The \((1+\lambda)\) Evolutionary Algorithm with Self-Adjusting Mutation Rate

Benjamin Doerr\(^1\)  Christian Gießen\(^2\)
Carsten Witt\(^2\)  Jing Yang\(^1\)

PGMO DAYS 2017

\(^1\)Laboratoire d'Informatique (LIX)
École Polytechnique
Palaiseau, France

\(^2\)DTU Compute
Technical University of Denmark
Kgs. Lyngby, Denmark
• **ONEMAX Problem:** find a string $x = x_1 x_2 \ldots x_n$ with $x_i \in \{0, 1\}$ that minimize

\[ \text{OM}(x) = \sum_{i=1}^{n} x_i. \]
• **Onemax Problem**: find a string $x = x_1 x_2 \ldots x_n$ with $x_i \in \{0, 1\}$ that minimize

$$\text{OM}(x) = \sum_{i=1}^{n} x_i.$$ 

• **Black-box complexity** (Lehre, Witt (2010)): the smallest expected number of function evaluations needed to solve a problem.
• **OneMax** Problem: find a string $x = x_1 x_2 \ldots x_n$ with $x_i \in \{0, 1\}$ that minimize

$$OM(x) = \sum_{i=1}^{n} x_i.$$ 

• Black-box complexity (Lehre, Witt (2010)): the smallest expected number of function evaluations needed to solve a problem.

• We only consider unary unbiased variation operators which are symmetric with respect to the bit positions $[n] := \{1, \ldots, n\}$ and the bit values 0 and 1.
Structure of $(1+\lambda)$ EA

Algorithm 1 $(1+\lambda)$ EA

1: Select $x$ uniformly at random from $\{0, 1\}^n$;
2: repeat
3:     for $i = 1$ to $\lambda$ do
4:         Create $x_i$ by flipping each bit in a copy of $x$ independently with probability $r/n$;
5:     end for
6:     $x^* \leftarrow \arg\min_{x_i} OM(x_i)$;
7:     if $OM(x^*) \leq OM(x)$ then
8:         $x \leftarrow x^*$
9:     end if
10: until $OM(x) = 0$
Previous work and motivation

- Performance of EAs on \textit{OneMax} depends on parameters.

- Static mutation rate $r$ (i.e. mutation prob. $p = r/n$) for the $(1+\lambda)$ EA (Gießen, Witt, 2016):
  
  $$(1 \pm o(1)) \left( \frac{1}{2} \cdot n \log \log \lambda \log \lambda + e^r \cdot n \log n \right)$$

- Dynamic mutation rate $r = \max\left\{ \ln \lambda / \ln\left( e^n / k \right), 1 \right\}$ where $k = O(n)$ for the $(1+\lambda)$ EA (Badkobeh, Lehre, Sudholt, 2014):
  
  $$O\left( n \log \lambda + n \log n \lambda \right)$$

- Best possible among all $\lambda$-parallel mutation-based algorithms.

- Mutation rate should depend on the state of the current search process.

- Idea: let the algorithm changes parameters automatically according to recent performance (similar to $1/5$-rule).
Previous work and motivation

- Performance of EAs on OneMax depends on parameters
- Classic runtime analysis focus on static parameters (constant mutation probability).
Previous work and motivation

- Performance of EAs on \textsc{OneMax} depends on parameters
- Static mutation rate $r$ (i.e. mutation prob. $p = r/n$) for the $(1+\lambda)$ EA (Gießen, Witt, 2016):

\[(1 \pm o(1)) \left( \frac{1}{2} \cdot \frac{n \log \log \lambda}{\log \lambda} + \frac{e^r}{r} \cdot \frac{n \log n}{\lambda} \right)\]

- Dynamic mutation rate $r = \max\{\ln \lambda / \ln (en/k), 1\}$ where $k = \Omega\left(x\right)$ for the $(1+\lambda)$ EA (Badkobeh, Lehre, Sudholt, 2014):

\[O\left(n \log \lambda + n \log n\right)\]

best possible among all $\lambda$-parallel mutation-based algorithms.

- Mutation rate should depend on the state of the current search process
- Idea: let the algorithm changes parameters automatically according to recent performance (similar to $1/5$-rule)
Previous work and motivation

- Performance of EAs on \texttt{ONE}\texttt{MAX} depends on parameters
- Static mutation rate $r$ (i.e. mutation prob. $p = r/n$) for the (1+$\lambda$) EA (Gießen, Witt, 2016):
  \[
  (1 \pm o(1)) \left( \frac{1}{2} \cdot \frac{n \log \log \lambda}{\log \lambda} + \frac{e^r}{r} \cdot \frac{n \log n}{\lambda} \right)
  \]
- Improvement of a factor of $\Theta(\log \log \lambda)$ can be obtained by using dynamic parameter settings
• Performance of EAs on \( \text{ONE\textsc{MAX}} \) depends on parameters
• Static mutation rate \( r \) (i.e. mutation prob. \( p = r/n \)) for the \((1+\lambda)\) EA (Gießen, Witt, 2016):

\[
(1 \pm o(1)) \left( \frac{1}{2} \cdot \frac{n \log \log \lambda}{\log \lambda} + \frac{e^r}{r} \cdot \frac{n \log n}{\lambda} \right)
\]

• Dynamic mutation rate \( r = \max \{ \ln \lambda / \ln(\epsilon n / k), 1 \} \) where \( k = \Omega_M(x) \) for the \((1+\lambda)\) EA (Badkobeh, Lehre, Sudholt, 2014):

\[
O \left( \frac{n}{\log \lambda} + \frac{n \log n}{\lambda} \right)
\]

best possible among all \( \lambda \)-parallel mutation-based algorithms.
• Performance of EAs on $\text{ONE}\text{MAX}$ depends on parameters
• Static mutation rate $r$ (i.e. mutation prob. $p = r/n$) for the $(1+\lambda)$ EA (Gießen, Witt, 2016):

$$(1 \pm o(1)) \left( \frac{1}{2} \cdot \frac{n \log \log \lambda}{\log \lambda} + \frac{e^r}{r} \cdot \frac{n \log n}{\lambda} \right)$$

• Dynamic mutation rate $r = \max \{ \ln \lambda/\ln(e(n/k)), 1 \}$ where $k = O_M(x)$ for the $(1+\lambda)$ EA (Badkobeh, Lehre, Sudholt, 2014):

$$O \left( \frac{n}{\log \lambda} + \frac{n \log n}{\lambda} \right)$$

best possible among all $\lambda$-parallel mutation-based algorithms.

• Mutation rate should depend on the state of the current search process
Previous work and motivation

- Performance of EAs on OneMax depends on parameters.
- Static mutation rate $r$ (i.e. mutation prob. $p = r/n$) for the $(1+\lambda)$ EA (Gießen, Witt, 2016):
  $$\left(1 \pm o(1)\right) \left(\frac{1}{2} \cdot \frac{n \log \log \lambda}{\log \lambda} + \frac{e^r}{r} \cdot \frac{n \log n}{\lambda}\right)$$
- Dynamic mutation rate $r = \max \{\ln \lambda/\ln(en/k), 1\}$ where $k = \Omega(x)$ for the $(1+\lambda)$ EA (Badkobeh, Lehre, Sudholt, 2014):
  $$O\left(\frac{n}{\log \lambda} + \frac{n \log n}{\lambda}\right)$$
  best possible among all $\lambda$-parallel mutation-based algorithms.
- Mutation rate should depend on the state of the current search process.
- Idea: let the algorithm changes parameters automatically according to recent performance (similar to $1/5$-rule).
$(1+\lambda)$ EA with two-rate standard bit mutation

- **Mutation rate** $\lambda/2$
- **Mutation rate** $2r$

**Parent $x$**

- Replace parent by a best offspring $x^*$ if better or equal (breaking ties randomly, favouring offspring).
- Flip a fair coin.
  - Heads: Replace $r$ by the rate $x^*$ has been created with.
  - Tails: Replace $r$ with $r/2$ or $2r$ with probability $1/2$.
- Cap $r$ at $2$ and $\lambda/4$. 
Replace parent by a best offspring $x^*$ if better or equal (breaking ties randomly, favouring offspring)
Replace parent by a best offspring $x^*$ if better or equal (breaking ties randomly, favouring offspring)
- Flip a fair coin
  - Heads: Replace $r$ by the rate $x^*$ has been created with
  - Tails: Replace $r$ with $r/2$ or $2r$ with probability $\frac{1}{2}$
(1+λ) EA with two-rate standard bit mutation

- Replace parent by a best offspring $x^*$ if better or equal (breaking ties randomly, favouring offspring)
- Flip a fare coin
  - Heads: Replace $r$ by the rate $x^*$ has been created with
  - Tails: Replace $r$ with $r/2$ or $2r$ with probability $1/2$
- Cap $r$ at 2 and $n/4$
Rough estimation of good $r$

- Zero-keeping offspring: flip no zero bit, and may flip $0, 1, 2, \ldots$ one bits.
Rough estimation of good $r$

- Zero-keeping offspring: flip no zero bit, and may flip 0, 1, 2 · · · one bits.
- The probability of keeping all zeros is:

$$
\left(1 - \frac{r}{n}\right)^k \geq \left(1 - \frac{r}{n}\right)^n \approx e^{-r}
$$
• Zero-keeping offspring: flip no zero bit, and may flip 0, 1, 2⋯ one bits.
• The probability of keeping all zeros is:
  \[
  \left(1 - \frac{r}{n}\right)^k \geq \left(1 - \frac{r}{n}\right)^n \approx e^{-r}
  \]
• By taking \( r = c \ln(\lambda) \), the zero-keeping population is at least:
  \[
  e^{c \ln \lambda} \cdot \lambda = \lambda^{1-c}
  \]
Rough estimation of good $r$

- Zero-keeping offspring: flip no zero bit, and may flip 0, 1, 2 \cdots one bits.
- The probability of keeping all zeros is:
\[
(1 - \frac{r}{n})^k \geq (1 - \frac{r}{n})^n \approx e^{-r}
\]
- By taking $r = c \ln(\lambda)$, the zero-keeping population is at least:
\[
e^{c \ln \lambda} \cdot \lambda = \lambda^{1-c}
\]
- The best zero-keeping offspring (probably) makes progress.
Three regions

- Near region
- Middle region
- Far region

$r$

$\frac{n}{4}$

$\frac{n}{\lambda}$

$\frac{n}{\ln \lambda}$

$\frac{n}{2}$
$\sqrt{n}$ drift on distance at the beginning

\[ \frac{n}{2} \pm O(\sqrt{n}) \]

\[ O(\sqrt{n}) \]
drift on $r$ in far region
good rate $r$ in far region

\[ c_1(k) \ln(\lambda) \]

\[ c_2(k) \ln(\lambda) \]
Runtime: $O\left(\frac{n}{\ln(\lambda)}\right)$ generations in far region

\[ c_1(k) \ln(\lambda) \leq \Theta \left( \frac{\ln(\lambda)}{\ln(en/k)} \right) \leq c_2(k) \ln(\lambda) \]
Runtime: $O\left(\frac{n}{\ln(\lambda)}\right)$ generations in middle region
Runtime: $O\left(\frac{n \ln(n)}{\lambda}\right)$ generations in far region
Conclusion

- Analysis of a simple self-adjusting mutation scheme for the $(1+\lambda)$ EA
Conclusion

- Analysis of a simple self-adjusting mutation scheme for the $(1+\lambda)$ EA
- Matches the lower bound for every parallel unbiased black-box algorithm

Side-result: fixed rate of $r = \ln \lambda / 2$ yields $O(n / \log \lambda + n \log(n) / \sqrt{\lambda})$ (also optimal for $\lambda$ not too small)
Conclusion

- Analysis of a simple self-adjusting mutation scheme for the $(1+\lambda)$ EA
- Matches the lower bound for every parallel unbiased black-box algorithm
- Side-result: fixed rate of $r = \ln \lambda/2$ yields

$$O(n/\log \lambda + n \log(n)/\sqrt{\lambda})$$

(also optimal for $\lambda$ not too small)
Average runtime over 10000 runs

- **Self-adj. \((1+\lambda)\) EA**
- **Self-adj. \((1+\lambda)\) EA** (no random steps, \(F = 1.2\))
- **Static \((1+\lambda)\) EA** (\(p = \ln(\lambda)/(2n)\))
- **Static \((1+\lambda)\) EA** (\(p = 1/n\))
- **\((1+\lambda)\) EA** using \(p = \max(1/n, \ln(\lambda)/(\ln(en/d(x))))\)
Average runtime over 10000 runs

- $F = 2.0$
- $F = 1.5$
- $F = 1.2$