	-	-	-	0

$$\mathrm{C}(1,3) = \min\{0 + 20 + 5 \cdot 2 \cdot 1, 100 + 0 + 5 \cdot 10 \cdot 1\} = \min\{30, 150\}$$

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

7) Matrices: A1 (5-by-2), A2 (2-by-10), A3 (10-by-1), A4 (1-by-10), and A5 (10-by-2)

i/j	1	2	3	4	5
1	0	100	30		
2	-	0	20		
3	-	-	0	100	
4	-	-	-	0	20
5	-	-	-	-	0

$$\mathrm{C}(2,4) = \min\{0 + 100 + 2 \cdot 10 \cdot 10, 20 + 0 + 2 \cdot 1 \cdot 10\} = \min\{300, 40\}$$

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

7) Matrices: A1 (5-by-2), A2 (2-by-10), A3 (10-by-1), A4 (1-by-10), and A5 (10-by-2)

i/j	1	2	3	4	5
1	0	100	30		
2	-	0	20	40	
3	-	-	0	100	
4	-	-	-	0	20
5	-	-	-	-	0

$$\mathrm{C}(3,5) = \min\{0 + 20 + 10 \cdot 1 \cdot 2, 100 + 0 + 10 \cdot 10 \cdot 2\} = \min\{40, 300\}$$

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

7) Matrices: A1 (5-by-2), A2 (2-by-10), A3 (10-by-1), A4 (1-by-10), and A5 (10-by-2)

i/j	1	2	3	4	5
1	0	100	30		
2	-	0	20	40	
3	-	-	0	100	40
4	-	-	-	0	20
5	-	-	-	-	0

$$\mathrm{C}(1,4) = \min\begin{Bmatrix} 0+40+5\cdot 2\cdot 10 \\ 100+100+5\cdot 10\cdot 10 \\ 30+0+5\cdot 1\cdot 10 \end{Bmatrix} = \min\{140, 700, 80\}$$

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

7) Matrices: A1 (5-by-2), A2 (2-by-10), A3 (10-by-1), A4 (1-by-10), and A5 (10-by-2)

i/j	1	2	3	4	5
1	0	100	30	80	
2	-	0	20	40	
3	-	-	0	100	40
4	-	-	-	0	20
5	-	-	-	-	0

$$C(2,5) = \min \begin{Bmatrix} 0 + 40 + 2 \cdot 10 \cdot 2 \\ 20 + 20 + 2 \cdot 1 \cdot 2 \\ 40 + 0 + 2 \cdot 10 \cdot 2 \end{Bmatrix} = \min\{80, 44, 80\}$$

Discussion Home Exercise

Exercise 2: Matrix Chain Multiplications

7) Matrices: A1 (5-by-2), A2 (2-by-10), A3 (10-by-1), A4 (1-by-10), and A5 (10-by-2)

i/j	1	2	3	4	5
1	0	100	30	80	
2	-	0	20	40	44
3	-	-	0	100	40
4	-	-	-	0	20

$$C(1,5) = \min \begin{Bmatrix} 0 + 44 + 5 \cdot 2 \cdot 2 \\ 100 + 40 + 5 \cdot 10 \cdot 2 \\ 30 + 20 + 5 \cdot 1 \cdot 2 \\ 80 + 0 + 5 \cdot 10 \cdot 2 \end{Bmatrix} = \min\{64, 240, 60, 180\}$$

Discussion Home Exercise

And the actual solution?

→ need to store where optimum was obtained

-10),

i/j	1	2	3	4	5
1	0	100	30	80	60
2	-	0	20	40	44
3	-	-	0	100	40
4	-	-	-	0	20

$$C(1,5) = \min \left\{ \begin{matrix} 0 + 44 + 5 \cdot 2 \cdot 2 \\ 100 + 40 + 5 \cdot 10 \cdot 2 \\ 30 + 20 + 5 \cdot 1 \cdot 2 \\ 80 + 0 + 5 \cdot 10 \cdot 2 \end{matrix} \right\} = \min\{64, 240, 60, 180\}$$

Discussion Home Exercise

And the actual solution?

→ need to store where optimum was obtained

-10),

i/j	1	2	3	4	5
1	0	100	30	80	60
2	-	0	20	40	44
3	-	-	0	100	40
4	-	-	-	0	20

$$C(1,5) = \min \begin{Bmatrix} 0 + 44 + 5 \cdot 2 \cdot 2 \\ 100 + 40 + 5 \cdot 10 \cdot 2 \\ 30 + 20 + 5 \cdot 1 \cdot 2 \\ 80 + 0 + 5 \cdot 10 \cdot 2 \end{Bmatrix} = \min\{64, 240, 60, 180\}$$

Discussion Home Exercise

And the actual solution?

→ need to store where optimum was obtained

-10),

i/j	1	2	3	4	5
1	0	100	30	80	60
2	-	0			
3	-	-			
4	-	-			

$(A_1 \ldots A_3) \cdot \ (A_4 \cdot A_5)$

then: $(A_1 \cdot (A_2 \cdot A_3)) \cdot \ (A_4 \cdot A_5)$

$$C(1{,}5) = \ \min \left\{ \begin{array}{c} 0 + 44 + 5 \cdot 2 \cdot 2 \\ 100 + 40 + 5 \cdot 10 \cdot 2 \\ 30 + 20 + 5 \cdot 1 \cdot 2 \\ 80 + 0 + 5 \cdot 10 \cdot 2 \end{array} \right\} = \min\{64, 240{,}60{,}180\}$$

Course Overview

Thu		Topic
Thu, 12.09.2019	PM	Introduction, Combinatorics, O-notation, data structures
Tue, 24.09.2019	PM	Sorting algorithms I
Tue, 1.10.2019	PM	Sorting algorithms II, recursive algorithms
Tue, 8.10.2019	PM	Recursive and Greedy Algorithms
Tue, 15.10.2019	PM	Dynamic programming
➨ Thu, 31.10.2019	AM	Randomized Algorithms and Blackbox Optimization
Tue, 5.11.2019	PM	Complexity theory I
Tue, 26.11.2019	PM	Complexity theory II
Tue, 17.12.2019	AM	Exam (written)

Randomized Algorithms and Blackbox Optimization

Coping with Difficult Problems

Exact

- brute-force often too slow
- better strategies such as dynamic programming
- still: often exponential runtime

Approximation Algorithms

- guarantee of low run time
- guarantee of high quality solution
- obstacle: often difficult to prove these guarantees

Heuristics

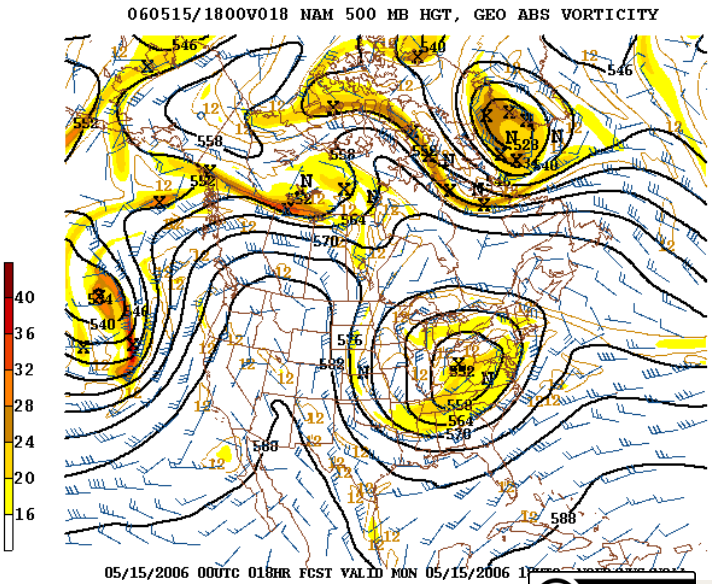
- intuitive algorithms
- guarantee to run in short time
- often no guarantees on solution quality
- designed for practice (become non-heuristic once theoretically analyzed ☺)

Randomized Algorithms

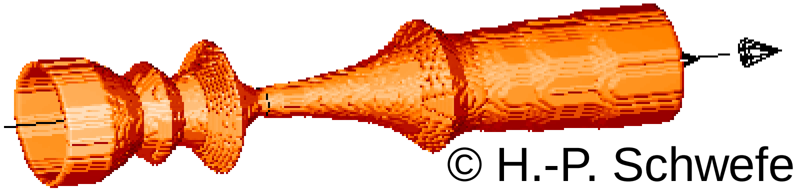
Randomized Algorithm = Stochastic Algorithm = an algorithm that uses randomness to make decisions

- first proposals in the 1940s (e.g. by N. Metropolis, J. v. Neumann, …) with applications in
    - optimization
    - numerical integration
    - generating draws from a probability distribution
- Monte Carlo algorithm: might not be correct with small probability
- Las Vegas: always correct, but might take long/exponential time

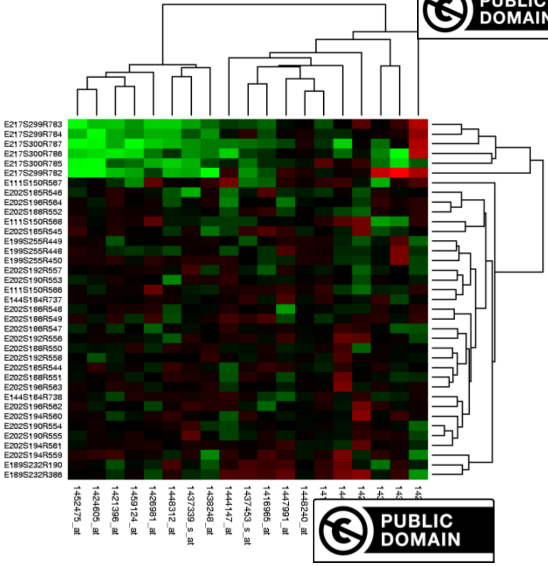
Difficult Optimization Problems are Everywhere



PUBLIC DOMAIN

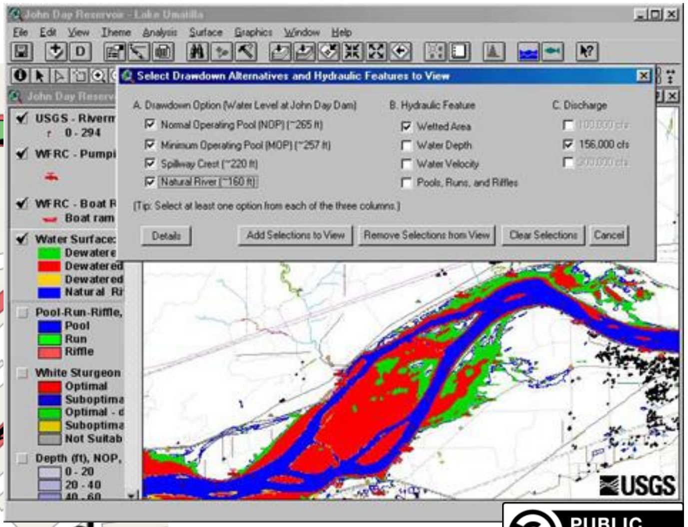


© H.-P. Schwefel



PUBLIC DOMAIN

Maly LOLek



Select Drawdown Alternatives and Hydraulic Features to View

A. Drawdown Option (Water Level at John Day Dam)
- ☑ Normal Operating Pool (NOP) (~265 ft)
- ☑ Minimum Operating Pool (MOP) (~257 ft)
- ☑ Spillway Crest (~220 ft)
- ☑ Natural River (~160 ft)

B. Hydraulic Feature
- ☑ Wetted Area
- ☐ Water Depth
- ☐ Water Velocity
- ☐ Pools, Runs, and Riffles

C. Discharge
- ☐ 100,000 cfs
- ☑ 156,000 cfs
- ☐ 200,000 cfs

(Tip: Select at least one option from each of the three columns.)

Details | Add Selections to View | Remove Selections from View | Clear Selections | Cancel

PUBLIC DOMAIN

What is Optimization?

Typically, we want to

- find solutions x which minimize $f(x)$ in the shortest time possible
  (maximization is reformulated as minimization)
- or find solutions x with as small $f(x)$ in the shortest time possible (if finding the exact optimum is not possible)

Black Box Scenario

$$x \in \Omega \rightarrow \boxed{\textbf{black box}} \rightarrow f(x) \in \mathbb{R}$$

Why are we interested in a black box scenario?

- objective function f often noisy, non-differentiable, or sometimes not even understood or available
- objective function f contains legacy or binary code, is based on numerical simulations or real-life experiments
- most likely, you will see such problems in practice...

Objective: find x with small $f(x)$ with as few function evaluations as possible

assumption: internal calculations of algo irrelevant

Motivation General Search Heuristics

- often, problem complicated and not much time available to develop a problem-specific algorithm
- general (blackbox) search heuristic: a “meta-algorithm” or “meta-heuristic” which can be applied to a large variety of problems
- search heuristics are a good choice:
  - relatively easy to implement
  - easy to adapt/change/improve
    - e.g. when the problem formulation changes in an early product design phase
    - or when slightly different problems need to be solved over time
- randomized/stochastic algorithms are a good choice because they are robust to noise

Stochastic Search Template

A stochastic blackbox search template to minimize $f\colon \Omega \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$

While happy do:

- Sample distribution $P(\boldsymbol{x}|\theta) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \Omega$
- Evaluate $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on f
- Update parameters $\theta \leftarrow F_\theta(\theta, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

Deterministic algorithms can be cast in this framework as well:

for example in $\mathbb{R}^n$: gradient descent

or local search in discrete Ω

well-known stochastic example:

Covariance Matrix Adaptation Evolution Strategy (CMA-ES):

sample distributions = multivariate Gaussian distributions

Lecture Outline Randomized Search Heuristics

Here, we touch only algorithms for discrete Ω

❶ Randomized Local Search (RLS)

❷ Evolutionary Algorithms (EAs)

❸ Compact GA: an estimation of distribution algorithm on bitstrings

Neighborhoods

For most (stochastic) search heuristics, we need to define a *neighborhood structure*

- which search points are close to each other?

Example: k-bit flip / Hamming distance k neighborhood

- search space: bitstrings of length n ($\Omega=\{0,1\}^n$)
- two search points are neighbors if their Hamming distance is k
- in other words: x and y are neighbors if we can flip exactly k bits in x to obtain y
- 0001001101 is neighbor of
  0001000101 for k=1
  0101000101 for k=2
  1101000101 for k=3

Neighborhoods II

Example: neighborhoods for permutation problems

- search space: all permutations of length n ($\Omega = S_n$)
- swap neighborhood:
  - swap two entries in the permutation
- equivalence to Hamming distance: swap distance
  - allow to swap k pairs
  - possible to sample in a given distance of k, but algorithm is not trivial
- more neighborhoods for permutations later

Randomized Local Search (RLS)

Idea behind (Randomized) Local Search:

- explore the local neighborhood of the current solution (randomly)

Pure Random Search:

- go to randomly chosen neighbor

First Improvement Local Search:

- go to first (randomly) chosen neighbor which is better

Best Improvement strategy:

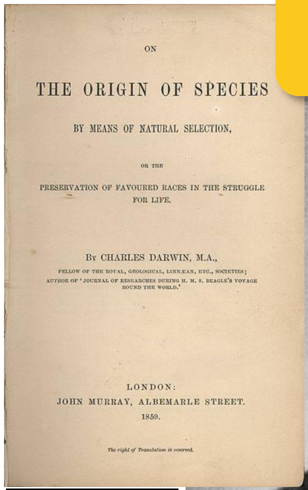
- always go to the best neighbor
- not random anymore
- computationally expensive if neighborhood large

Stochastic Optimization Algorithms

One class of (bio-inspired) stochastic optimization algorithms: Evolutionary Algorithms (EAs)

- Class of optimization algorithms originally inspired by the idea of biological evolution
- selection, mutation, recombination

1859



ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

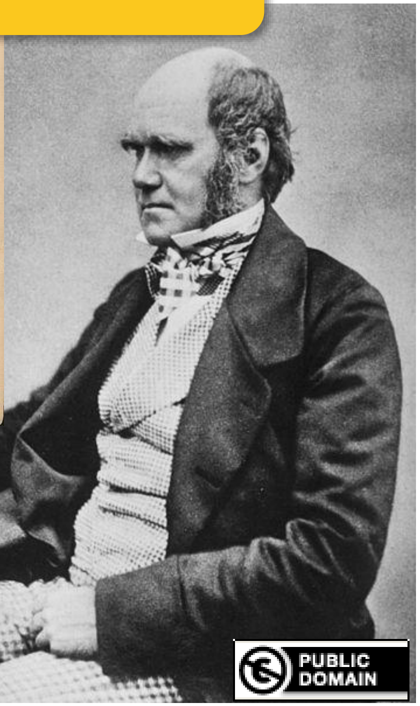
OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE
FOR LIFE.

By CHARLES DARWIN, M.A.,
FELLOW OF THE ROYAL, GEOLOGICAL, LINNÆAN, ETC., SOCIETIES;
AUTHOR OF 'JOURNAL OF RESEARCHES DURING H. M. S. BEAGLE'S VOYAGE
ROUND THE WORLD.'

LONDON:
JOHN MURRAY, ALBEMARLE STREET.
1859.

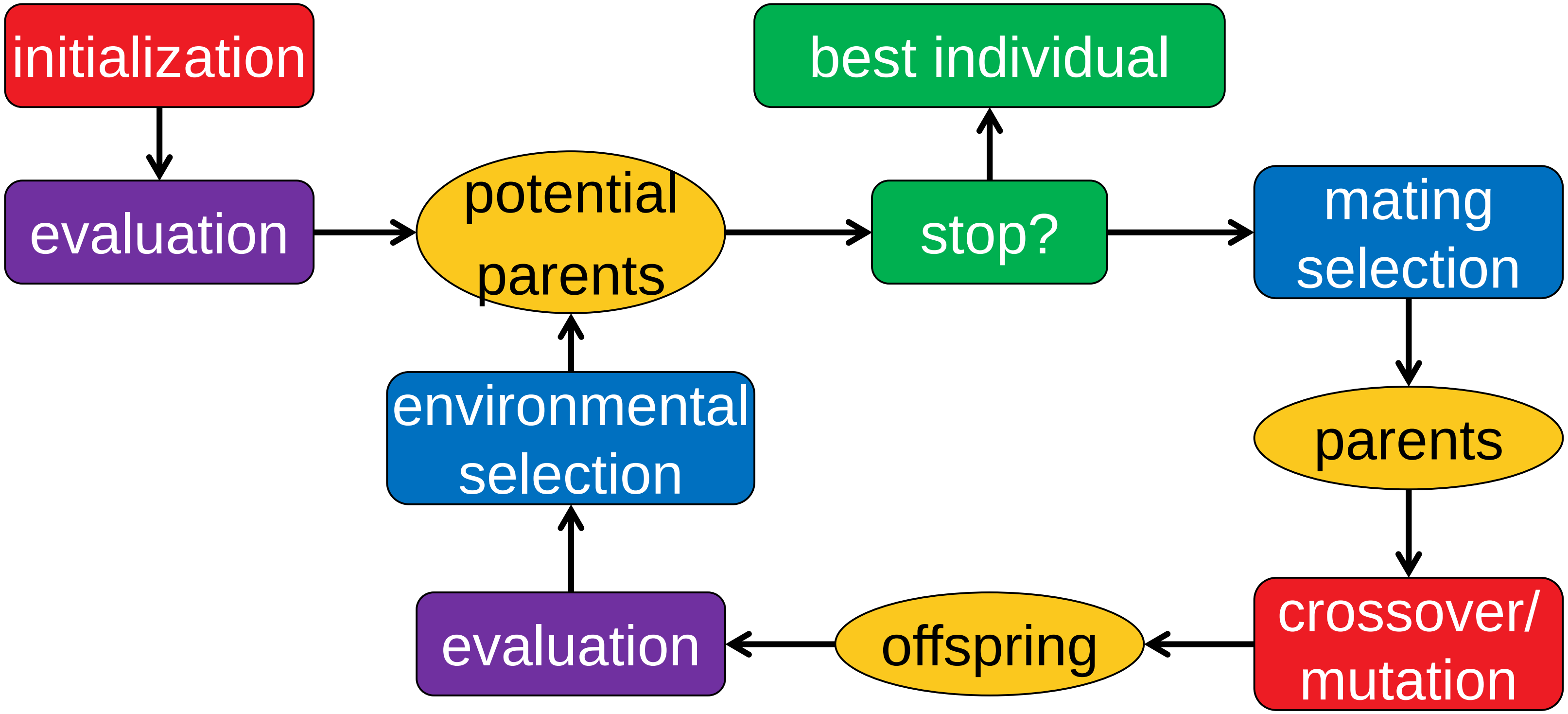
The right of Translation is reserved.



Metaphors

Classical Optimization	Evolutionary Computation
variables or parameters	variables or chromosomes
candidate solution vector of decision variables / design variables / object variables	individual, offspring, parent
set of candidate solutions	population
objective function loss function cost function error function	fitness function
iteration	generation

Generic Framework of an EA



🟥 stochastic operators

🟦 “Darwinism”

🟩 stopping criteria

Important:
representation (search space)

The Historic Roots of EAs

Genetic Algorithms (GA)

J. Holland 1975 and D. Goldberg (USA)

$$\Omega = \{0,1\}^n$$

Evolution Strategies (ES)

I. Rechenberg and H.P. Schwefel, 1965 (Berlin)

$$\Omega = \mathbb{R}^n$$

Evolutionary Programming (EP)

L.J. Fogel 1966 (USA)

Genetic Programming (GP)

J. Koza 1990 (USA)

$$\Omega = \text{space of all programs}$$

nowadays one umbrella term: evolutionary algorithms

Genotype – Phenotype mapping

The genotype – phenotype mapping

- related to the question: how to come up with a fitness ("quality") of each individual from the representation?
- related to DNA vs. actual animal (which then has a fitness)

fitness of an individual not always = f(x)

- include constraints
- include diversity
- others
- but needed: always a total order on the solutions

Examples for some EA parts

Selection

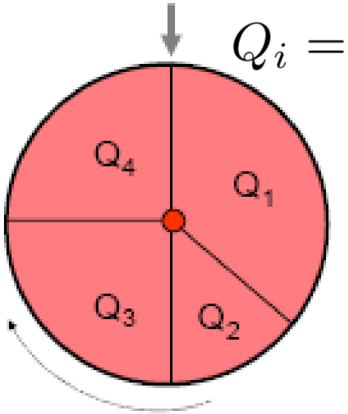
Selection is the major determinant for specifying the trade-off between exploitation and exploration

Selection is either

stochastic or deterministic

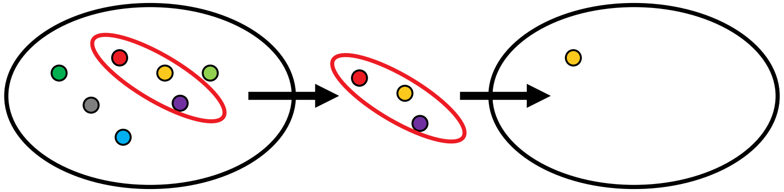
e.g. fitness proportional

$$Q_i = \frac{f(x_i)}{\sum_{j=1}^{\mu} f(x_j)}$$



Disadvantage: depends on scaling of f

e.g. via a tournament



e.g. (μ+λ), (μ,λ)

best μ from offspring *and* parents

best μ from offspring only

Mating selection (selection for variation): usually stochastic

Environmental selection (selection for survival): often deterministic

Variation Operators

Variation aims at generating new individuals on the basis of those individuals selected for mating

Variation = Mutation and Recombination/Crossover

mutation: *mut:* $\Omega \to \Omega$

recombination: *recomb:* $\Omega^r \to \Omega^s$ where $r \geq 2$ and $s \geq 1$

- choice always depends on the problem and the chosen representation
- however, there are some operators that are applicable to a wide range of problems and tailored to standard representations such as vectors, permutations, trees, etc.

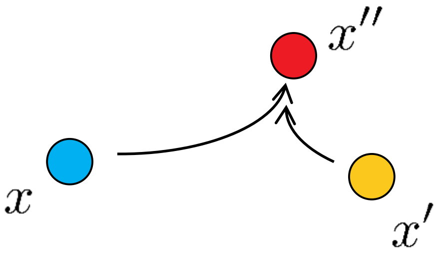
Variation Operators: Guidelines

Two desirable properties for mutation operators:

- every solution can be generation from every other with a probability greater than 0 (“exhaustiveness”)
- $d(x, x') < d(x, x'') => Prob(\mathrm{mut}(x) = x') > Prob(\mathrm{mut}(x) = x'')$ (“locality”)

Desirable property of recombination operators (“in-between-ness”):

$$x'' = \ \mathrm{recomb}(x, x') \Rightarrow d(x'', x) \leq d(x, x') \wedge d(x'', x') \leq d(x, x')$$



Examples of Mutation Operators on $\{0,1\}^n$

1-bit flip mutation

- flip a randomly chosen bit (from 1 to 0 or vice versa)

k-bit flip mutation

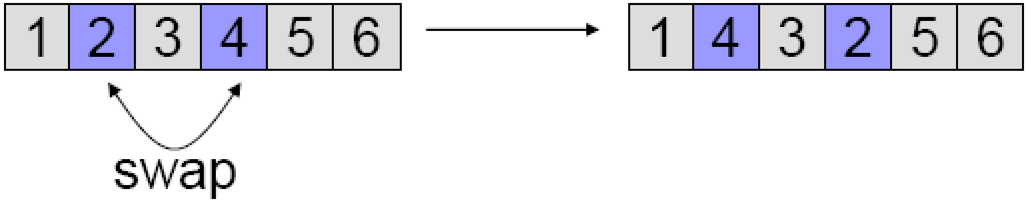
- choose k (different) bits uniformly at random
- flip each of those bits (from 1 to 0 or vice versa)

Standard bitflip mutation

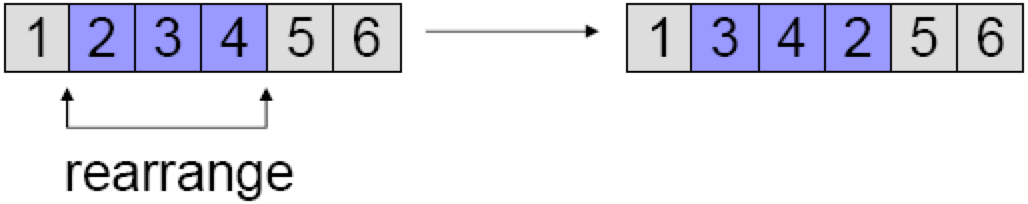
- flip each bit independently with probability 1/n
- expected number of bits changed: 1
- but also: $\lim_{n \to \pm\infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.367879$ i.e. no bit flipped with constant probability

Examples of Mutation Operators on Permutations

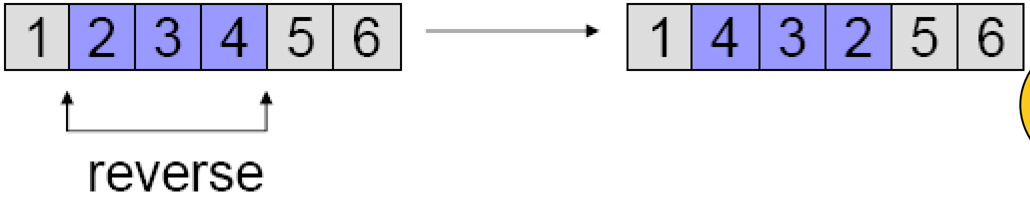
Swap:



Scramble:

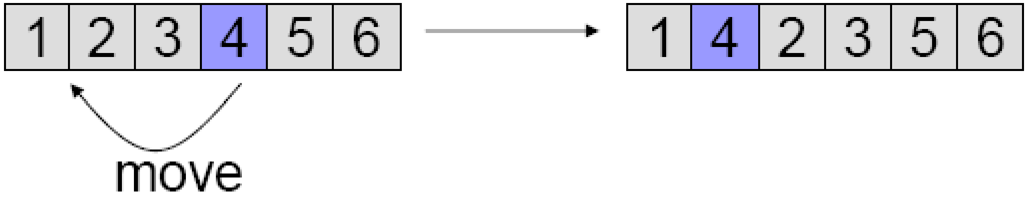


Invert:



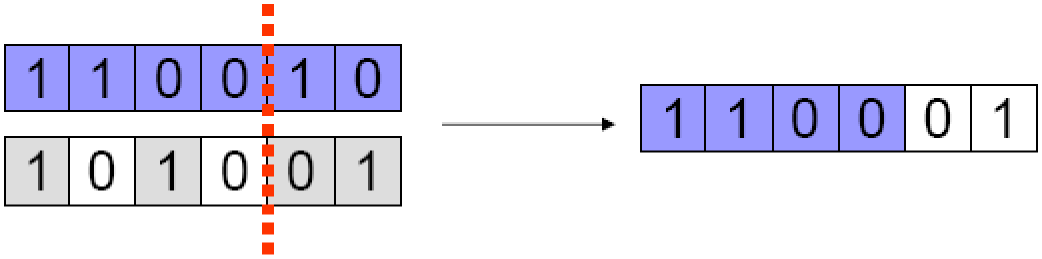
also known as 2-opt

Insert:

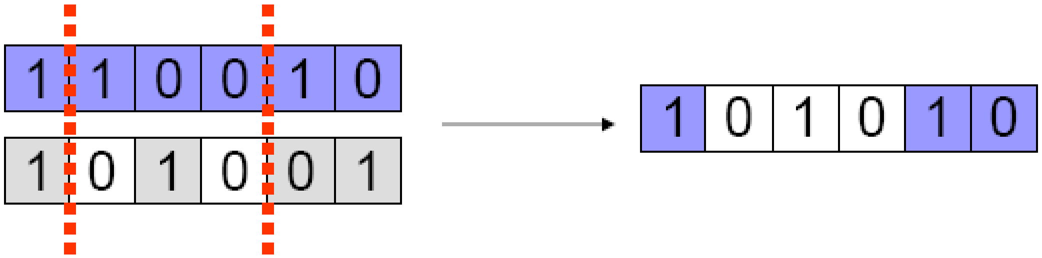


Examples of Recombination Operators on $\{0,1\}^n$

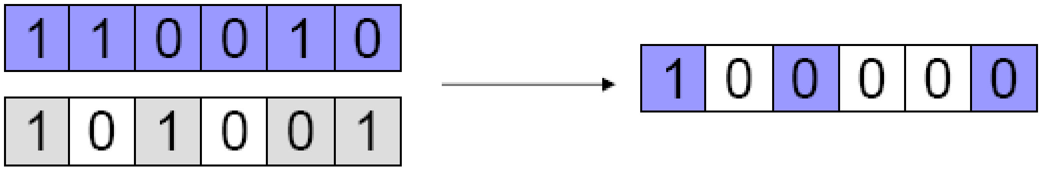
1-point crossover



n-point crossover



uniform crossover



choose each bit independently from one parent or another

A Canonical Genetic Algorithm

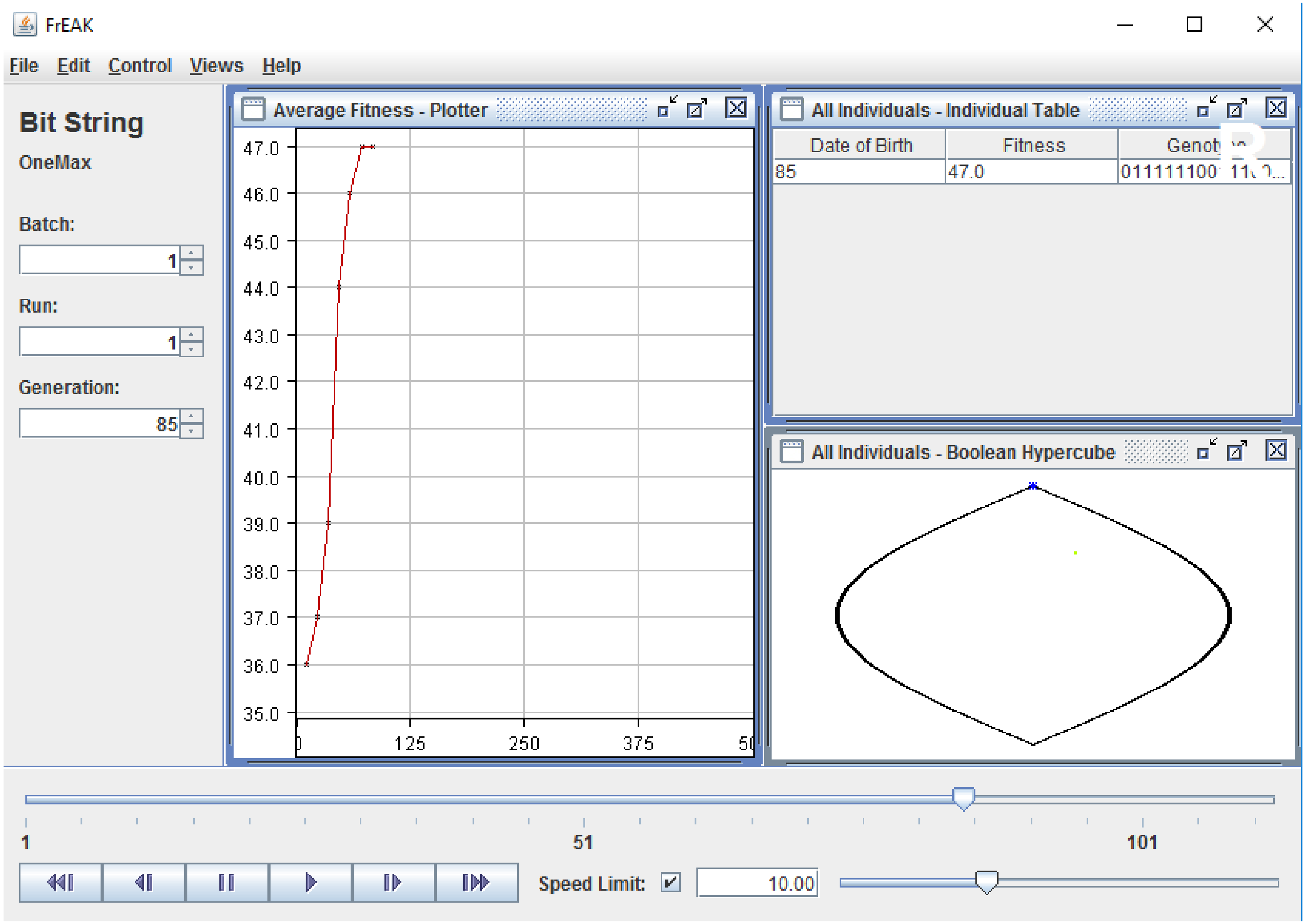
- binary search space, maximization
- uniform initialization
- generational cycle:
    - evaluation of solutions
    - mating selection (e.g. roulette wheel)
    - crossover (e.g. 1-point)
    - environmental selection (e.g. plus-selection)

You may ask: how does this fit
into the stochastic search template?
it does: population contained in state θ,
but update function difficult to write down

FrEAK

If you want to play around a bit with these algorithms:

- https://sourceforge.net/projects/freak427/



Bit String

OneMax

Batch: 1

Run: 1

Generation: 85

Date of Birth	Fitness	Genotype
85	47.0	011111100 11...

Speed Limit: ☑ 10.00

Estimation of Distribution Algorithms

- Estimation of Distribution Algorithms (EDAs) fit more obviously into the search template
- here, example of the compact Genetic Algorithm (cGA)
    - search space: $\Omega = \{0,1\}^n$
    - probability distribution: Bernoulli
        - store for each bit a probability p_i to sample a 1
        - sample bit i with probability p_i to 1 and with $(1 - p_i)$ to 0

The Compact GA

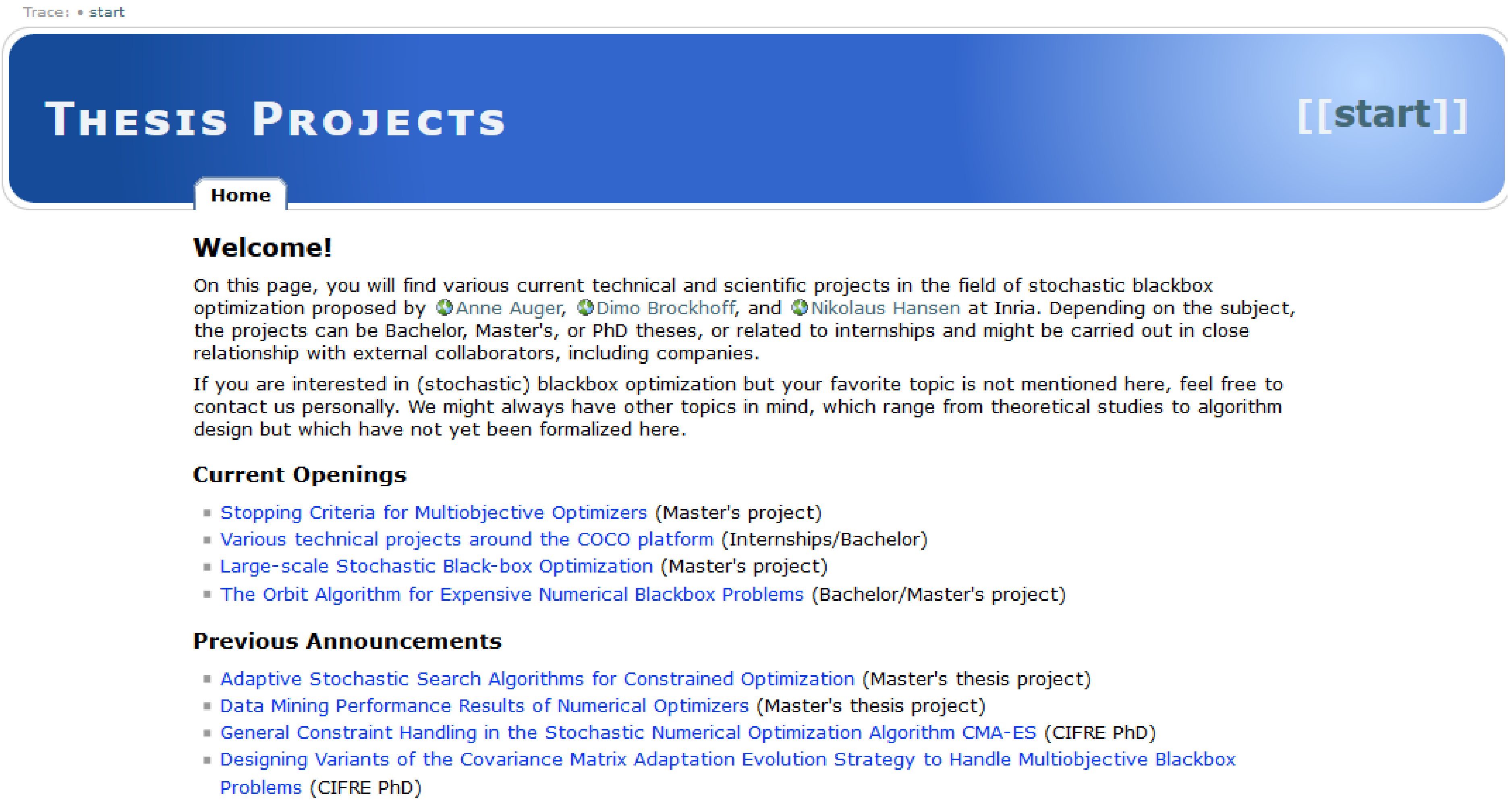
Parameters: number of variables n, learning rate K (typically $= n$)

```
Init:
    $p = \left(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}\right) \in [0,1]^n$ # probabilities to sample new solutions
While happy:
    create $S = (s_1, \ldots, s_n)$ by sampling each $s_i$ with probability $p_i$
    create $S' = (s'_1, \ldots, s'_n)$ by sampling each $s'_i$ with probability $p_i$
    evaluate $S$ and $S'$ on $f$
    if $f(S) > f(S')$:    # make sure that S is the better solution
        $S, S' \leftarrow S', S$
    # update p parameter:
    for $i \in \{1, \ldots, n\}$:
        $p_i \leftarrow \min\{\max\{p_i + (s_i - s'_i)/K, 1/n\}, 1 - 1/n\}$
return $S$
```

Potential Master's/PhD thesis projects

Potential Research Topics for Master's/PhD Theses

`http://randopt.gforge.inria.fr/thesisprojects/`



Trace: • start

THESIS PROJECTS

[[start]]

Welcome!

On this page, you will find various current technical and scientific projects in the field of stochastic blackbox optimization proposed by Anne Auger, Dimo Brockhoff, and Nikolaus Hansen at Inria. Depending on the subject, the projects can be Bachelor, Master's, or PhD theses, or related to internships and might be carried out in close relationship with external collaborators, including companies.

If you are interested in (stochastic) blackbox optimization but your favorite topic is not mentioned here, feel free to contact us personally. We might always have other topics in mind, which range from theoretical studies to algorithm design but which have not yet been formalized here.

Current Openings

- Stopping Criteria for Multiobjective Optimizers (Master's project)
- Various technical projects around the COCO platform (Internships/Bachelor)
- Large-scale Stochastic Black-box Optimization (Master's project)
- The Orbit Algorithm for Expensive Numerical Blackbox Problems (Bachelor/Master's project)

Previous Announcements

- Adaptive Stochastic Search Algorithms for Constrained Optimization (Master's thesis project)
- Data Mining Performance Results of Numerical Optimizers (Master's thesis project)
- General Constraint Handling in the Stochastic Numerical Optimization Algorithm CMA-ES (CIFRE PhD)
- Designing Variants of the Covariance Matrix Adaptation Evolution Strategy to Handle Multiobjective Blackbox Problems (CIFRE PhD)

Conclusions

- EAs are generic algorithms (randomized search heuristics, meta-heuristics, ...) for black box optimization

no or almost no assumptions on the objective function

- They are typically less efficient than problem-specific (exact) algorithms in discrete domain (in terms of #funevals)

but competitive in the continuous case

- Allow for an easy and rapid implementation and therefore to find good solutions fast

easy (recommended!) to incorporate problem-specific knowledge to improve the algorithm

Conclusions

I hope it became clear...

...that heuristics is what we typically can afford in practice (no guarantees and no proofs)

...what are the main ideas behind evolutionary algorithms

...and that evolutionary algorithms and genetic algorithms are no synonyms