Evolutionary Algorithms and Parameter Control

Carsten Witt

Technical University of Denmark, Kgs. Lyngby, Denmark

September 29, 2021

Why Evolutionary Algorithms?

White-box vs. black-box optimization

Task: find optimum of a function $f : S \to \mathbb{R}$

If we have explicit representation of f, including derivatives, we can apply classical mathematical optimization techniques (white-box scenario).

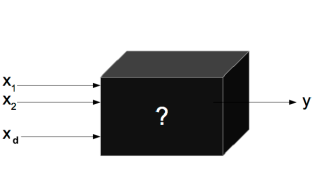
Why Evolutionary Algorithms?

White-box vs. black-box optimization

Task: find optimum of a function $f: S \to \mathbb{R}$

If we have explicit representation of f, including derivatives, we can apply classical mathematical optimization techniques (white-box scenario).

If f is only given implicitly (e. g., outcome of an experiment), we are in a black-box scenario where only sampling f reveals information.



Further possible challenges: uncertainty, e. g. noise and dynamic functions

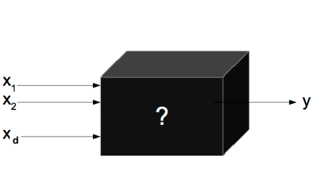
Why Evolutionary Algorithms?

White-box vs. black-box optimization

Task: find optimum of a function $f: S \to \mathbb{R}$

If we have explicit representation of f, including derivatives, we can apply classical mathematical optimization techniques (white-box scenario).

If f is only given implicitly (e. g., outcome of an experiment), we are in a black-box scenario where only sampling f reveals information.



Further possible challenges: uncertainty, e. g. noise and dynamic functions

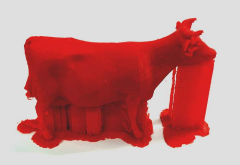
Evolutionary Algorithms (EAs) are well-established black-box optimization techniques with numerous applications in engineering.

More general umbrella term: Randomized Optimization Heuristics

Applications of Evolutionary Algorithms

- Complex optimization problems, e. g., planning the layout of a wind farm (Univ. Adelaide, AUS), minimizing waste in 3D printing, ...

(Erik Wilde, CC BY-SA)

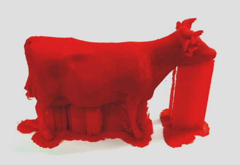


(DOI: 10.1145/3071178.3071310)

Applications of Evolutionary Algorithms

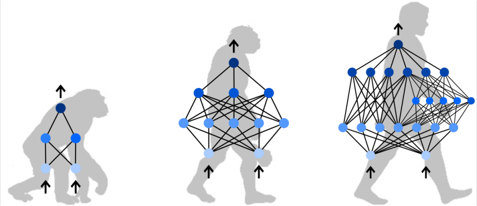
- Complex optimization problems, e. g., planning the layout of a wind farm (Univ. Adelaide, AUS), minimizing waste in 3D printing, ...

(Erik Wilde, CC BY-SA)



(DOI: 10.1145/3071178.3071310)

- Optimizing the topology and weights of a neural network
  $\rightarrow$ **Neuroevolution**, extremely popular these days

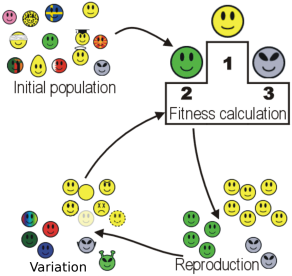


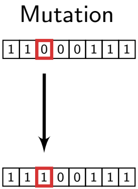
(https://github.com/PaulPauls/Primer_on_Neuroevolution_Blog_Post/)

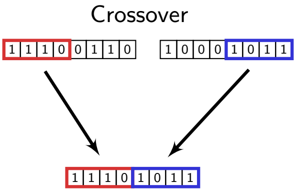
"Evolution is the new deep learning" (R. Miikkulainen, Cognizant Tech.)

What are Evolutionary Algorithms?

Optimization loop inspired by evolution theory ("survival of the fittest")







Crucial design components

- Representation: what is the search space S? $\mathbb{R}^n$, $\{0,1\}^n$, Σ_n, ...?
- Population size: how many solutions from S to maintain in parallel?
- Selection: which solutions should undergo variation?
- Variation: how to mutate solutions? How to combine two solutions to a new one (crossover)?
- ...

A Typical Evolutionary Algorithm

Pseudocode of "Generational EA"

Initialize population P_0 of size μ. Set $t \leftarrow 0$.
while stopping criterion not fulfilled **do**
  for $i \leftarrow 1, \ldots, \mu$ **do**
    Choose two individuals x and y from P_t by applying selection.
    Create z by applying crossover to x and y.
    Create z' by applying mutation to z.
    Add z' to P_{t+1}. (assumption: P_{t+1} initially empty)
  end for
  $t \leftarrow t + 1$.
end while

A Typical Evolutionary Algorithm

Pseudocode of “Generational EA”

```
Initialize population P_0 of size μ. Set t ← 0.
while stopping criterion not fulfilled do
    for i ← 1, ..., μ do
        Choose two individuals x and y from P_t by applying selection.
        Create z by applying crossover to x and y.
        Create z' by applying mutation to z.
        Add z' to P_{t+1}.    (assumption: P_{t+1} initially empty)
    end for
    t ← t + 1.
end while
```

Scheme is typical but does not cover all variants of EAs. Not considered:

- varying population size
- mutation and crossover not performed in every “generation” (→ parameters for mutation and crossover probability)
- ...

Immense empirical knowledge on parameter choices for EAs available.
Want to support parameter choice using theory: runtime analysis.

A Very Simple Scenario

Algorithm: (1+1) EA for maximization of $f\colon \{0,1\}^n \to \mathbb{R}$

$t := 0$. Choose u. a. r. $x_0 \in \{0,1\}^n$.
repeat
 Create x' by flipping each bit in x_t independ. w. prob. p (often $\frac{1}{n}$).
 $x_{t+1} := x'$ if $f(x') \geq f(x_t)$, and $x_{t+1} := x_t$ otherwise.
 $t := t + 1$.
until some stopping criterion is fulfilled.

Runtime of (1+1) EA: number of iterations t until optimum found.

A Very Simple Scenario

Algorithm: (1+1) EA for maximization of $f\colon \{0,1\}^n \to \mathbb{R}$

$t := 0$. Choose u. a. r. $x_0 \in \{0,1\}^n$.
repeat
  Create x' by flipping each bit in x_t independ. w. prob. p (often $\frac{1}{n}$).
  $x_{t+1} := x'$ if $f(x') \geq f(x_t)$, and $x_{t+1} := x_t$ otherwise.
  $t := t + 1$.
until some stopping criterion is fulfilled.

Runtime of (1+1) EA: number of iterations t until optimum found.

A very simple problem

$f(x_1, \dots, x_n) = w_1 x_1 + \dots + w_n x_n$, where $w_i \in \mathbb{R}$ (linear function)

Static parameter control (= optimization)

Theorem (W., 2013): $p = 1/n$ minimizes expected runtime of (1+1) EA on any linear function $\to en \ln n + O(n)$.

Dynamic Parameter Control

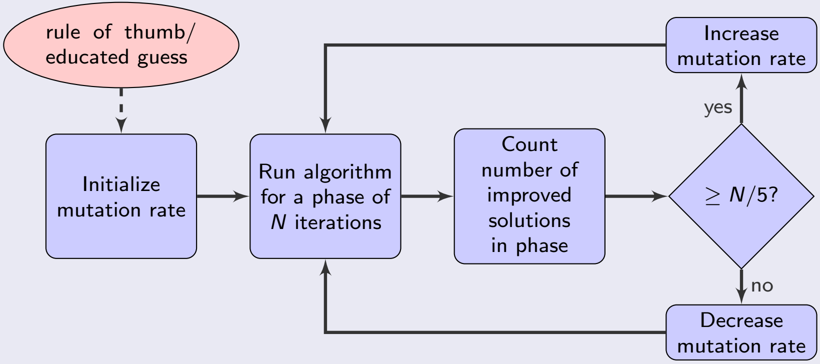
Fixing mutation prob. at $1/n$ throughout the run is a compromise.
If far away from optimum, progress is easy $\rightarrow$ higher prob. promising.

Dynamic Parameter Control

Fixing mutation prob. at $1/n$ throughout the run is a compromise.
If far away from optimum, progress is easy $\rightarrow$ higher prob. promising.

Idea: learn promising mutation probability on the fly based on successes.

Example 1: 1/5-rule (Rechenberg, 1973)



EAs with dynamic parameter control are also called self-adjusting EAs.

Dynamic Parameter Control for Mutation Strength

Doerr et al., 2016: learn the number $k \in \{1, \dots, r\}$ of bits flipped by the (1+1) EA like in a multi-armed bandit problem.

Ex. 2: self-adjusting (1+1) EA learning parameter confidences

- Assign each $k \in \{1, \dots, r\}$ a confidence c_k.
- With probability $1 - \varepsilon$, select k with highest confidence and with probability ε, choose a random $k \in \{1, \dots, r\}$. Flip k bits u. a. r.
- Update confidences based on decay parameter δ:

$$c_i(t) := \frac{\sum_{s=1}^{t} \mathbb{1}_{k(s)=i}(1-\delta)^{t-s}(f(x_s) - f(x_{s-1}))}{\sum_{s=1}^{t} \mathbb{1}_{k(s)=i}(1-\delta)^{t-s}}$$

- Rest of algorithm like (1+1) EA.

Dynamic Parameter Control for Mutation Strength

Doerr et al., 2016: learn the number $k \in \{1, \ldots, r\}$ of bits flipped by the (1+1) EA like in a multi-armed bandit problem.

Ex. 2: self-adjusting (1+1) EA learning parameter confidences

- Assign each $k \in \{1, \ldots, r\}$ a confidence c_k.
- With probability $1 - \varepsilon$, select k with highest confidence and with probability ε, choose a random $k \in \{1, \ldots, r\}$. Flip k bits u. a. r.
- Update confidences based on decay parameter δ:

$$c_i(t) := \frac{\sum_{s=1}^{t} \mathbb{1}_{k(s)=i}(1-\delta)^{t-s}(f(x_s) - f(x_{s-1}))}{\sum_{s=1}^{t} \mathbb{1}_{k(s)=i}(1-\delta)^{t-s}}$$

- Rest of algorithm like (1+1) EA.

Up to lower-order terms, expected optimization time on OneMax benchmark is as good as if an oracle always told the optimal k.

Runtime $\leq n \ln n - 0.25n \rightarrow$ speed-up over static case roughly $0.14n$.

Dynamic Parameter Control for Mutation Strength

Doerr et al., 2016: learn the number $k \in \{1, \ldots, r\}$ of bits flipped by the (1+1) EA like in a multi-armed bandit problem.

Ex. 2: self-adjusting (1+1) EA learning parameter confidences

- Assign each $k \in \{1, \ldots, r\}$ a confidence c_k.
- With probability $1 - \varepsilon$, select k with highest confidence and with probability ε, choose a random $k \in \{1, \ldots, r\}$. Flip k bits u. a. r.
- Update confidences based on decay parameter δ:

$$c_i(t) := \frac{\sum_{s=1}^{t} \mathbb{1}_{k(s)=i}(1-\delta)^{t-s}(f(x_s) - f(x_{s-1}))}{\sum_{s=1}^{t} \mathbb{1}_{k(s)=i}(1-\delta)^{t-s}}$$

- Rest of algorithm like (1+1) EA.

Up to lower-order terms, expected optimization time on OneMax benchmark is as good as if an oracle always told the optimal k.

Runtime $\leq n \ln n - 0.25n \rightarrow$ speed-up over static case roughly $0.14n$.

Empirical promising performance on MST instances.

Asymptotic Speed-ups Through Parameter Control

(1+1) EA too simple to see pronounced effect of varying mutation rate.

Parameter control most promising together with populations.

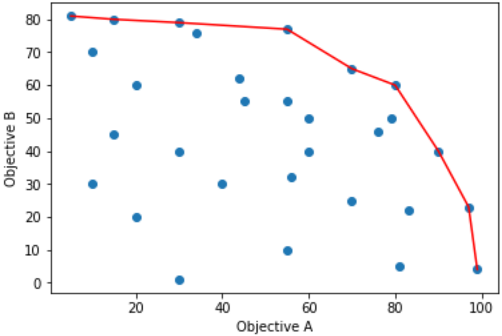
Asymptotic Speed-ups Through Parameter Control

(1+1) EA too simple to see pronounced effect of varying mutation rate.

Parameter control most promising together with populations.

Populations:

- can diversify the search (cover different areas of search space)
- crucial in multi-objective optimization



Consider an example of speed-up by combining populations, mutation, and parameter control.

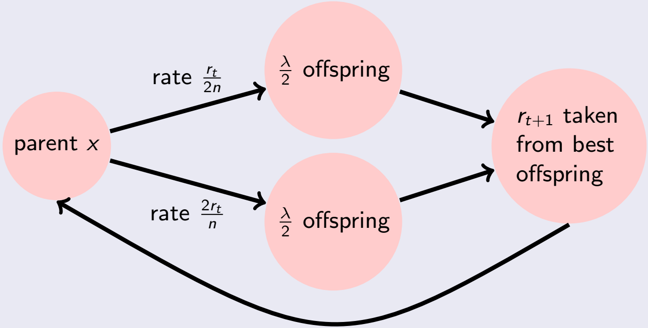
Asymptotic Speed-ups Through Parameter Control

Ex. 3: self-adjusting (1+λ) EA using two rates for offspring

Select x uniformly at random from $\{0,1\}^n$ and set $r_0 \leftarrow 2$.
repeat
  for $i \leftarrow 1, \ldots, \lambda$ **do**
    Create y_i by flipping each bit in a copy of x independently with probability $\frac{r_t}{2n}$ if $i \leq \lambda/2$ and with probability $\frac{2r_t}{n}$ otherwise.
  end for
  $x^* \leftarrow \arg\max_{y_i} f(y_i)$ (breaking ties randomly).
  if $f(x^*) \geq f(x)$ **then**
    $x \leftarrow x^*$.
  end if
  Replace r_t with the mutation rate that x^* has been created with.
  Replace r_t with $\min\{\max\{2, r_t\}, n/4\}$; $t \leftarrow t+1$
until some stopping criterion is fulfilled.

Asymptotic Speed-ups Through Parameter Control

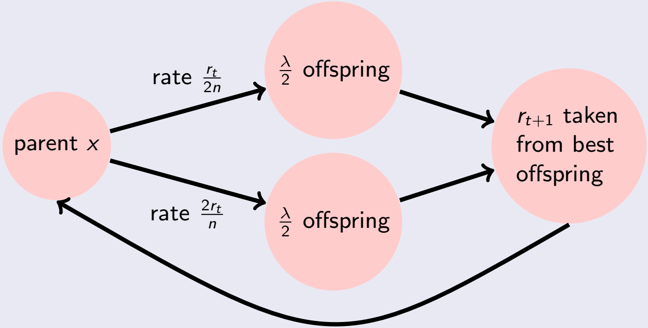
Ex. 3: self-adjusting (1+λ) EA using two rates for offspring



$r_0 = 2$; $r_t \in \{2, \ldots, n/4\}$

Asymptotic Speed-ups Through Parameter Control

Ex. 3: self-adjusting (1+λ) EA using two rates for offspring



$r_0 = 2$; $r_t \in \{2, \ldots, n/4\}$

Theorem (Doerr et al., 2018): self-adjusting (1+λ) EA asymptotically faster on OneMax benchmark (time $O(n/\log\lambda)$) than static variant ($O(n\log\log\lambda/\log\lambda)$)

Param. Control in Discrete Optimization: Recent Research

Highlights

- Self-adjusting $(1+(\lambda,\lambda))$ GA with 1/5-rule optimizes $\textsc{OneMax}$ in $O(n)$, beating any static parameter setting [Doerr and Doerr, 2015]; similar speedups on random 3-CNF formulas [Buzdalov and Doerr, 2017]
- Self-adjusting $(1+1)$ EA with stagnation detection much more efficient than static variant at escaping local optima [Rajabi and W., 2020]
- With "$\frac{1}{5}$-rule" best possible runtime (up to $o(1)$) on $\textsc{LeadingOnes}$ benchmark [Doerr et al., 2019]

Param. Control in Discrete Optimization: Recent Research

Highlights

- Self-adjusting $(1+(\lambda, \lambda))$ GA with 1/5-rule optimizes OneMax in $O(n)$, beating any static parameter setting [Doerr and Doerr, 2015]; similar speedups on random 3-CNF formulas [Buzdalov and Doerr, 2017]
- Self-adjusting $(1+1)$ EA with stagnation detection much more efficient than static variant at escaping local optima [Rajabi and W., 2020]
- With "$\frac{1}{5}$-rule" best possible runtime (up to $o(1)$) on LeadingOnes benchmark [Doerr et al., 2019]

Research questions

- Great variety of mechanisms for self-adaptation. Compare these empirically and theoretically. Suggestions for new mechanisms?
- Advance and unify the toolbox for the analysis (so far many ad-hoc "drift" theorems and hard-to-compare techniques)
- Analyses on combinatorial optimization problems (on graphs etc.)
- Relation to hyperheuristics?

Param. Control in Discrete Optimization: Recent Research

Highlights

- Self-adjusting $(1+(\lambda, \lambda))$ GA with 1/5-rule optimizes OneMax in $O(n)$, beating any static parameter setting [Doerr and Doerr, 2015]; similar speedups on random 3-CNF formulas [Buzdalov and Doerr, 2017]
- Self-adjusting $(1+1)$ EA with stagnation detection much more efficient than static variant at escaping local optima [Rajabi and W., 2020]
- With "$\frac{1}{5}$-rule" best possible runtime (up to $o(1)$) on LeadingOnes benchmark [Doerr et al., 2019]

Research questions

- Great variety of mechanisms for self-adaptation. Compare these empirically and theoretically. Suggestions for new mechanisms?
- Advance and unify the toolbox for the analysis (so far many ad-hoc "drift" theorems and hard-to-compare techniques)
- Analyses on combinatorial optimization problems (on graphs etc.)
- Relation to hyperheuristics?

Thank you!