Extreme Value Based Adaptive Operator Selection

Álvaro Fialho[1], Luis Da Costa[2], Marc Schoenauer[1,2] and Michèle Sebag[1,2]

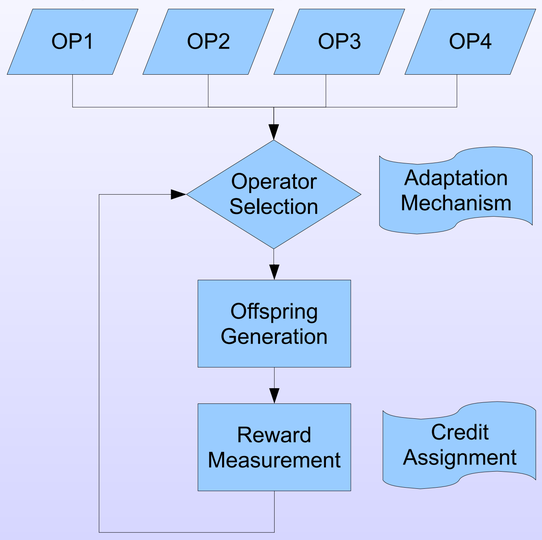
{alvaro.fialho,luis.dacosta,marc.schoenauer,michele.sebag}@inria.fr

[1] Microsoft Research-INRIA Joint Centre, Orsay, France
[2] TAO team, INRIA Saclay - Île-de-France & LRI (UMR CNRS 8623), Orsay, France

We address the question of how to estimate the quality of the operators used on an evolutionary algorithm. We argue that operators giving rare but **extreme** improvements are very valuable, and we propose a mechanism to reward them.

Adaptive Operator Selection

Dynamically select "best" operator during evolution



- **Credit assignment**: What reward?
- **Adaptation mechanism**: How to adapt operator selection depending on reward?

Credit Assignment

Previous work

- Reward $\propto$ fitness improvement,
- possibly averaged with decay factor,
- eventually distributed among offspring's back lineage

Bucket Brigade-like algorithm

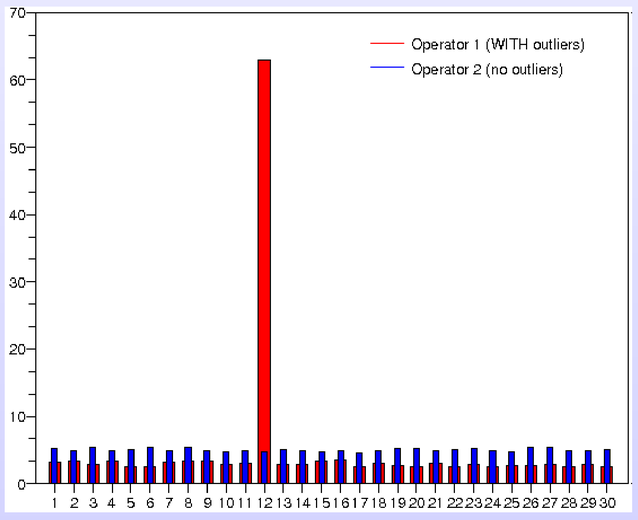
CLaim: Outliers matter

Very rare but extremely beneficial events can be more important than averages

- Indications from other disciplines

e.g. finance, rogue waves, ...

- Suggested in AOS context [Whitacre *et al*, 2006]



Two operators ("**red**" and "**blue**"), with same expectation. **Red** is extreme, **blue** is not

Proposed here: Extreme Reward

Best fitness improvement over a sliding window

Typically 50

Adaptation Mechanisms

Previous Work: Probability Matching and Adaptive Pursuit

see e.g. [Thierens, 2007]

Select an operator based on probabilities s_i ($\sum s_i = 1$)

- $\hat{Q}_{i,t}$ = estimated quality of operator i at time t
- Selecting operator $i \rightarrow$ reward $r_{i,t}$
- Estimated quality updated by relaxation

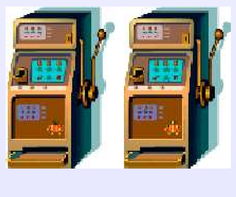
$$\hat{Q}_{i,t+1} = (1-\alpha)\hat{Q}_{i,t} + \alpha\, r_{i,t}$$

- **Probability Matching**: s_i proportional to Q_i
- **Adaptive Pursuit**: increase s_i for operator with best estimated quality, decrease all other s_j's evenly.
- Greediness: $s_i \geq P_{min}$ User-defined parameter

Dynamic Multi-Armed Bandits

proposed at GECCO 2008

- **Multi-Armed Bandit setting**



Reward for choosing arm j is

$$r_j = \begin{cases} 1 \text{ with prob } = p_j \\ 0 \text{ with prob } = 1 - p_j \end{cases}$$

- **Goal** Maximizze cumulated reward
- **UCB1 algorithm** [Auer *et al*, 2002]

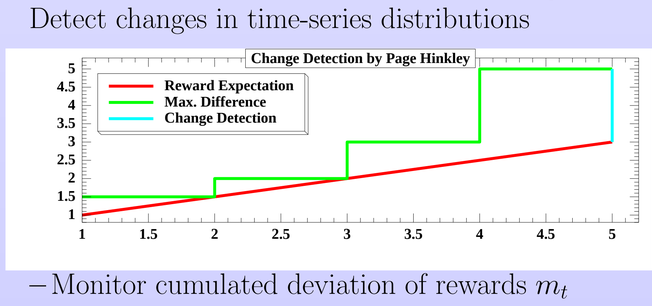
Be optimistic when facing the unknown:
At time t, choose arm j maximizing

$$\hat{r}_{j,t} + \sqrt{\frac{2\log\sum_k n_{k,t}}{n_{j,t}}},$$

$\begin{cases} \hat{r}_{j,t} : \text{ empirical reward for arm } j, \text{ favors exploitation} \\ n_{j,t} : \#\text{ times arm } j \text{ was chosen, favors exploration} \end{cases}$

- **Scaling** If $r_{j,t} \in [a,b] \neq [0,1]$: balance between exploitation and exploration requires scaling of $\hat{r}_{j,t}$
- **Dynamic rewards**
  - Operators' rewards likely to change during a run
  - UCB1 algorithm reacts slowly
  - $\rightarrow$ should be restarted in case of change in rewards
- **Page-Hinkley statistics** [Page, 1954]

Detect changes in time-series distributions



  - Monitor cumulated deviation of rewards m_t
  - Trigger restart when $|\text{Max}_{i \leq t}(m_i) - m_t| > \lambda$

λ user-defined parameter

Dynamic Multi-Armed Bandits (D-MAB):
UCB1 + Scaling + Page Hinkley

Winner Pascal EvE Challenge [Hartland *et al*, 2007]

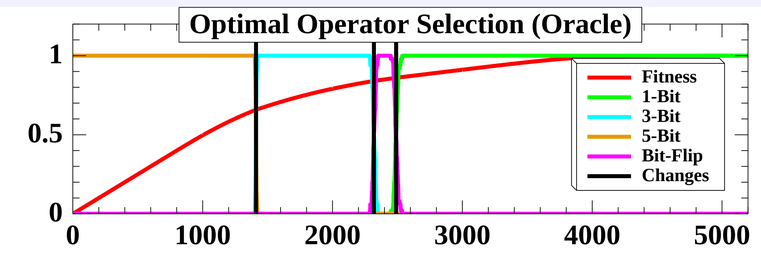
Experiments

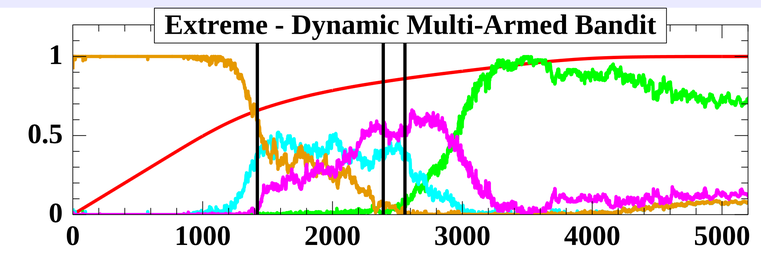
- **10000-bits OneMax** the drosophila of EC!
- **4 mutation operators**, (1+50)-EA
  - k-bit mutation (k=1,3,5) flips exactly k uniformly chosen bits
  - standard **bit-flip** mutation flips all bits with probability $\frac{1}{n}$
- Which is the best operator only depends on the fitness value

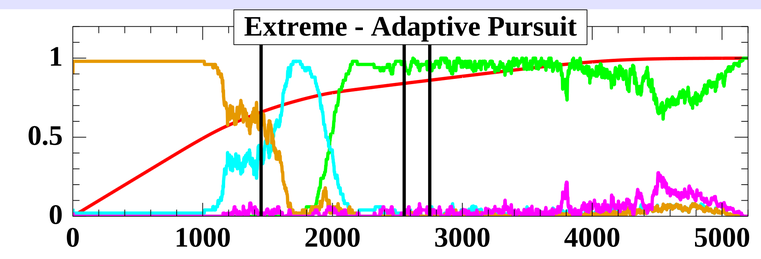
5-bit when low, 1-bit close to optimum, ...

Typical Behaviors:

Proportion of operators actually used, together with **fitness**







Extreme vs Average?

Dynamic-MAB	
Credit Assignment	Generations to Optimum
Extreme	5467 $\pm$ 513
Average	7727 $\pm$ 642

Adaptive Pursuit	
Credit Assignment	Generations to Optimum
Extreme	5478 $\pm$ 299
Average	5830 $\pm$ 324

Baseline	
Non-Adaptive Strategy	Generations to Optimum
Optimal according to Oracle	5069$\pm$ 292
Best Naive: $\mathcal{U}$(1-Bit+5-Bit)	6793 $\pm$ 625
Complete Naive: $\mathcal{U}$(all ops.)	7813 $\pm$ 708

Open Issues

- How much exploration are we really doing?
- Can those results be generalized? need a sound test suite
- Make some meta-parameters (self-)adaptive? e.g. λ in PH test
- What about a rank-based reward? $\rightarrow$scale-invariance

Acknowledgments: Luis Da Costa is funded by the European Project EvoTest number IST-33472.