A Novel Hybrid Evolutionary Strategy and its Periodization with Multi-objective Genetic Optimizers

Paul Kaufmann, Tobias Knieper and Marco Platzner

Abstract— **This work investigates the effects of the periodization of local and global multi-objective search algorithms. To this, we introduce a model for periodization and define a new multi-objective evolutionary algorithm adopting concepts from Evolutionary Strategies and NSGAII. We show that our method, especially when periodized with standard multi-objective genetic algorithms, excels for the evolution of digital circuits on the Cartesian Genetic Programming model as well as on some standard benchmarks such as the ZDT6.**

I. Introduction

Multi-criteria optimization is an important task in many application domains. With x as vector of decision variables and f as vector of objective functions, the task of concurrently minimizing multiple objective functions can be stated as:

$$min_{x \in \mathbb{R}^n} f(x) = (f_1(x), f_2(x), \dots, f_k(x))^T : \mathbb{R}^n \rightarrow \mathbb{R}^k$$

To be able to minimize, a vector-valued order relation has to be defined on the images of x under f. A prominent example for such a relation is Pareto-dominance stating for two decision vectors $x_1, x_2 \in \mathbb{R}^n$: x_1 dominates x_2 if

$$x_1 \prec x_2 \Leftrightarrow \forall i : f_i(x_1) \leq f_i(x_2) \wedge \exists j : f_j(x_1) < f_j(x_2).$$

Additionally, weak Pareto-dominance is defined as:

$$x_1 \preceq x_2 \Leftrightarrow \forall i : f_i(x_1) \leq f_i(x_2).$$

If $x_1 \preceq x_2$ and $x_2 \preceq x_1$, it follows that $f(x_1) \overset{!}{=} f(x_2)$. If neither condition holds, x_1 and x_2 are incomparable.

In recent years, several Pareto-based multi-objective genetic algorithms have been presented that show excellent performance when optimizing for multiple and often conflicting goals. In our work, we are interested in multi-criteria optimization of digital hardware [1], [2] using the Cartesian Genetic Programming model [3] to represent circuits. Experience shows that for this specific application domain global multi-objective genetic optimizers can be rather slow, especially when compared with local Evolutionary Search (ES) techniques [4]. However, in the presence of multiple objectives local search techniques typically work with fitness functions that are linear combinations of the single objectives, rather than with the Pareto-based principle.

In this paper, we describe a *periodization technique* that alternates the execution of global and local evolutionary optimizers. Our technique is inspired by the earlier work of Schwiegelshohn [5] and Ierardi [6] on Periodic Sorting Networks (PSN) where different sorting meshes are applied alternately on the input data. We extend this idea by proposing a *periodized execution model* that blends algorithm properties, functionality and convergence behavior in a simple and straight-forward way. A typical example is the combination of global search for the early phase of an optimization run with local search for the final phase [7], [8].

We introduce a novel local search algorithm termed *hybrid Evolutionary Strategy (hES)*, a synthesis of a standard ES and a Pareto set preserving technique, and investigate its performance when periodized with multi-objective genetic optimizers NSGAII and SPEA2. The novelty of hES is that it applies a $\mu + \lambda$ ES on the Pareto-dominant individuals obtained by a multi-objective genetic algorithm while keeping diversity in and avoiding deterioration of the Pareto set.

The remainder of the paper is structured as follows: Section II presents related work on hybrid evolutionary search techniques. Our periodization model is defined in Section III, followed by a discussion of the novel hybrid Evolutionary Strategy (hES) in Section IV. Sections V and VI define the benchmarks and fitness metrics used in our experiments, and Section VII presents the results. Finally, Section VIII concludes the paper.

II. Related Work

Hernández-Díaz et al. [7] presented a two-stage multi-objective evolutionary algorithm based on Differential Evolution (DE) and Rough Sets (RS) theory. In the first stage, the authors employed a fast converging multi-objective DE scheme to compute an initial Pareto front approximation. In the second stage, they improve the Pareto set diversity using RS theory for detecting loosely-covered regions. The algorithm's performance is verified on the standard ZDT{1,. . . ,6} and DTLZ{1,. . . ,4} benchmarks [9], [10]. To compare the computed Pareto sets, the authors used three metrics, the unary additive epsilon indicator [11], the standard deviation of crowding distances (SDC) [12] and the space covered by a Pareto set [13]. The proposed algorithm generally outperformed NSGAII, except on the DTLZ2 and DTLZ4 benchmarks using the SDC metric.

Talbi et al. [8] proposed a similar two-stage approach and

Paul Kaufmann is with the Department of Computer Science, University of Paderborn, 33098 Paderborn, Germany (phone: +49 5251 60 5398; email: paul.kaufmann@uni-paderborn.de).

Tobias Knieper is with the Department of Computer Science, University of Paderborn, 33098 Paderborn, Germany.

Marco Platzner is with the Department of Computer Science, University of Paderborn, 33098 Paderborn, Germany (phone: +49 5251 60 5250; email: platzner@uni-paderborn.de).

used a multi-objective genetic algorithm (GA) to calculate a first rough Pareto front approximation, followed by a local search technique for refining the approximation. The authors observed improved behavior to a GA-only approach as soon as the complexity of the test problems increases.

Zapotecas et al. [14] presented a hybrid approach combining the global optimizer NSGAII with the local optimizers of Nelder and Mead [15] and the golden section method. The authors enhanced the exploratory NSGAII by local search methods in order to reduce the number of fitness evaluations. The hybrid algorithm was compared to standard NSGAII on continuous benchmarks ZDT{1,...,4}, ZDT6 and DTLZ{1,2} using the metrics inverted generational distance [16], spacing [17] and coverage indicator [18]. With the exception of the ZDT6 and DTLZ{1,2} benchmarks in combination with the spacing metric, the hybrid algorithm outperformed NSGAII.

Harada et al. [19] analyzed *GA-and-LS* and *GA-then-LS* schemata in which local search is applied either after each generation or after a completed run of a genetic algorithm. The authors concluded that *GA-then-LS* is superior to *GA-and-LS* on multiple benchmarks and used generational and Pareto-optimal front distances [12] for comparison.

Close to our approach is the work of Ishibuchi et al. [20], [21]. The authors discuss various implementations of standard multi-objective optimizers such as SPEA2 and NSGAII combined with local search. The key idea of their approach is to periodically swap between different optimizers during a run. The authors conclude that the performance of such a hybrid optimizer is sensitive to the balance between global and local search. However, by carefully weighting global and local search strategies the periodized hybrid optimizer outperformed the standard multi-objective optimizer.

In our work, we investigate a novel hybrid Evolutionary Strategy (hES) and its periodization with the multi-objective optimizers NSGAII and SPEA2 in a *GA-and-LS* manner.

III. The periodization Model

In this section, we formalize the periodized execution model combining global and local search techniques. Let $A = (a_1, a_2, \ldots, a_n)$ be the set of algorithms used in the periodization. As an illustrative example, consider $A = \{\mathrm{GA1}, \mathrm{GA2}, \mathrm{LS}\}$. For a hypothetical periodized algorithm that executes a single step/generation of GA1, followed by two steps of LS, then a single step of GA2 and two steps of LS, the index sequence I for the algorithm selection is given by (a_1, a_3, a_2, a_3), and the repetition sequence F is $(f_1, f_2, f_3, f_4) = (1, 2, 1, 2)$.

While in this specific example, F is a vector of constants, the number of repetitions can be adaptively adjusted based on the history $\mathcal{H}$ of the optimization run. In particular, global GAs with fast convergence in the beginning of an optimization run could be repeated more often in the early search phases, while local search algorithms that excel in improving nearly optimal Pareto sets could be used more intensively in the final optimization phase.

With t as the current generation number, $\mathcal{H}$ as the history of the current optimization run, $A = (a_1, a_2, \ldots, a_n)$, $n \in \mathbb{N}$ as the set of algorithms used in the periodization, $I = (i_1, i_2, \ldots, i_m)$, $m \in \mathbb{N}$, $i_k \in (1, 2, \ldots, n)$ as the set of indices for the selected algorithms in the execution sequence and $F = (f_1, f_2, \ldots, f_m)$, $f_k(t, \mathcal{H}) \to \mathbb{N}$ as the number of repetitions for the algorithms in I, the complete *periodized execution model* P can be defined as:

$$\begin{aligned} P &= A_I^F \\ &= (a_{i_1}^{f_1(t,\mathcal{H})}, a_{i_2}^{f_2(t,\mathcal{H})}, \ldots, a_{i_m}^{f_m(t,\mathcal{H})}). \end{aligned}$$

The history $\mathcal{H}$ can be large if considering the complete information of an optimization run, or more compact if considering, for example, only the space dominated by the current Pareto-set. In our experiments, we choose $f_k(t, \mathcal{H})) := f_k(t) \equiv$ *const*.

IV. The Hybrid ES

Evolutionary Strategies (ES) were introduced by Rechenberg [22] and Schwefel [23] and rely solely on a mutation operator to discover the search space. The $\{\mu \overset{,}{+} \lambda\}$ ES uses μ parents to create λ offspring individuals and selects μ new parents from all individuals in case of a '**+**' variant or from the new individuals in case of the '**,**' variant, respectively.

Our new hES local search technique is a $1 + \lambda$ ES designed for periodization with multi-objective genetic optimization algorithms. In particular, we include two concepts from the Elitist Non-dominated Sorting GA (NSGAII) [24] in hES: the fast non-dominated sorting and the crowding distance as diversity preserving metric. Fast non-dominated sorting calculates the different non-domination sets for a set of points in the objective space. The crowded distance for a point is defined as the hyper volume of a cuboid bounded by the adjoining points in the same non-domination set. Consequently, the crowded distance creates an order, denoted as $\prec_n$, on the points of a non-domination set. Our hES local search technique uses fast non-dominated sorting to select parents for offspring creation, and crowded distances to decide which of the offspring individuals might be skipped in order to keep the Pareto set diverse.

In summary, the key ideas for our hES algorithm are:

1) A local search algorithm is executed for every element of a given set of solutions. Exactly one individual from a parent and its offspring individuals proceeds to the next population. Offspring individuals that are mutually non-dominant to their parents but have a different Pareto vector are skipped.

2) Genetic drift, as presented by Miller in [3], is achieved by skipping a parent if at least one of its offspring individuals holds an equal Pareto vector.
3) Parents and offspring individuals are partitioned into non-domination sets and the new parents are selected using the global crowding distance metric.

Algorithm 1 shows the pseudocode of our hES implementation `hES-step`. The offspring individuals are generated by Lines 1 to 4 calling the `ES-generate` procedure outlined in Algorithm 2. Algorithm 1 proceeds with the concatenation of parents and offspring individuals by calling the `add-replace` procedure listed in Algorithm 3. The `add-replace` procedure clones the parent population and successively adds all offspring individuals that have a unique Pareto vector to this population. An offspring with a Pareto vector identical to its parent replaces the parent.

The `hES-step` algorithm then partitions the concatenated set R_t into non-dominated sets $\mathcal{F}_i$ using the NSGAII method `fast-non-dominated-sort` in Line 6. After that, starting with the dominant set $\mathcal{F}_1$, the algorithm partitions the set by the parents into $G = \{G_1, G_2, \ldots\}$. That means, G_i is defined as $\mathcal{F}_1 \cap S$ where $S \subseteq P_t \cup Q_t$ contains a parent p and its offspring individuals and p is parent of G_i. Should a non-empty set G_i not contain the parent p, one of the least crowded points of G_i is selected to proceed to the next generation. Otherwise, the parent proceeds to the next generation. All other elements of S are discarded by `hES-step`.

V. Benchmarks

We experiment with several benchmarks to compare hES and the periodized variants of hES, NSGAII and SPEA2. First, we use the standard benchmarks for multi-objective algorithms DTLZ{2,6} and ZDT6. These benchmarks are available with the PISA toolbox [25] and are described in [9] [10]. Second, we compare our algorithms on the evolution of digital circuits, i.e., 2×2 adders and multipliers, using Cartesian Genetic Programming (CGP) [3] as the hardware representation model. Figure 1 illustrates the CGP phenotype. Besides the functional quality of the digital circuit, which in this case is set as a constraint, we select the circuit's area and speed to define a multi-objective benchmark [26].

In our experiments we execute 20 repetitions for every combination of benchmark and algorithm. The population size is set to 50 for the parent and offspring populations and to 100 for the archive. For the hES algorithm, the offspring population amounts to 32 individuals for the CGP benchmark. For the other benchmarks hES is configured to have one offspring per parent.

Table I presents the configuration of the benchmarks DTLZ{2,6} and ZDT6. For these benchmarks, NSGAII and SPEA2 employ the SBX crossover operator [27]. The

Algorithm 1: `hES-step(`λ,P_t`)` - perform a single hES step

```
   Input: $\lambda$, archive $P_t$
   Output: new archive $P_{t+1}$
 1 $Q_t \leftarrow \emptyset$
 2 foreach $p \in P_t$ do
 3     $Q_t \leftarrow Q_t \cup$ ES-generate($p, \lambda$)
 4 end
 5 $R_t \leftarrow$ add-replace($P_t, Q_t$)
 6 $\mathcal{F} \leftarrow$ fast-non-dominated-sort($R_t$)
 7 $P_{t+1} \leftarrow \emptyset$
 8 foreach $\mathcal{F}_i \in \mathcal{F}$ do
 9     crowding-distance-assignment($\mathcal{F}_i$)
10     $\mathcal{G} \leftarrow$ group-set-by-parent($\mathcal{F}_i$)
11     foreach $\mathcal{G}_j \in \mathcal{G}$ do
12         if parent of $\mathcal{G}_j$ not already replaced then
13             if parent($\mathcal{G}_j$) $\in \mathcal{G}_j$ then
14                 $P_{t+1} \leftarrow P_{t+1} \cup \{$parent($\mathcal{G}_j$)$\}$
15             else
16                 sort($\mathcal{G}_j, \prec_n$)
17                 $P_{t+1} \leftarrow P_{t+1} \cup \{\mathcal{G}_j[0]\}$
18             end
19             mark parent of $\mathcal{G}_j$ as replaced
20         end
21     end
22 end
```

Algorithm 2: `ES-generate(`p,λ`)` - generate λ offspring individuals

```
Input: parent $p$, number of offspring individuals $\lambda$
Output: offspring set $Q$
$Q \leftarrow \emptyset$
for $i \leftarrow 1$ to $\lambda$ do
    $p' \leftarrow$ mutate($p$)
    $Q \leftarrow Q \cup \{p'\}$
end
```

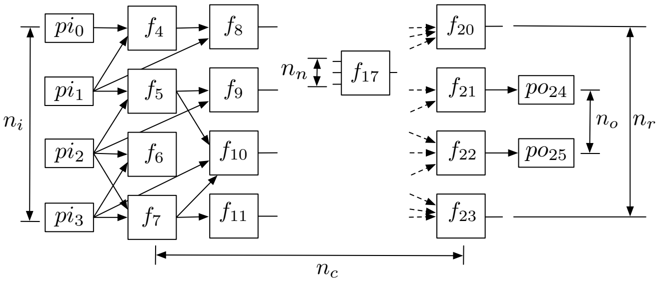


Fig. 1: Cartesian Genetic Programming (CGP) encodes a two dimensional grid of functional units connected by feed forward wires, thus forming a directed acyclic graph. The CGP model is parametrized with the number of primary inputs n_i and outputs n_o, number of rows n_r and columns n_c, number of functional block inputs n_n, the maximal length of a wire l and the functional set F that can be computed by the nodes.

Algorithm 3: `add-replace(`P,Q`)` - return copy of P joined by Q, replace parents in P by offspring individuals in Q with equal Pareto vectors, avoid adding multiple offspring individuals with equal Pareto-vectors.

```
Input: sets $P$, $Q$
Output: set $R$
$R \leftarrow P$
foreach $q \in Q$ do
    if $\nexists r \in R : r \preceq q \wedge q \preceq r$ then
        $R \leftarrow R \cup \{q\}$
    end
    if $\exists r \in R :$
    $r \preceq q \wedge q \preceq r \wedge \mathtt{parent}(\{q\}) == r$ then
        $R \leftarrow R \cup \{q\}$
        $R \leftarrow R \backslash \{r\}$
    end
end
```

optimization runs are stopped after 10.000 fitness evaluations.

Table II shows the configuration of the CGP representation model for the 2×2 adder and multiplier benchmarks. We limit the functional set to the Boolean functions presented in Table III. The number of primary inputs and outputs are set to three and four for the adder and multiplier benchmarks, respectively. The evolution is stopped after 400.000 fitness evaluations for the adder benchmark, and after 1.600.000 fitness evaluations for the more complex multiplier benchmark.

TABLE I: DTLZ2, DTLZ6, KUR and ZDT6 benchmark configuration

dimension	2
number of decision variables	100
individual mutation probability	1.0
individual recombination probability	1.0
variable mutation probability	1.0
variable swap probability	0.5
variable recombination probability	1.0
eta mutation	20
eta recombination	15
use symmetric recombination	1.0

TABLE II: CGP representation model configuration

dimension	3
mutation probability	0.1
crossover probability	0.5
n_r	1
n_c	200
l	200
n_n	2

VI. Performance Assessment

To analyze the performance of the proposed multi-objective optimizers, we need to compare the calculated Pareto sets. In this paper we employ two methods: the ranking of Pareto sets by a quality indicator and the analysis of the mean Pareto set, attained during multiple runs. Both methods are described by Knowles et al. [28] and are also implemented in the PISA toolbox by Bleuler et al. [25].

TABLE III: CGP configuration: functions in F [3]

Number	Function	Number	Function
0	0	10	$a \oplus b$
1	1	11	$a \oplus \overline{b}$
2	a	12	$a + b$
3	b	13	$a + \overline{b}$
4	$\overline{a}$	14	$\overline{a} + b$
5	$\overline{b}$	15	$\overline{a} + \overline{b}$
6	$a \cdot b$	16	$a \cdot \overline{c} + b \cdot c$
7	$a \cdot \overline{b}$	17	$a \cdot \overline{c} + \overline{b} \cdot c$
8	$\overline{a} \cdot b$	18	$\overline{a} \cdot \overline{c} + b \cdot c$
9	$\overline{a} \cdot \overline{b}$	19	$\overline{a} \cdot \overline{c} + \overline{b} \cdot c$

A. Quality Indicators

To compare Pareto sets calculated by benchmarked algorithms, Zitzler et al. [11] introduced the concept of a Quality Indicator (QI) as a function mapping a set of Pareto sets to a set of real numbers. Under QI, the Pareto sets define a relation on the Pareto set quality. In our work, we use the unary additive epsilon indicator $I^1_{\epsilon+}$. It is based on the binary additive epsilon indicator $I_{\epsilon+}$ which is defined for two Pareto sets A and B as:

$$I_{\epsilon+}(A, B) = \inf_{\epsilon \in \mathbb{R}} \{\forall b \in B \; \exists a \in A : a \preceq_{\epsilon+} b\}.$$

Here, the relation $\preceq_{\epsilon+}$ is defined as $a \preceq_{\epsilon+} b \Leftrightarrow \forall i : a_i \leq \epsilon + b$. For a reference Pareto set R, the unary additive epsilon indicator $I^1_{\epsilon+}$ can be now derived as

$$I^1_{\epsilon+}(A) = I_{\epsilon+}(A, R).$$

Following Knowles et al. [28], we use the non-parametric Kruskal-Wallis test [29] to statistically evaluate sequences of quality numbers. The Kruskal-Wallis test differentiates between the null hypothesis $H_0 =$"The distribution functions of the sequences are identical" and the alternative hypothesis $H_A =$"At least one sequence tends to yield better observations than another sequence". In case the test rejects H_0, we provide for all sequence pairs the one-tailed p-value. Table IV presents an example: for an algorithm tuple $(A_{\text{row}}, A_{\text{col}})$ a p-value equal or below α indicates a lower mean for A_{row}. Thus, one can conclude for Table IV that A_1 outperforms A_2 and A_3, and A_3 outperforms A_2. In our experiments, we configure the significance level α to 0.01.

B. Empirical Attainment Functions

An additional way of interpreting the results of multi-objective optimizers is to look at the Pareto points that are covered, i.e., weakly dominated, with a certain probability during the multiple repetitions of an optimization algorithm.

TABLE IV: Interpretation of the Kruskal-Wallis test: Given the Kruskal-Wallis test rejects H_0, a dot denotes a p-value higher than α.

	A_1	A_2	A_3
A_1	-	0.002	0.007
A_2	·	-	·
A_3	·	0.003	-

All Pareto points that have been reached in $x\%$ of the runs are referred to as the $x\%$-attainment. The attainment allows for a direct graphical interpretation as shown in the examples of Figures 2a, 2b and 3.

In order to statistically compare the attainments we use the two-tailed Kolmogorov-Smirnov test [30]. It distinguishes between H_0="Sequences A and B follow the same distribution" and H_A="Sequences A and B follow different distributions". Table V contains exemplary results for the Kolmogorov-Smirnov test. It can be interpreted as: A_1 differs significantly from A_2 and A_3. In our experiments, we configure the significance level α to 0.05.

TABLE V: Interpretation of the Kolmogorov-Smirnov test: A dot denotes an accepted H_0 hypothesis at the given α.

	A_1	A_2	A_3
A_1	-	*	*
A_2	*	-	·
A_3	*	·	-

VII. Results

In this section, we first compare the performance of NSGAII, SPEA2, hES and periodized hES with NSGAII on the DTLZ{2,6}, ZDT6 and 2 × 2 adder and multiplier benchmarks. Then, we investigate our periodization scheme more thoroughly on the ZDT6 benchmark. For the sake of a more compact experiment description, we use the following abbreviations:

NSGAII	→ n
SPEA2	→ s
hES	→ h

A. Performance and Effect of hES

To examine the effect of local search, we first execute the standard NSGAII and SPEA2 for a given benchmark in order to determine the reference performance. Then, we increase the influence of local search step-by-step by periodizing NSGAII with hES until only hES is executed. In terms of our periodization model, we investigate the six periodization schemes: (n), (s), (nh), (nh^4), (nh^{10}) and (h). The results are as follows:

1) DTLZ{2,6}: Table VI shows the results of the Kruskal-Wallis test applied to DTLZ2 and DTLZ6 with respect to the unary additive epsilon indicator $I^1_{\epsilon+}$ at the significance level α of 1%. For both benchmarks, NSGAII and SPEA2 significantly dominate the algorithm combinations of NSGAII and hES. While performing similar on DTLZ2, SPEA2 is better than NSGAII on DTLZ6.

The central observation in this experiment is that the quality of the Pareto set degrades with increasing influence of local search. Starting with the alternation of NSGAII and hES, the Kruskal-Wallis test shows falling performance when increasing the number of iterations of hES. The hES-only experiment results in the worst performance over all algorithms. The Kruskal-Wallis test results are confirmed by the graphical interpretation of the 75% attained Pareto sets in Figure 2. The Kolmogorov-Smirnov test reveals a significant difference between all attained Pareto sets at the significance level α of 5%.

2) ZDT6: Table VII presents the results of the Kruskal-Wallis test applied to the ZDT6 benchmark with respect to the unary additive epsilon indicator $I^1_{\epsilon+}$ at the significance level α of 1%. The table shows that nh^4 performs significantly better on ZDT6 than all other algorithms. Furthermore, nh outperforms all other algorithm combinations except nh^4.

Figure 3 shows the 75%-attainment of the ZDT6 benchmark and confirms the results of the Kruskal-Wallis test. The Kolmogorov-Smirnov test at $\alpha = 5\%$ reveals that NSGAII and SPEA2 are no different, but all other algorithms show significant deviations.

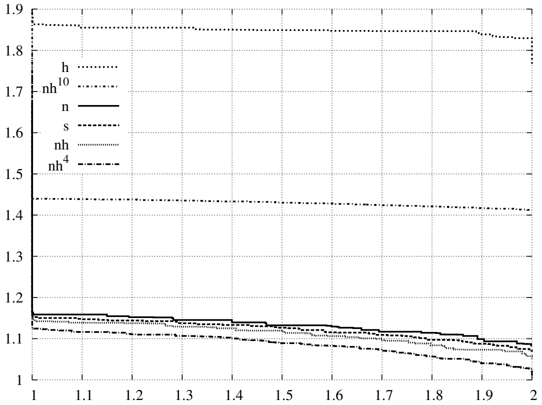
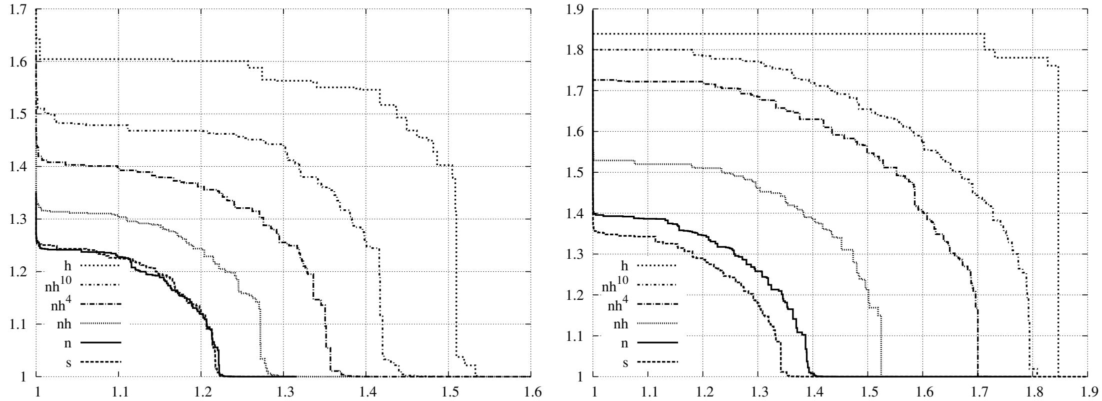


Fig. 3: 75%-attainment for the ZDT6 benchmark

3) 2 × 2 adder and multiplier: In contrast to the previous benchmarks, the adder and multiplier benchmarks have an additional optimization dimension: the functional quality of a circuit. In our work we typically focus on hardware functions with continuous fitness measurements such as, for example, classification accuracy [31] and cache performance [4]. In this paper, we use arithmetic functions with binary fitness measurements as they are commonly employed benchmarks for the comparison of evolutionary algorithms within the CGP context. We treat the circuit's functional quality as an ordinary objective function but are interested in evolving correct circuits with various combinations of area and speed.

TABLE VI: Kruskal-Wallis test applied to the DTLZ2 and DTLZ6 benchmarks using the unary additive epsilon indicator $I^1_{\epsilon+}$ at the significance level α of 1%.

	DTLZ2						DTLZ6					
	n	s	h	nh	nh^4	nh^{10}	n	s	h	nh	nh^4	nh^{10}
n	-	·	5.5_{10}^{-61}	1.4_{10}^{-20}	7.9_{10}^{-37}	1.8_{10}^{-50}	-	·	9.6_{10}^{-57}	1.2_{10}^{-13}	3.1_{10}^{-33}	1.1_{10}^{-42}
s	·	-	1.6_{10}^{-60}	8.2_{10}^{-20}	3.7_{10}^{-36}	6.3_{10}^{-50}	9.4_{10}^{-08}	-	2.8_{10}^{-64}	1.8_{10}^{-26}	4.9_{10}^{-44}	3.8_{10}^{-52}
h	·	·	-	·	·	·	·	·	-	·	·	·
nh	·	·	2.6_{10}^{-43}	-	1.2_{10}^{-11}	5.4_{10}^{-29}	·	·	3.1_{10}^{-43}	-	1.9_{10}^{-14}	1.3_{10}^{-25}
nh^4	·	·	2.4_{10}^{-28}	·	-	5.5_{10}^{-12}	·	·	1.8_{10}^{-25}	·	-	2.3_{10}^{-06}
nh^{10}	·	·	5.2_{10}^{-11}	·	·	-	·	·	2.5_{10}^{-14}	·	·	-



(a) (b)

Fig. 2: 75%-attainment for the DTLZ2 (a) and DTLZ6 (b) benchmarks

TABLE VII: Kruskal-Wallis test applied to the ZDT6 benchmark using the unary additive epsilon indicator $I^1_{\epsilon+}$ and a significance level α of 1%.

	n	s	h	nh	nh^4	nh^{10}
n	-	·	1.9_{10}^{-10}	·	·	·
s	·	-	2.6_{10}^{-12}	·	·	·
h	·	·	-	·	·	·
nh	1.8_{10}^{-04}	2.7_{10}^{-03}	7.6_{10}^{-19}	-	·	1.9_{10}^{-05}
nh^4	3.6_{10}^{-10}	2.1_{10}^{-08}	6.8_{10}^{-26}	1.4_{10}^{-03}	-	1.7_{10}^{-11}
nh^{10}	·	·	3.7_{10}^{-09}	·	·	-

Table IX summarizes the number of runs that resulted in functionally correct solutions. The main observation, which was the motivation for this work in the first place, is that local search is indeed beneficial for the CGP domain and the evolution of digital circuits. With more frequent use of hES more correct circuits are being evolved. This insight is also partly confirmed by the Kruskal-Wallis test for the 2×2 multiplier presented in Table VIII. There, the reference multi-objective optimizer SPEA2 manages to outperform only the NSGAII and the hES periodization.

For the 2×2 adder, the Kruskal-Wallis test reveals no significant differences between the algorithms at $\alpha = 1\%$.

TABLE VIII: Kruskal-Wallis test applied to the 2×2 multiplier benchmark using the unary additive epsilon indicator $I^1_{\epsilon+}$ and a significance level α of 1%.

	n	s	h	nh	nh^4	nh^{10}
n	-	·	·	·	·	·
s	1.2_{10}^{-10}	-	7.2_{10}^{-05}	·	·	·
h	1.6_{10}^{-03}	·	-	·	·	·
nh	7.8_{10}^{-09}	·	1.3_{10}^{-03}	-	·	·
nh^4	1.0_{10}^{-06}	·	·	·	-	·
nh^{10}	2.5_{10}^{-07}	·	·	·	·	-

B. Periodization of ZDT6

For a more detailed analysis of periodized local and global search algorithms, we fixate on the ZDT6 benchmark and apply 2- and 3-tuple permutations of the NSGAII, SPEA2 and hES algorithms on it. All experiments are repeated for 100 times and the execution is stopped after 200 generations.

Table X shows the result for 2-tuple combinations of

TABLE IX: The number of circuits with perfect functional quality evolved by the various algorithms for the 2 × 2 adder benchmark within 400.000 fitness evaluations and 2×2 multiplier within 1.600.000 fitness evaluations.

	2 × 2 add	2 × 2 mul
n	1	0
s	6	7
h	12	15
nh	7	8
$\mathtt{nh}^{4}$	9	11
$\mathtt{nh}^{10}$	11	14

NSGAII, SPEA2 and hES. The general statement of the Kruskal-Wallis test is that hES periodized with either NSGAII or SPEA2 outperforms standard NSGAII, SPEA2 and combinations of them. Interestingly, the *hES-after-SPEA2* outperforms the *hES-after-NSGAII*, while *SPEA2-after-hES* does not. This shows that the performance of this particular periodization scheme is sensitive to the initial order of the executed algorithms.

The Kolmogorov-Smirnov test confirms the results observed before. There are basically two classes of algorithms showing significantly different results, namely the class of algorithms periodized with hES, and the class of NSGAII, SPEA2 and combinations of them. In contrast to the previous test, the differences between (hs) and (nh) are now identified as significant.

Next, Table XI shows the results of 3-tuple combinations of hES with NSGAII and SPEA2. Analogous to the results achieved for the 2-tuple tests, all periodized algorithms outperform and differ from NSGAII and SPEA2 under the Kruskal-Wallis and the Kolmogorov-Smirnov tests, respectively.

VIII. Conclusion

In this paper, we investigated the periodization of multi-objective local and global search algorithms. We defined a periodized execution model and introduced the novel hybrid Evolutionary Strategy as a local search technique tailored to periodization with Pareto-based genetic multi-objective optimizers such as NSGAII and SPEA2.

The results show that for the DTLZ$\{2{,}6\}$ benchmarks, the new algorithm and its periodization with NSGAII underperforms. For ZDT6 and, most importantly, for the evolution of digital circuit benchmarks on the CGP model, the new algorithm and its periodizations are significantly better than the reference algorithms NSGAII and SPEA2. Furthermore, the periodized execution model proved to be a simple, fast and flexible approach to combine multiple optimization algorithms for merging functional and behavior properties. Thus, blending multi- and single-objective optimizers, local and global search algorithms and differently converging methods creates a new family of optimization algorithms.

Acknowledgment

This work was supported by the German Research Foundation under project number PL 471/2-3 within the priority program *Organic Computing*.

References

[1] P. Kaufmann and M. Platzner, “Toward Self-adaptive Embedded Systems: Multi-objective Hardware Evolution,” in *Proceedings of the 20th International Conference on Architecture of Computing Systems (ARCS’07)*, ser. LNCS, vol. 4415. Springer, March 2007, pp. 199–208.

[2] T. Knieper, B. Defo, P. Kaufmann, and M. Platzner, “On Robust Evolution of Digital Hardware,” in *Proceedings of the 2nd IFIP Conference on Biologically Inspired Collaborative Computing (BICC’08)*. Milano, Italy: Springer, September 2008.

[3] J. Miller and P. Thomson, “Cartesian Genetic Programming,” in *Proceedings 3rd European Conference on Genetic Programming (EuroGP’00)*. Springer, 2000, pp. 121–132.

[4] P. Kaufmann, C. Plessl, and M. Platzner, “EvoCaches: Application-specific Adaptation of Cache Mappings,” in *Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 11–18.

[5] U. Schwiegelshohn, “A Short Periodic Two-dimensional Systolic Sorting Algorithm,” in *International Conference on Systolic Arrays*. Computer Society Press, 1988, pp. 257–264.

[6] D. Ierardi, “2d-Bubblesorting in Average Time O(sqrt N lg N)*,” in *Symposium on Parallel Algorithms and Architectures (SPAA’94)*. ACM Press, 1994, pp. 36–45.

[7] A. G. Hernández-Díaz, L. V. Santana-Quintero, C. A. C. Coello, R. Caballero, and J. M. Luque, “A New Proposal for Multi-objective Optimization Using Differential Evolution and Rough Sets Theory,” in *Genetic and evolutionary computation (GECCO’06)*. ACM, 2006, pp. 675–682.

[8] E.-G. Talbi, M. Rahoual, M. H. Mabed, and C. Dhaenens, “A Hybrid Evolutionary Approach for Multicriteria Optimization Problems: Application to the Flow Shop,” in *Evolutionary Multi-Criterion Optimization (EMO’01)*. Springer-Verlag, 2001, pp. 416–428.

[9] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[10] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable Test Problems for Evolutionary Multi-Objective Optimization,” in *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*. Springer, 2005, ch. 6, pp. 105–145.

[11] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, “Performance Assessment of Multiobjective Optimizers: An Analysis and Review.” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[12] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[13] E. Zitzler and L. Thiele, “Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach,” in *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, 1999, pp. 257–271.

[14] S. Zapotecas Martínez and C. A. Coello Coello, “A Proposal to Hybridize Multi-Objective Evolutionary Algorithms with Non-gradient Mathematical Programming Techniques,” in *Parallel Problem Solving from Nature (PPSN’08)*. Springer-Verlag, 2008, pp. 837–846.

[15] J. A. Nelder and R. Mead, “A Simplex Method for Function Minimization,” *The Computer Journal*, vol. 7, no. 4, pp. 308–313, January 1965.

TABLE X: ZDT6: comparing 2-tuple combinations of NSGAII, SPEA2 and hES to NSGAII and SPEA2. Numbers and *'s denote significant events under the Kruskal-Wallis test using the unary additive epsilon indicator $I^1_{\epsilon+}$ at the significance level α of 1% and under the Kolmogorov-Smirnov test at the significance level α of 5%.

	Kruskal-Wallis								Kolmogorov-Smirnov							
	n	s	nh	hn	sh	hs	ns	sn	n	s	nh	hn	sh	hs	ns	sn
n	-	·	·	·	·	·	·	·	-	·	*	*	*	*	·	·
s	·	-	·	·	·	·	·	·	·	-	*	*	*	*	·	·
nh	$3.1_{10}{}^{-22}$	$1.1_{10}{}^{-17}$	-	·	·	·	$1.8_{10}{}^{-23}$	$8.3_{10}{}^{-22}$	*	*	-	·	*	*	*	*
hn	$9.5_{10}{}^{-28}$	$1.1_{10}{}^{-22}$	·	-	·	·	$4.1_{10}{}^{-29}$	$2.8_{10}{}^{-27}$	*	*	·	-	·	·	*	*
sh	$2.7_{10}{}^{-32}$	$7.2_{10}{}^{-27}$	$9.1_{10}{}^{-03}$	·	-	·	$9.7_{10}{}^{-34}$	$8.6_{10}{}^{-32}$	*	*	*	·	-	·	*	*
hs	$1.1_{10}{}^{-30}$	$2.2_{10}{}^{-25}$	·	·	·	-	$4.2_{10}{}^{-32}$	$3.4_{10}{}^{-30}$	*	*	*	·	·	-	*	*
ns	·	·	·	·	·	·	-	·	·	·	*	*	*	*	-	·
sn	·	·	·	·	·	·	·	-	·	·	*	*	*	*	·	-

TABLE XI: ZDT6: comparing 3-tuple combinations of NSGAII, SPEA2 and hES to NSGAII and SPEA2. Numbers and *'s denote significant events under the Kruskal-Wallis test using the unary additive epsilon indicator $I^1_{\epsilon+}$ at the significance level α of 1% and under the Kolmogorov-Smirnov test at the significance level α of 5%.

	Kruskal-Wallis								Kolmogorov-Smirnov							
	n	s	nhs	nsh	shn	snh	hns	hsn	n	s	nhs	nsh	shn	snh	hns	hsn
n	-	·	·	·	·	·	·	·	-	·	*	*	*	*	*	*
s	·	-	·	·	·	·	·	·	·	-	*	*	*	*	*	*
nhs	$3.6_{10}{}^{-10}$	$3.0_{10}{}^{-07}$	-	·	·	·	·	·	*	*	-	·	·	·	·	·
nsh	$2.4_{10}{}^{-10}$	$2.1_{10}{}^{-07}$	·	-	·	·	·	·	*	*	·	-	·	·	·	·
shn	$1.2_{10}{}^{-08}$	$5.4_{10}{}^{-06}$	·	·	-	·	·	·	*	*	·	·	-	·	·	·
snh	$4.9_{10}{}^{-13}$	$1.1_{10}{}^{-09}$	·	·	·	-	·	·	*	*	·	·	·	-	·	·
hns	$4.9_{10}{}^{-11}$	$5.6_{10}{}^{-08}$	·	·	·	·	-	·	*	*	·	·	·	·	-	·
hsn	$6.6_{10}{}^{-10}$	$4.9_{10}{}^{-07}$	·	·	·	·	·	-	*	*	·	·	·	·	·	-

[16] D. A. Van Veldhuizen, "Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations," Ph.D. dissertation, Department of Electrical and Computer Engineering. Airforce Institute of Technology, Wright-Patterson AFB, OH, 1999.

[17] J. R. Scott, "Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization," Master's thesis, Department of Aeronautics and Astronautics. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995.

[18] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," in *Evolutionary Computation*, vol. 8, no. 2. Cambridge, MA, USA: MIT Press, 2000, pp. 173–195.

[19] K. Harada, K. Ikeda, and S. Kobayashi, "Hybridization of Genetic Algorithm and Llocal Search in Multiobjective Function Optimization: Recommendation of GA then LS," in *Genetic and Evolutionary Computation (GECCO'06)*. ACM, 2006, pp. 667–674.

[20] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between Genetic Search and Local Search in Hybrid Evolutionary Multi-Criterion Optimization Algorithms," in *Genetic and Evolutionary Computation Conference (GECCO'2002)*. Morgan Kaufmann Publishers, July 2002, pp. 1301–1308.

[21] H. Ishibuchi and K. Narukawa, "Some Issues on the Implementation of Local Search in Evolutionary Multiobjective Optimization," in *Genetic and Evolutionary Computation (GECCO'04)*, ser. LNCS. Springer-Verlag, June 2004, pp. 1246–1258.

[22] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

[23] H.-P. Schwefel, "Evolutionsstrategie und numerische Optimierung," Ph.D. dissertation, TUB, 1975.

[24] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-Objective Optimisation: NSGA-II," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN'00)*. Springer, 2000, pp. 849–858.

[25] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA — A Platform and Programming Language Independent Interface for Search Algorithms," in *Evolutionary Multi-Criterion Optimization (EMO 2003)*, ser. Lecture Notes in Computer Science. Springer, 2003, pp. 494–508.

[26] P. Kaufmann and M. Platzner, "MOVES: A Modular Framework for Hardware Evolution," in *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS'07)*. IEEE, 5-8 Aug. 2007, pp. 447–454.

[27] K. Deb and R. Agrawal, "Simulated Binary Crossover for Continuous Search Space," in *Complex Systems*, vol. 9, 1995, pp. 115–148.

[28] J. Knowles, L. Thiele, and E. Zitzler, "A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, Tech. Rep., Feb. 2006, revised version.

[29] W. Conover, *Practical Nonparametric Statistics (Third ed.)*. John Wiley and Sons, 1999.

[30] K. J. Shaw, A. L. Nortcliffe, M. Thompson, J. Love, C. M. Fonseca, and P. J. Fleming, "Assessing the Performance of Multiobjective Genetic Algorithms for Optimization of a Batch Process Scheduling Problem," in *Evolutionary Computation*. IEEE Press, July 1999, pp. 37–45.

[31] K. Glette, T. Gruber, P. Kaufmann, J. Torresen, B. Sick, and M. Platzner, "Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control," in *Proceedings 3nd NASA/ESA Conference on Adaptive Hardware and Systems (AHS'08)*. IEEE Computer Society, 2008, pp. 32–39.