Self-adaptation via multi-objectivisation: an empirical study

Qin, Xiaoyu; Lehre, Per Kristian

DOI:
10.1007/978-3-031-14714-2_22

Citation for published version (Harvard):
Qin, X & Lehre, PK 2022, Self-adaptation via multi-objectivisation: an empirical study. in G Rudolph, AV Kononova, H Aguirre, P Kerschke, G Ochoa & T Tušar (eds), Parallel Problem Solving from Nature – PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I. 1 edn, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 13398 LNCS, Springer, pp. 308–323, The seventeenth International Conference on Parallel Problem Solving from Nature, Dortmund, Germany, 10/09/22. https://doi.org/10.1007/978-3-031-14714-2_22

Appendix 1 Omitted algorithms in this paper

Algorithm 3 Multi-objective sorting mechanism [33]

Require: Population sizes $\lambda \in \mathbb{N}$. Population $P_t \in \mathcal{Y}^\lambda$. Fitness function f.

Sort P_t into strict non-dominated fronts $\mathcal{F}_0^t, \mathcal{F}_1^t, \ldots$ based on $f_1(x,\chi) := f(x)$ and $f_2(x,\chi) := \chi$.
for $\mathcal{F} = \mathcal{F}_0^t, \mathcal{F}_1^t, \ldots$ **do**
    Sort $\mathcal{F}$ such that $f_1(\mathcal{F}(1)) > f_1(\mathcal{F}(2)) > \ldots$.
$P_t := \left(\mathcal{F}_0^t, \mathcal{F}_1^t, \ldots\right)$.
return P_t.

Algorithm 4 Strict non-dominated sorting [33]

Require: Population sizes $\lambda \in \mathbb{N}$. Population $P \in \mathcal{Z}^\lambda$, where $\mathcal{Z}$ is a finite state space. Objective functions $f_1, f_2, \ldots : \mathcal{Z} \to \mathbb{R}$ (assume to maximise all objective functions).

for each individual $P(i)$ **do**
    Set $S_i := \emptyset$ and $n_i := 0$.
for $i = 1, \ldots, \lambda$ **do**
    for $j = 1, \ldots, \lambda$ **do**
        if $P(i) \prec P(j)$ based on $f_1, f_2, \ldots$ **then**
            $S_i := S_i \cup \{P(i)\}$,
        else if $P(j) \prec P(i)$ based on $f_1, f_2, \ldots$ **then**
            $n_i := n_i + 1$,
        else if $f_\ell(P(i)) = f_\ell(P(j))$ where $\ell = 1, 2, \ldots$ **then**
            if $P(i) \notin S_j$ **then** $S_i := S_i \cup \{P(i)\}$ **else** $n_i := n_i + 1$.
    if $n_i = 0$ **then** $\mathcal{F}_0 = \mathcal{F}_0 \cup \{P(i)\}$.
Set $k := 0$.
while $\mathcal{F}_k \neq \emptyset$ **do**
    $Q := \emptyset$.
    for each individual $P(i) \in \mathcal{F}_k$ and $P(j) \in S_i$ **do**
        Set $n_j := n_j - 1$.
        if $n_j = 0$ **then** $Q := Q \cup \{P(j)\}$.
    Set $k := k + 1$, $\mathcal{F}_k := Q$.
return $\mathcal{F}_0, \mathcal{F}_1, \ldots$.

Algorithm 5 Multi-objective sorting mechanism (alternative)

Require: Population sizes $\lambda \in \mathbb{N}$. Population $P_t \in \mathcal{Y}^\lambda$. Fitness function f.
1: Sort P_t into $P_t^1, P_t^1, \ldots$ where P_t^1 containing all individuals with the highest fitness f, P_t^2 containing all individuals with the 2nd highest fitness $f, \ldots$.
2: **for** $i = 1, \ldots, \lambda$ **do**
3: Set $\hat{\chi} := -\infty$.
4: **for** $Q = P_t^1, P_t^1, \ldots$ **do**
5: Find (x', χ') which is the element with the highest χ in Q.
6: **if** $Q \neq \emptyset$ and $\chi' > \hat{\chi}$ **then**
7: $P_t(i) := (x', \chi')$ and $\hat{\chi} := \chi'$.
8: Pop (x', χ') from Q.
9: Break.
10: **return** P_t.

Algorithm 6 (μ, λ) selection

Require: Population size $\lambda \in \mathbb{N}$. Parameter $\mu \in [\lambda]$[3].
1: $I_t \sim \text{Unif}([\mu])$.
2: **return** I_t.

Algorithm 7 Fitness-first sorting mechanism [7]

Require: Population sizes $\lambda \in \mathbb{N}$. Population $P_t \in \mathcal{Y}^\lambda$. Fitness function f.
1: Sort P_t such that $P_t(1) \succeq \cdots \succeq P_t(\lambda)$, according to
2: $(x, \chi) \succeq (x', \chi') \Leftrightarrow f(x) > f(x') \vee (f(x) = f(x') \wedge \chi \geq \chi')$.
3: **return** P_t.

[3] For any $n \in \mathbb{N}$, we define $[n] := \{1, \ldots, n\}$

Appendix 2 Omitted statistical results of experiments

Table 2: Statistical results of experiments on random NK-Landscape problems. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and MOSA-EA.

k	**Stat.**	**RS**	**cGA**	**UMDA**	**RLS**	**SA-$(1,\lambda)$EA**	$(1+1)$**EA**	**FastGA**	$(1+(\lambda,\lambda))$**GA**	(μ,λ)**EA**	**3-tour.EA**	**MOSA-EA**
5	Median	66.6591	72.9964	74.8631	71.3547	74.8418	76.6613	76.9230	79.2846	78.2089	79.2846	**79.2846**
	p-value	2.1e-22	2.3e-04	0.0213	6.5e-08	0.0226	0.2668	0.4215	0.9299	0.7985	0.8805	-
10	Median	66.4442	69.5499	73.2968	68.3100	71.0248	75.5792	76.1340	77.1520	79.2680	78.7832	**82.5270**
	p-value	2.6e-34	1.5e-26	2.0e-15	2.6e-34	3.5e-34	5.0e-18	1.1e-12	2.2e-09	0.0030	0.0063	-
15	Median	66.2055	66.5517	70.9576	66.4446	67.8968	73.7253	74.2253	74.6407	76.0777	76.9053	**80.4417**
	p-value	2.6e-34	2.6e-34	5.5e-22	2.6e-34	2.6e-34	2.6e-34	1.8e-33	5.2e-33	1.3e-20	1.1e-17	-
20	Median	66.1233	64.4191	69.6786	64.9865	66.0533	72.8025	72.8783	73.0882	74.2580	75.3662	**78.5247**
	p-value	2.6e-34	2.6e-34	7.0e-31	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	4.0e-33	1.2e-31	-
25	Median	66.2207	63.1222	68.5683	64.3685	65.1886	70.8648	71.7564	71.9623	73.4398	74.8115	**77.5024**
	p-value	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	1.4e-33	-

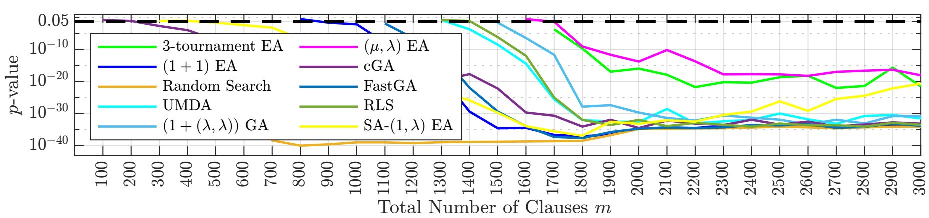

Fig. 11: The p-values of Wilcoxon rank-sum tests between the algorithms and the MOSA-EA on 100 random k-Sat instances. The y-axis is log-scaled.

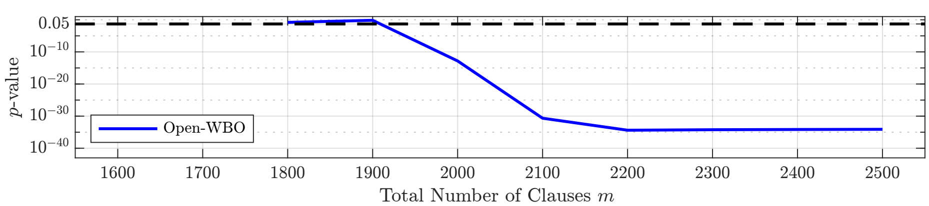

Fig. 12: The p-value of Wilcoxon rank-sum test between Open-WBO and the MOSA-EA on 100 random k-Sat instances. The y-axis is log-scaled.