Optimizing Multi-Valued Decision Variables with Evolutionary Algorithms

Benjamin Doerr (École Polytechnique)

joint work with Carola Doerr (Paris 6) and Timo Kötzing (HPI Potsdam)

This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Programme Gaspard Monge en Optimisation et Recherche Opérationnelle.

Context: Rigorous Theoretical Research on Evolutionary Algorithms

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

- Young field: Started in the 1990s, greatly developed by Ingo Wegener

- Goals:
  - explain why evolutionary algorithms (EAs) often work surprisingly well
  - give well-founded advice how to choose algorithms, parameters, etc.

- Main difference to the classic theory of algorithms field:
  - we try to understand how to solve problems with *existing* algorithms rather than inventing new algorithms
  - to understand these algorithms, we also analyze how they perform on test/benchmark problems designed only for this purpose

Overview of this Work

- Observation: The vast majority of the evolutionary computation literature (applied and theory) concentrates on binary decision variables
  - advantage: you can use well-established mutation/crossover operators
- Problem: Many optimization problems contain decision variables taking more than two values, these have to be transformed into the bit-string world
- Target of this work: Gain some problem-independent, mathematically proven insight in how to deal directly with multi-valued variables!
  - Define a multi-valued analogue of the classic ONEMAX test-problem
  - Suggest several natural extensions of the classic bit-mutation operator
  - Conduct a mathematical (tight) runtime analysis for these
- Main insights:
  - not a single best static mutation operator
  - self-adjusting mutation strength beats all static mutation operators

Higher-Order OneMax Functions

the fruit fly of EA theory

- Classic OneMax test-function: $OM: \{0,1\}^n \to \mathbb{R}; x \mapsto x_1 + \cdots + x_n$
  - Unique minimum at $(0, \dots, 0)$ [we minimize in this talk]
  - Benchmark problem for good fitness-distance correlation
  - To avoid cheating, one better regards the class of all OneMax functions
    - $OM_z: \{0,1\}^n \mapsto \mathbb{R}; x \mapsto |x_1 - z_1| + \cdots + |x_n - z_n|, \ z \in \{0,1\}^n$
    - a reasonable algorithm should have the same performance on all these isomorphic optimization problems

- OneMax with multi-values decision variables:
  - $OM_z: \{0, \dots, r-1\}^n \to \mathbb{R}; x \mapsto |x_1 - z_1| + \cdots + |x_n - z_n|$
  - many non-isomorphic fitness landscapes
  - alternative: replace $|\cdot|$ by the cyclic distance on $\{0, \dots, r-1\}$

Definition: Higher-Order ONEMAXs

- Definition: A function $f: \{0,..,r-1\}^n \to \mathbb{R}$ is called OneMax function of order r with optimum $z \in \{0,...r-1\}^n$ if for all $x \in \{0,...,r-1\}^n$ we have

$$f(x) = d(x_1, z_1) + \cdots + d(x_n, z_n),$$

where d is either

  - the usual distance $d(a,b) := |a-b|$ in $\mathbb{R}$, or
  - the distance in the cycle $[0,r) = \mathbb{R}/\mathbb{Z}\_n$ defined by
$$d(a,b) := \min\{a-(b-r), |a-b|, (b+r)-a\}$$

- All results we have are valid for the minimization of all these functions.

- For this talk, let's make our life easy (not really) and only look at the minimization of the function

$$f: \{0,...,r-1\}^n \to \mathbb{R}; x \mapsto x_1 + \cdots + x_n$$

Next Question: What is the Right Way of Doing Mutation?

- Classic mutation operators for n binary variables:
  - one-bit flips: choose $i \in \{1, \dots, n\}$ uniformly at random and flip the ith bit
  - standard bit mutation: flip each bit independently with probability $1/n$

- Our higher-order analogues:
  - We keep the principles "modify a random decision variable" and "for each variable, decide independently with probability $1/n$ to modify it"
  - However, there are different ways to modify a variable x_i now
    - change x_i to a random value different from x_i
    - change x_i to a random value in $\{x_i - 1, x_i + 1\}$ (boundaries?)
    - change x_i to a random different value, but such that $x_i \pm k$ appears with probability proportional to $1/k$ (boundaries?)
  - we call these different step sizes (random, unit, harmonic)

First Result

- Theorem: Consider a run of the (1+1) EA on an arbitrary OneMax function of order r. Then the optimization time (number of fitness evaluations until an optimum is found) T satisfies the following.
  1. Random step size (change x_i to a random different value):
     - $E[T] = \Theta(nr \log n)$
  2. Unit step size (change x_i to a random value in $\{x_i - 1, x_i + 1\}$):
     - $E[T] = \Theta(nr + n \log n)$
  3. Harmonic step size (change x_i to a random different value, but such that $x_i \pm k$ appears with probability proportional to $1/k$):
     - $E[T] = \Theta(n \log^2 r + n \log r \log n)$

(1+1) evolutionary algorithm for minimizing a function $f: \{0, \dots, r-1\}^n \to \mathbb{R}$:

1. choose $x \in \{0, \dots, r-1\}^n$ uniformly at random
2. repeat
3. mutation: generate y from x by changing (with the given step size) a random entry *or* each entry of x independently with probability $1/n$
4. selection: if $f(y) \leq f(x)$ then $x := y$
5. until *terminate*

First Result

- Theorem: Consider a run of the (1+1) EA on an arbitrary OneMax function of order r. Then the optimization time (number of fitness evaluations until an optimum is found) T satisfies the following.
  1. Random step size (change x_i to a random different value):
     - $E[T] = \Theta(nr \log n)$
  2. Unit step size (change x_i to a random value in $\{x_i - 1, x_i + 1\}$):
     - $E[T] = \Theta(nr + n \log n)$
  3. Harmonic step size (change x_i to a random different value, but such that $x_i \pm k$ appears with probability proportional to $1/k$):
     - $E[T] = \Theta(n \log^2 r + n \log r \log n)$
- Notes: (1.) is (only) slightly worse than (2.)
  - (2.) and (3.) are incomparable: (2.) is better when $r = O(\log n \cdot \log \log n)$
- Next: Obtain $E[T] = \Theta(n \log r + n \log n)$ with a self-adjusting step size

Self-Adjusting Step-Size

(1+1) EA with one-variable changes and self-adjusting step size

- Choose $x \in \{0,1\}^n$ uniformly at random
- Choose $s \in \{1, \dots, r/4\}^n$ uniformly at random
- repeat
  - $y := x$
  - choose $i \in \{1, \dots, n\}$ uniformly at random
  - choose $y_i \in \{y_i - s_i, y_i + s_i\}$ uniformly at random
  - if $f(y) < f(x)$ then $s_i := \min\{s_i * 1.8, r/4\}$ else $s_i := \max\{s_i * 0.9\ , 1\}$
  - if $f(y) \leq f(x)$ then $x := y$
- until *terminate*

step size s_i for each variable x_i

step size adaption: increase s_i when making true progress, otherwise decrease mildly

- Theorem: This self-adjusting EA optimizes any r-valued OneMax function in expected time $E[T] = \Theta(n \log r + n \log n)$. This is asymptotically best-possible among all dynamic step sizes.

At Least One Proof:

- To give some idea of the methods we use, let us prove the result for unit mutation strengths:

- Theorem: The following algorithms finds the optimum of
$$f: \{0, \dots, r-1\}^n \to \mathbb{R}; x \mapsto x_1 + \cdots + x_n$$
in an expected number of $E[T] = O(nr + n \log n)$ iterations.

(1+1) EA, minimizing $f: \{0, \dots, r-1\}^n \to \mathbb{R}$, using unit mutation strength:

1. choose $x \in \{0, \dots, r-1\}^n$ uniformly at random
2. repeat
3. $\quad y := x$
4. $\quad$ for $i = 1, \dots, n$ do
$\quad\quad$ with probability $1/n$ do choose $y_i \in \{y_i - 1, y_1 + 1\}$ randomly
5. selection: if $f(y) \leq f(x)$ then $x := y$
6. until *terminate*

Difficulties

- Low correlation between the objective value and the runtime:
  - The two solutions $x^1 = (1, \dots, 1)$ and $x^2 = (\log n, 0, \dots, 0)$ have very different objective values (n vs. $\log n$), but from both we find the optimum in $\Theta(n \log n)$ iterations.
- The mutation operator allows progress in some variable at the cost of others: If the parent solution is $x = (20,8,2,0, \dots, 0)$ and the offspring is $y = (21,7,1,1, \dots, 0)$, then this mutation is accepted (we continue with y).
  - difficult to argue that the algorithm makes progress

(1+1) EA, minimizing $f: \{0, \dots, r-1\}^n \to \mathbb{R}$, using unit mutation strength:

1. choose $x \in \{0, \dots, r-1\}^n$ uniformly at random
2. repeat
3. $y := x$
4. for $i = 1, \dots, n$ do
   with probability $1/n$ do choose $y_i \in \{y_i - 1, y_1 + 1\}$ randomly
5. selection: if $f(y) \leq f(x)$ then $x := y$
6. until *terminate*

Solution: Potential Function

- Key proof idea: use $g(x) := \sum_{i=1}^{n} (1.1)^{x_i} - n$ as measure for the quality of the search point x ($g(x) = 0$ means that x is the optimum)

- Analyze the expected progress with respect to g: Let x' be the current solution after one iteration. Then
$$E[g(x) - g(x')] \geq C\, g(x)/n$$
for some constant C (this is the difficult part of the proof)

- Multiplicative drift theorem: If you have a random process $X_0, X_1, \ldots$ in the non-negative integers with $E[X_{t+1} | X_t] \leq (1-\delta) X_t$, then the first hitting time T of zero satisfies $E[T] \leq \frac{1+\ln X_0}{\delta}$.

- Apply this theorem with $X_0 \leq n\,(1.1)^r$ and $\delta = C/n$.

Summary and Conclusion

- Multi-valued decision variables give rise to several new phenomena (low fitness-runtime correlation for OneMax) and ask for new algorithm design choices (step size).

- Particular example: Minimization of a OneMax function of order r.
  1. Random step size: $E[T] = \Theta(nr \log n)$
  2. Unit step size: $E[T] = \Theta(nr + n \log n)$
  3. Harmonic step size: $E[T] = \Theta(n \log^2 r + n \log r \ \log n)$
  4. Self-adjusting: $E[T] = \Theta(n \log r + n \log n)$

Thanks!

- Open question: What does this mean for practical optimization?
  - Classic way to deal with multi-value decision variables: encode them in binary and then use your favorite evolutionary algorithm.
  - We feel that dealing with multi-valued variables in the more direct way we regard is more natural and should give better results. Let's see…