Zastosowanie algorytmu UCB w optymalizacji ciągłej

Mateusz Zaborski
M.Zaborski@mini.pw.edu.pl

Faculty of Mathematics and Information Science
Warsaw University of Technology

13.04.2022

Table of Contest

Motivatation

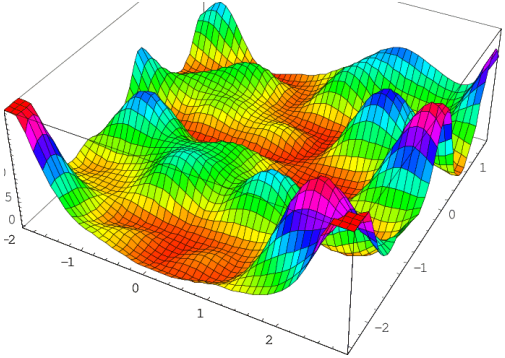


Figure 1: Source [Pintér, 2007]

Figure 2: Source: [Wikipedia, 2022]

Single-objective continuous optimization

Single-objective continuous black box optimization problem

- Goal - minimize an objective function (or fitness function or cost function)

$$f : \mathbb{R}^n \to \mathbb{R}$$

- Single-objective continuous optimization
- Black Box scenario
  - Function values of evaluated search points are the only accessible information
  - Gradients are not available
- Search cost - number of function evaluations

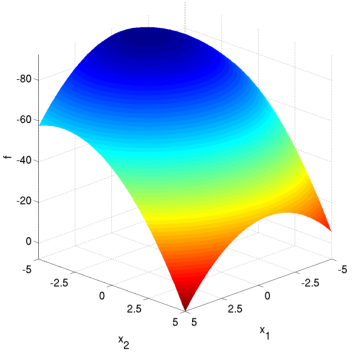


Figure 3: Sphere 2D function from COCO BBOB benchmark [Hansen et al., 2019]

Single-objective continuous black box optimization problem

Examples

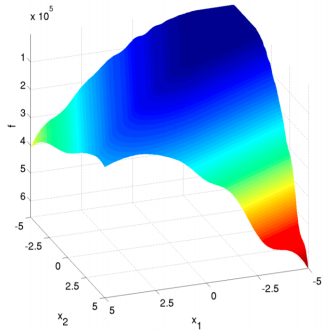


Figure 4: Attractive Sector Function (2D)[Hansen et al., 2019]

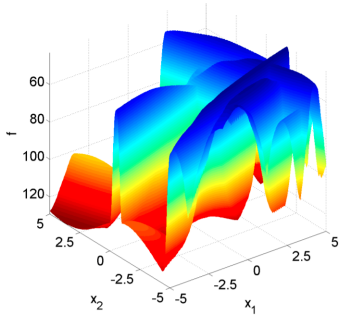


Figure 5: Gallagher's Gaussian 21-hi Peaks Function (2D)[Hansen et al., 2019]

Single-objective continuous black box optimization problem

Challenges

Challenges:

- Infinite number of solutions in a continuous domain
- Multidimensional problems are difficult for grid search
- The goal is to find a global (not local) optimum
- Unknown function shape
  - Non-linear, non-quadratic
  - Discontinuities, sharp ridges
  - Non-separability
  - Ill-conditioning
- **Unknown target value**

Per-instance algorithm selection

How to determine **a priori** which algorithm should be used to solve a given instance?

Per-instance algorithm selection problem - given a computational problem, a set of algorithms for solving this problem, and a specific instance that needs to be solved, determine which of the algorithms can be expected to perform best on that instance [**?**].

No-free-lunch (NFL) theorem [Wolpert et al., 1995]

"(...) all algorithms that search for an extremum of a cost function perform exactly the same, according to any performance measure, when averaged over all possible cost functions"

Problems of optimization algorithms

Problems:

- Over-specialization (algorithm class/type)
  - Per benchmark (e.g., COCO BBOB restart mechanisms)
  - Per optimization budget (e.g., # evaluations, # iterations)
  - Per measure
- Over-parameterization
  - Parameters
  - Population size
  - Custom mechanisms (population size reduction)
- overcomplication

Examples: CMA-ES (restart conditions), MadDE (multiple adaptations), R-SHADE / RL-SHADE + COCO BBOB [Hansen et al., 2019]

Tuning DE for cheap, medium and expensie budgets [Tanabe and Fukunaga, 2015]

1. R-DE, R-SHADE, RL-SHADE considered
2. Various parameterization applied
3. Various restart mechanisms applied
4. CEC2014 benchmarks as training set, using SMAC [Hutter et al., 2011]
5. COCO BBOB as test set

Tuning DE for cheap, medium and expensie budgets [Tanabe and Fukunaga, 2015]

(b) R-SHADE

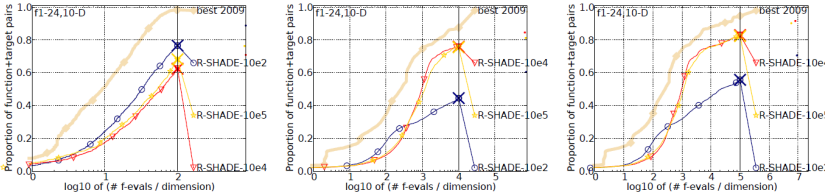
Parameters	Range	Default	MaxEvals $10^2 \times D$	MaxEvals $10^4 \times D$	MaxEvals $10^5 \times D$
population rate	[0, 10]	5.0	0.45	3.74	3.96
initial M_F	[0, 1]	0.5	0.90	0.53	0.38
initial M_{CR}	[0, 1]	0.5	0.06	0.71	0.94
p	[0, 0.2]	0.05	0.01	0.13	0.09
archive rate	[0, 2]	1.0	1.92	0.65	0.12
memory size	[1, 20]	10	16	10	11

Tuning DE for cheap, medium and expensie budgets [Tanabe and Fukunaga, 2015]

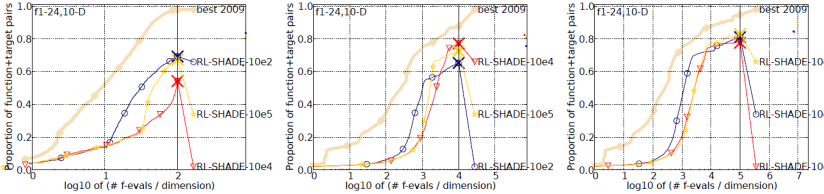
(c) RL-SHADE

Parameters	Range	Default	MaxEvals $10^2 \times D$	MaxEvals $10^4 \times D$	MaxEvals $10^5 \times D$
initial population rate	[10, 20]	15	5.19	13.63	16.39
initial M_F	[0, 1]	0.5	0.84	0.93	0.28
initial M_{CR}	[0.1, 1]	0.5	0.12	0.72	0.43
p	[0, 0.2]	0.05	0.01	0.09	0.02
archive rate	[0, 2]	1.0	1.13	1.86	0.94
memory size	[1, 20]	10	7	3	7
B	[1, 10]	1	8	1	5

Tuning DE for cheap, medium and expensie budgets [Tanabe and Fukunaga, 2015]



(b) R-SHADE



(c) RL-SHADE

How to employ UCB?

UCB

$$\pi_{\mathrm{UCB1}}(n) = \mathrm{argmax}_i \left(\widehat{\mu}_i(n) + c\sqrt{\frac{2\ln n}{T_i(n)}} \right)$$

Figure 6: UCB-1 formula [Baudiš and Pošík, 2014]

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Online Black-box Algorithm Portfolios for Continuous Optimization

Petr Baudiš and Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Technická 2, 166 27 Prague 6, Czech Republic
pasky@ucw.cz, petr.posik@fel.cvut.cz

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

General idea:

- Treat algorithms as "black-boxes"
- Consider *online* selection (no features or landscape)
- Dynamically switch between algorithms during the optimization run)
- UCB1 multi-armed bandit policy applied to unbounded rewards
- COCO BBOB benchmarking

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Considered Bandit Policies:

- epsilon-greedy policy (choose the best with $1 - \epsilon$ probability)
- **Upper Connfdence Bound (UCB1)**
- **MetaMax**
- probability matching (probabilities based on proportions of their reward estimate)
- adaptive pursuit ("winner takes all" + learning rate β)
- Max k-Armed Bandit Problem (maximize the highest rather than cumulative reward)
- Threshold Ascent (s-sized window of the best-so-far rewards produced)

$$\pi_{TA}(n) = \text{argmax}_i \left(\widehat{\mu}_i(n) + \frac{\alpha + \sqrt{2T_i(n)\widehat{\mu}_i(n)\alpha + \alpha^2}}{T_i(n)} \right) \qquad \alpha = ln\left(\frac{2nK}{\delta}\right)$$

$$\widehat{\mu}_i(n) = S_{it}/T_i(n)$$

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Reward estimate

- raw value (negative function value) - only if value approaches the optimum smoothly
- value rank - rank algorithms using value linear scaling to $[0, 1]$
- exponentially weighted moving average (EWMA) - extension of raw value and value rank

Problems:

- the Multi-Armed Bandit Problem assumes that the reward distributions are stationary and rewards are independent
  - UCB-1 policy can be used as-is for non-stationary if c is set correctly
- functional landscape is evolving
- rate of improvement changes

Online Black-box Algorithm Portfolios

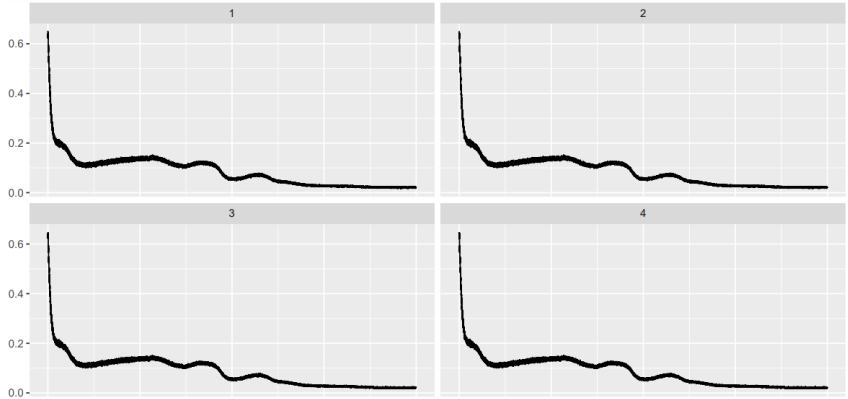


Figure 7: Improvements (binary measure) for F_1 - F_4, $D = 10$ from CEC2021

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

UCB-1 and raw values:

With these two assumptions, a *log-rescaling* process is straightforward. First, we convert the absolute f_i values to values relative to the supposed optimum and rescale the values logarithmically:

$$g_i = \log(f_i - f_{opt?})$$

Second, we assign rewards by linear rescaling of the preprocessed values:

$$\mu_i = 1 - \frac{g_i - \min_j g_j}{\max_j g_j - \min_j g_j}$$

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Algorithm Selection Strategies:

RR: As one baseline strategy, we used a round robin policy that samples each algorithm equally, in their portfolio order. (This is different from a "run in parallel" strategy in that the algorithms consume different budgets to sample a single iteration.)

EG: The epsilon-greedy policy with $\varepsilon = 0.5$.

PM: The Probability Matching policy with EWMA adaptation rate $\alpha = 0.1$ and $P_{\min} = 0.025$ (effectively $P_{\max} = 0.825$).

AP: The Adaptive Pursuit policy with EWMA adaptation rate $\alpha = 0.5$, probability learning rate $\beta = 0.1$ and $P_{\min} = 0.075$ ($P_{\max} = 0.55$).

TA: The Threshold Ascent policy with window $s = |\text{portfolio}|$ and $\delta = 1/s$.

MMK: The MetaMax(K) policy with arms count $K = 2 \cdot |\text{portfolio}|$.

MML: The MetaMaxLog(L) policy with instance addition pace $L = 8$.

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Algorithm Selection Strategies:

SR: The Sum of Ranks reward assignment in conjunction with the UCB1 exploration policy with $c = 4$, window $W = 50$ and decay $D = 0.95$.

AUC: The Area Under the Curve reward assignment in conjunction with the UCB1 exploration policy with $c = 2$, window $W = 50$ and decay $D = 0.75$.

RUCB: The UCB1 policy with EWMA-recent ranks as reward estimates ($\widehat{\mu}_i$ in Equation 1), with $c = 8$ and adaptation rate $\alpha = 0.9$.

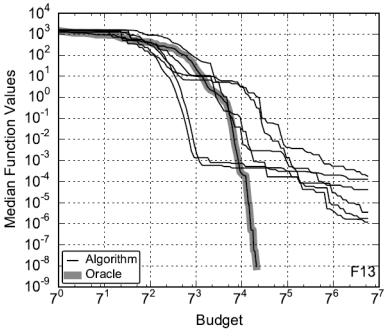
LUCB: The UCB1 policy with EWMA-recent log-rescaled values as reward estimates ($\widehat{\mu}_i$ in Equation 1), with $c = 16$ and adaptation rate $\alpha = 0.7$.

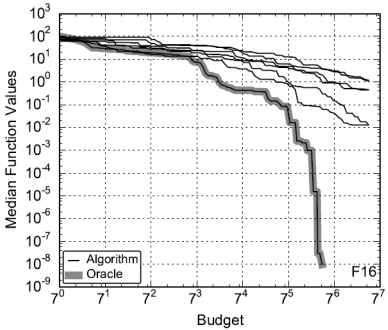
Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Experimental evaluation

- Portfolio of 7 algorithms: CMA-ES and 6 numerical optimization algorithms from SciPy
- Algorithms are black-box
- Algorithms run parallel
- One algorithm is chosen per one iteration

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]





Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Table 1: The assignment of individual COCO benchmark functions to the classes we have devised, determined on the performance of our portfolio on the functions. Vertical lines delineate the standard function classes used by COCO.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
By solvers (s/m)	m	s	s	s	m	s	s	m	m	s	s	s	s	m	s	s	s	s	s	s	s	s	s	m
By winner (g/b)	b	b	b	b	b	g	g	b	b	g	g	g	g	b	b	g	g	g	b	b	g	g	g	b
By converg. (s/v)	s	s	s	s	s	v	s	v	v	v	v	v	v	v	v	s	s	s	v	v	v	v	v	v

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Table 3: The log-slowdown (average, standard deviation and median of per-function medians) of portfolio algorithms and strategies compared to oracle strategy (i.e. the best algorithm for each function).

Solver	all	multi	single	volatile	stable	CMA-good	CMA-bad
CMA	$0.7^{\pm 0.9}_{\mid 0.0}$	$0.3^{\pm 0.4}_{\mid 0.0}$	$0.6^{\pm 1.0}_{\mid 0.0}$	$0.5^{\pm 0.8}_{\mid 0.0}$	$1.1^{\pm 1.0}_{\mid 1.1}$	$0.0^{\pm 0.0}_{\mid 0.0}$	$1.5^{\pm 0.9}_{\mid 1.1}$
BFGS	$1.5^{\pm 1.3}_{\mid 1.1}$	$2.0^{\pm 1.4}_{\mid 3.0}$	$1.8^{\pm 1.2}_{\mid 2.0}$	$1.4^{\pm 1.2}_{\mid 1.1}$	$1.7^{\pm 1.3}_{\mid 2.0}$	$2.2^{\pm 1.0}_{\mid 3.0}$	$0.8^{\pm 1.1}_{\mid 0.2}$
L-BFGS-B	$1.9^{\pm 1.2}_{\mid 2.2}$	$2.3^{\pm 1.0}_{\mid 3.0}$	$2.3^{\pm 0.9}_{\mid 3.0}$	$1.8^{\pm 1.2}_{\mid 2.1}$	$1.9^{\pm 1.2}_{\mid 2.6}$	$2.5^{\pm 0.8}_{\mid 3.0}$	$1.1^{\pm 1.1}_{\mid 0.8}$
Nelder-Mead	$2.1^{\pm 1.1}_{\mid 3.0}$	$3.0^{\pm 0.0}_{\mid 3.0}$	$2.6^{\pm 0.8}_{\mid 3.0}$	$2.0^{\pm 1.2}_{\mid 3.0}$	$2.4^{\pm 0.8}_{\mid 3.0}$	$2.5^{\pm 0.9}_{\mid 3.0}$	$1.8^{\pm 1.1}_{\mid 1.8}$
CG	$2.3^{\pm 1.2}_{\mid 3.0}$	$2.2^{\pm 1.2}_{\mid 3.0}$	$2.7^{\pm 0.7}_{\mid 3.0}$	$2.3^{\pm 1.1}_{\mid 3.0}$	$2.3^{\pm 1.3}_{\mid 3.0}$	$2.8^{\pm 0.6}_{\mid 3.0}$	$1.7^{\pm 1.3}_{\mid 1.8}$
SLSQP	$2.3^{\pm 1.0}_{\mid 3.0}$	$3.0^{\pm 0.0}_{\mid 3.0}$	$2.8^{\pm 0.6}_{\mid 3.0}$	$2.4^{\pm 1.0}_{\mid 3.0}$	$2.3^{\pm 1.0}_{\mid 3.0}$	$2.9^{\pm 0.4}_{\mid 3.0}$	$1.8^{\pm 1.2}_{\mid 1.7}$
Powell	$2.3^{\pm 1.2}_{\mid 3.0}$	$3.0^{\pm 0.0}_{\mid 3.0}$	$2.4^{\pm 1.2}_{\mid 3.0}$	$3.0^{\pm 0.0}_{\mid 3.0}$	$1.2^{\pm 1.4}_{\mid 0.5}$	$3.0^{\pm 0.0}_{\mid 3.0}$	$1.6^{\pm 1.4}_{\mid 1.8}$
LUCB	$0.9^{\pm 0.9}_{\mid 0.8}$	$0.1^{\pm 0.3}_{\mid 0.0}$	$0.9^{\pm 1.0}_{\mid 0.8}$	$1.3^{\pm 0.9}_{\mid 1.1}$	$\mathbf{0.3}^{\pm 0.6}_{\mid 0.2}$	$1.1^{\pm 0.9}_{\mid 1.1}$	$0.8^{\pm 0.9}_{\mid 0.6}$
RUCB	$1.3^{\pm 1.0}_{\mid 1.1}$	$2.2^{\pm 1.1}_{\mid 3.0}$	$1.4^{\pm 1.1}_{\mid 1.3}$	$1.4^{\pm 0.8}_{\mid 1.2}$	$1.1^{\pm 1.3}_{\mid 0.5}$	$1.7^{\pm 0.9}_{\mid 1.4}$	$0.8^{\pm 0.9}_{\mid 0.7}$
EG	$1.4^{\pm 1.0}_{\mid 1.4}$	$2.3^{\pm 1.0}_{\mid 3.0}$	$1.6^{\pm 1.0}_{\mid 1.6}$	$1.6^{\pm 0.7}_{\mid 1.5}$	$1.1^{\pm 1.3}_{\mid 0.5}$	$2.0^{\pm 0.7}_{\mid 1.9}$	$0.9^{\pm 0.9}_{\mid 0.7}$
SR	$1.5^{\pm 1.2}_{\mid 1.3}$	$2.1^{\pm 1.3}_{\mid 3.0}$	$1.6^{\pm 1.3}_{\mid 1.3}$	$1.9^{\pm 0.9}_{\mid 1.3}$	$0.9^{\pm 1.3}_{\mid 0.4}$	$1.9^{\pm 1.1}_{\mid 1.3}$	$1.1^{\pm 1.1}_{\mid 0.8}$
RR	$1.5^{\pm 1.0}_{\mid 1.5}$	$2.2^{\pm 1.1}_{\mid 3.0}$	$1.7^{\pm 1.1}_{\mid 1.7}$	$1.8^{\pm 0.8}_{\mid 1.7}$	$1.2^{\pm 1.2}_{\mid 0.5}$	$2.1^{\pm 0.8}_{\mid 1.8}$	$1.0^{\pm 0.9}_{\mid 0.7}$
PM	$1.5^{\pm 1.1}_{\mid 1.6}$	$2.4^{\pm 0.9}_{\mid 3.0}$	$1.6^{\pm 1.2}_{\mid 1.9}$	$1.8^{\pm 0.8}_{\mid 1.9}$	$1.1^{\pm 1.4}_{\mid 0.5}$	$2.0^{\pm 0.9}_{\mid 2.3}$	$1.0^{\pm 1.1}_{\mid 0.9}$
MMK	$1.6^{\pm 1.0}_{\mid 1.5}$	$2.4^{\pm 0.9}_{\mid 3.0}$	$1.7^{\pm 1.0}_{\mid 1.8}$	$1.8^{\pm 0.8}_{\mid 1.7}$	$1.2^{\pm 1.2}_{\mid 0.6}$	$2.1^{\pm 0.8}_{\mid 2.0}$	$1.1^{\pm 0.9}_{\mid 0.9}$
TA	$1.7^{\pm 1.2}_{\mid 1.6}$	$3.0^{\pm 0.0}_{\mid 3.0}$	$1.9^{\pm 1.2}_{\mid 2.3}$	$2.0^{\pm 1.0}_{\mid 1.7}$	$1.3^{\pm 1.4}_{\mid 0.5}$	$2.1^{\pm 1.0}_{\mid 3.0}$	$1.3^{\pm 1.2}_{\mid 0.9}$
AP	$1.7^{\pm 1.1}_{\mid 1.7}$	$3.0^{\pm 0.0}_{\mid 3.0}$	$1.9^{\pm 1.1}_{\mid 1.9}$	$2.0^{\pm 0.8}_{\mid 1.8}$	$1.2^{\pm 1.3}_{\mid 0.5}$	$2.1^{\pm 0.7}_{\mid 1.9}$	$1.2^{\pm 1.1}_{\mid 0.9}$
MML	$1.7^{\pm 1.2}_{\mid 1.9}$	$3.0^{\pm 0.0}_{\mid 3.0}$	$1.9^{\pm 1.2}_{\mid 2.1}$	$2.1^{\pm 0.8}_{\mid 2.0}$	$1.1^{\pm 1.3}_{\mid 0.5}$	$2.2^{\pm 0.9}_{\mid 2.2}$	$1.2^{\pm 1.2}_{\mid 1.0}$
AUC	$1.8^{\pm 1.2}_{\mid 1.9}$	$2.1^{\pm 1.2}_{\mid 3.0}$	$1.9^{\pm 1.2}_{\mid 2.4}$	$2.3^{\pm 0.7}_{\mid 2.9}$	$0.9^{\pm 1.2}_{\mid 0.4}$	$2.2^{\pm 1.0}_{\mid 3.0}$	$1.4^{\pm 1.2}_{\mid 1.3}$

Online Black-box Algorithm Portfolios [Baudiš and Pošík, 2014]

Comments:

- Is portfolio better then the best algorithm?
- There is no "the worst" measure
- Does the portfolio (UCB-1) protects against local minima?
- Can re-scalling and ranking techniques be applied to non-UCB strategies?
- Can population switching improve the results?
  - Evaluations saving
  - Limiting the variability of the function value

Open questions

Can UCB-1 be used for other purposes?

Multi-restart strategy

Efficient Multi-Start Strategies for Local Search Algorithms

András György GYA@SZIT.BME.HU
Machine Learning Research Group
Computer and Automation Research Institute
of the Hungarian Academy of Sciences
1111 Budapest, Hungary

Levente Kocsis KOCSIS@SZTAKI.HU
Data Mining and Web Search Research Group, Informatics Laboratory
Computer and Automation Research Institute
of the Hungarian Academy of Sciences
1111 Budapest, Hungary

UCB-1 in space dividing [Notsu et al., 2016]

Application of the UCT Algorithm for Noisy Optimization Problems

Akira Notsu, Satoshi Kane, Seiki Ubukata, Katsuhiro Honda
Osaka Prefecture University
Gakuen-cho 1-1, Naka-ku, Sakai, Osaka, 599-8531 JAPAN
Email: notsu@cs.osakafu-u.ac.jp

UCB-1 in space dividing [Notsu et al., 2016]

Idea:

- search area is divided into several subareas (e.g., 5x5)
- area == slot == bandit
- grid area search (GAS)
- recursively choosing best subarea

UCB-1 in space dividing [Notsu et al., 2016]

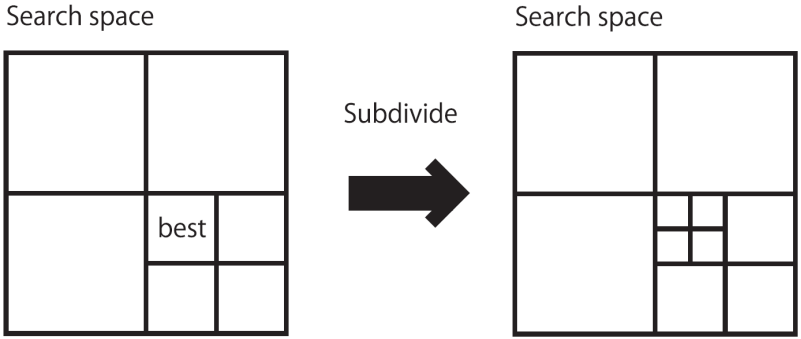


Fig. 1. Subdivide best-value subarea

UCB-1 in space dividing [Notsu et al., 2016]

$$UCB_i = \frac{1}{1 + \mathrm{e}^{C_1 \min X_i}} + \sqrt{\frac{C_2 \log n}{n_i}} \tag{4}$$

where C_1 and C_2 are parameters, $\min X_i$ is the best solution of subarea i. We set C_1 and C_2 as 1 for simplification in the following simulations.

UCB-1 in space dividing [Notsu et al., 2016]

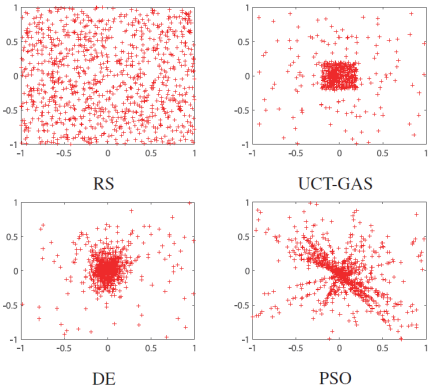


Fig. 9. F_1 with 0.5 noise: example of search points

UCB-1 in space dividing [Notsu et al., 2016]

function	algorithm	at 250 cycles	at 500 cycles	at 1,000 cycles
F_1 with 0.1 noise	RS	0.0443 ± 0.0247	0.0307 ± 0.0163	0.0224 ± 0.0128
	UCT-GAS	0.0072 ± 0.0046	0.0043 ± 0.0033	0.0027 ± 0.0023
	PSO	0.0170 ± 0.0094	0.0088 ± 0.0046	0.0038 ± 0.0024
	DE	0.0135 ± 0.0078	0.0040 ± 0.0024	0.0015 ± 0.0009
F_1 with 0.5 noise	RS	0.0974 ± 0.0481	0.0713 ± 0.0364	0.0509 ± 0.0272
	UCT-GAS	0.0231 ± 0.0206	0.0127 ± 0.0126	0.0074 ± 0.0048
	PSO	0.0478 ± 0.0294	0.0226 ± 0.0129	0.0111 ± 0.0074
	DE	0.0408 ± 0.0219	0.0160 ± 0.0096	0.0071 ± 0.0040
F_2 with 0.1 noise	RS	0.2792 ± 0.1222	0.2111 ± 0.0933	0.1569 ± 0.0698
	UCT-GAS	0.1077 ± 0.0960	0.0667 ± 0.0702	0.0492 ± 0.0532
	PSO	0.1922 ± 0.0896	0.1208 ± 0.0610	0.0698 ± 0.0396
	DE	0.1761 ± 0.0858	0.0933 ± 0.0455	0.0309 ± 0.0277
F_2 with 0.5 noise	RS	0.3767 ± 0.1544	0.2965 ± 0.1121	0.2382 ± 0.0915
	UCT-GAS	0.1969 ± 0.1222	0.1212 ± 0.0862	0.0920 ± 0.0648
	PSO	0.3132 ± 0.1304	0.1992 ± 0.0940	0.1247 ± 0.0626
	DE	0.2882 ± 0.1116	0.1849 ± 0.0751	0.1040 ± 0.0486

UCB-1 in space dividing [Notsu et al., 2016]

Observations:

- Trivial functions examined ($D = 2$)
- Small number of evaluations
- Interesting idea (e.g., not rectangles)

Idea of UCB application

Artificial Bee Colony[Karaboga and Basturk, 2007]

A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm

Dervis Karaboga · Bahriye Basturk

Artificial Bee Colony [Karaboga and Basturk, 2007]

- Initialize.
- REPEAT.
  (a) Place the employed bees on the food sources in the memory;
  (b) Place the onlooker bees on the food sources in the memory;
  (c) Send the scouts to the search area for discovering new food sources.
- UNTIL (requirements are met).

Open questions

Potential UCB-1 applications:

- Parameter adaptation (e.g., SHADE modification)
- Managing population
- Balancing exploration and exploitation
- Restart elimination

Conclusions:

- UCB-1 is usefull in Algorithm / portfolio selection
- UCB-1 might be usefull in search space dividing
- Exploration - exploitation balancing is an open issue
- Elimination of restarts is an open issue

Bibliography I


Baudiš, P. and Pošík, P. (2014).
Online black-box algorithm portfolios for continuous optimization.
In *International Conference on Parallel Problem Solving from Nature*, pages 40–49. Springer.


György, A. and Kocsis, L. (2011).
Efficient multi-start strategies for local search algorithms.
Journal of Artificial Intelligence Research, 41:407–444.


Hansen, N., Brockhoff, D., Mersmann, O., Tusar, T., Tusar, D., ElHara, O. A., Sampaio, P. R., Atamna, A., Varelas, K., Batu, U., Nguyen, D. M., Matzner, F., and Auger, A. (2019).
COmparing Continuous Optimizers: numbbo/COCO on Github.


Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011).
Sequential model-based optimization for general algorithm configuration.
In *International conference on learning and intelligent optimization*, pages 507–523. Springer.

Bibliography II

Karaboga, D. and Basturk, B. (2007).
A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm.
Journal of global optimization, 39(3):459–471.

Notsu, A., Kane, S., Ubukata, S., and Honda, K. (2016).
Application of the uct algorithm for noisy optimization problems.
In *2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 48–52. IEEE.

Pintér, J. D. (2007).
Nonlinear optimization with gams/lgo.
Journal of Global Optimization, 38(1):79–101.

Bibliography III

Tanabe, R. and Fukunaga, A. (2015).
Tuning differential evolution for cheap, medium, and expensive computational budgets.
In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2018–2025. IEEE.

Wikipedia (2022).
Multi-armed bandit — Wikipedia, the free encyclopedia.
http://en.wikipedia.org/w/index.php?title=Multi-armed%20bandit&oldid=1081655278.

[Online; accessed 12-April-2022].

Wolpert, D. H., Macready, W. G., et al. (1995).
No free lunch theorems for search.
Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute.