ZAVOIANU, A.-C., SAMINGER-PLATZ, S. and AMRHEIN, W. 2019. Comparative analysis of two asynchronous parallelization variants for a multi-objective coevolutionary solver. In Proceedings of the 2019 Institute of Electrical and Electronics Engineers (IEEE) congress on evolutionary computation (IEEE CEC 2019), 10-13 June 2019, Wellington, New Zealand. Piscataway: IEEE [online], article number 8790133, pages 3078-3085. Available from: https://doi.org/10.1109/CEC.2019.8790133.

Comparative analysis of two asynchronous parallelization variants for a multi-objective coevolutionary solver.

ZAVOIANU, A.-C., SAMINGER-PLATZ, S., AMRHEIN, W.

2019

Comparative Analysis of Two Asynchronous Parallelization Variants for a Multi-Objective Coevolutionary Solver

Alexandru-Ciprian Zăvoianu*, Susanne Saminger-Platz‡ and Wolfgang Amrhein§,

*School of Computing Science and Digital Media, Robert Gordon University, Aberdeen, Scotland, UK
‡Department of Knowledge-based Mathematical Systems, Johannes Kepler University Linz, Austria
§Institute for Electrical Drives and Power Electronics, Johannes Kepler University Linz, Austria

***Abstract*—We describe and compare two steady state asynchronous parallelization variants for DECMO2++, a recently proposed multi-objective coevolutionary solver that generally displays a robust run-time convergence behavior. The two asynchronous variants were designed as trade-offs that maintain only two of the three important synchronized interactions / constraints that underpin the (generation-based) DECMO2++ coevolutionary model. A thorough performance evaluation on a test set that aggregates 31 standard benchmark problems shows that while both parallelization options are able to generally preserve the competitive convergence behavior of the baseline coevolutionary solver, the better parallelization choice is to prioritize accurate run-time search adaptation decisions over the ability to perform equidistant fitness sharing.**

***Index Terms*—multi-objective optimization, asynchronous coevolution, master-slave parallelization, performance analysis**

I. Introduction

Formally, a multi-objective optimization problem (MOOP) can be defined as:

$$\text{minimize } F(x) = (f_1(x), \ldots, f_m(x))^T, \tag{1}$$

where $x \in D^n \subset \mathbb{R}^n$ and the m single-objective functions contained in F need to be optimized simultaneously and are usually conflicting. The general solution to a MOOP is a Pareto-optimal set (PS) that contains all the individual solution candidates $x^* \in D^n$ with the property that there is no single element in D^n that is better than x^* with regard to all m considered objectives. In most cases, MOOP solvers settle for discovering high-quality Pareto non-dominated sets (PNs) that provide a near-perfect approximation of the PS using a limited number of solution candidates.

Since multi-objective evolutionary algorithms (MOEAs) are designed to generate full PNs after single optimization runs, these approaches have become very successful in tackling complicated MOOPs [1]. Nevertheless, when the optimization problems stem from real world scenarios, the (fitness) evaluation of a potential solution candidate (i.e., individual) $x \in D^n$ can be very computationally-intensive due to required simulations or even on-site experimentation. This is problematic because MOEAs usually need to perform a rather high number of fitness evaluations (nfe) in order to discover useful PNs. Whenever the optimization scenario allows it, the easiest way to speed-up the MOEA-based optimization is to parallelize fitness evaluations. In its simplest form, this parallelization follows the master-slave paradigm in which:

- the process running on the master node contains the main algorithmic logic (population initialization, parent selection, application of genetic operators to generate offspring, replacement strategy, etc.);
- processes running on the slave nodes are only responsible for evaluating the fitness of one individual at a time.

Master-slave MOEAs can adopt either a classic synchronous *generation-based* parallelization or an *asynchronous* parallelization strategy that is more similar to a steady-state evolutionary cycle. The generation-based parallelization aims to preserve the $(\mu + \lambda)$ evolutionary model implicitly assumed by most solvers where the μ parent individuals of generation (i.e., iteration) t are selected from the union between the μ parent and the λ offspring individuals of generation $t-1$ after all the offspring have been previously evaluated on the slave nodes. The asynchronous parallelization, on the other hand, generates a new individual as soon as an offspring has been evaluated on a slave node so the evolutionary models becomes $(\mu + 1)$. Some of the pros and cons of asynchronous MOEAs have been investigated in [2], [3], and [4].

The analysis in [5] highlights the difference between synchronous and generation-based MOEAs starting from two simple observations:

- given a fixed wall-clock computational budget, the asynchronous parallelization will be able to compute more individuals than its generational counterpart (i.e., *quantitative aspect*);
- given a fixed nfe computational budged, the generation-based parallelization is very likely to deliver higher-quality PNs than the steady-state asynchronous parallelization across most MOEAs (i.e., *qualitative aspect*).

This dual point of view is useful because the quantitative aspects tend to depend on the software and hardware constraints of the parallel or distributed computing environment, while the qualitative aspects depend on the MOEA-MOOP interaction.

One key (empirical) finding from [5] that was confirmed in subsequent studies – e.g., [6] – was that a high level of variance in the time-wise distribution of the fitness evaluation function clearly favors the asynchronous parallelization.

A. Motivation and approach

One idea to help MOEA practitioners successfully tackle complicated industrial problems is to develop coevolutionary multi-objective solvers that focus on robustness – i.e., generally effective run-time convergence across a wide range of MOOPs when using a fixed parameterization. Examples of such solvers can be found in [7], [8], and [9]. The increase in robustness advocated by coevolutionary methods usually comes at the expense of structural simplicity, which in turn makes even a master-slave parallelization attempt challenging, as one aims to preserve the efficient convergence behavior.

The main motivation for this paper stems from the requirement to develop an asynchronous steady-state version variant of the DECMO2++ [9] coevolutionary solver. While the highly competitive and robust early-phase convergence behavior of DECMO2++ has been confirmed on comprehensive test sets featuring both benchmark and industrial multi-objective optimization problems, the structural complexity of this solver and the carefully orchestrated interactions between its components proved rather problematic when switching to an asynchronous $(\mu + 1)$ evolutionary cycle. Therefore, in Section II, an analysis of the coevolutionary interactions that characterize DECMO2++ is presented, as based on this, two asynchronous parallelization variants were developed. The two variants were thoroughly tested using the performance evaluation strategy described in Section III and the results and their implications are discussed in detail in Section IV.

II. Asynchronous Coevolutionary Multi-Objective Optimization

The architecture of the generation-based coevolutionary model we aim to efficiently parallelize is presented in Figure 1. The two steady-state asynchronous parallelization options we investigate are described in Sections II-C (equidistant fitness sharing) and II-D (sub-population-centered fitness sharing).

A. The baseline coevolutionary strategy: DECMO2++

The generation-based DECMO2++ [9] solver follows the same structural design of its predecessor [10] as it aims to integrate three different multi-objective space exploration paradigms via dedicated solution sets (i.e., sub-populations) that are coevolved during the optimization run.

The members of the first coevolved sub-population (marked by P) are obtained via the SPEA2 [11] evolutionary model that revolves around the *environmental selection operator* – a selection for survival mechanism that trims a set of individuals (e.g., the $|P|$ parent individuals of generation $t-1$ and all their offspring created at generation t) to a desired size (e.g., the $|P|$ individuals that will form the preliminary parent set of generation t). Environmental selection uses Pareto dominance as a primary selection metric and a crowding distance in

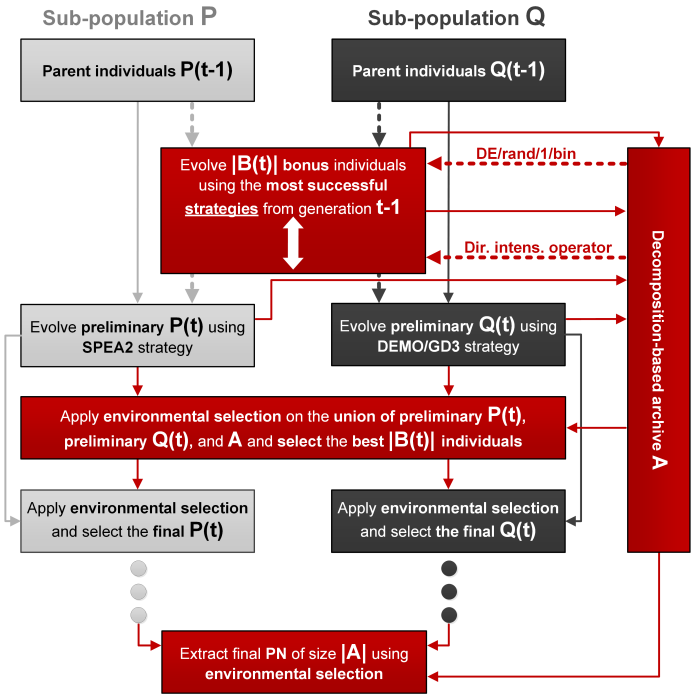

Fig. 1. The architecture of the DECMO2++ coevolutionary model.

objective space as a secondary (tie-breaking) metric. At each generation, offspring are created from the parent set using simulated binary crossover (SBX) [12] and polynomial mutation (PM) [13] – the two genetic operators that were popularized by NSGA-II [14].

The second coevolved sub-population (marked by Q) is focused on exploiting the very good performance displayed by the differential evolution paradigm [15] on continuous optimization problems with real-valued objective functions. More precisely, Q implements an evolutionary model that is very similar to the one proposed by GDE3 [16] and DEMO [17] as it maintains the usage of a Pareto-based selection for survival operator (i.e., environmental selection), but replaces the SBX and PM operators with the *DE/rand/1/bin* strategy.

DECMO2++ also features an archive (marked by A) that is maintained according to a *decomposition-based principle* that uses a weighted Tschebyscheff distance measure to ensure the uniform objective-wise spacing of stored solutions. This approach is similar to the one proposed in MOGLS [18] and mainstreamed by the success of MOEA/D [19]. While new individuals are periodically evolved directly from A, the archive has a mainly passive purpose and its main role inside the solver is two-fold:

1) preserve an accurate well-spread approximation of the Pareto-optimal set;
2) act as a run-time estimator of the comparative success of the three optimization strategies coevolved by DECMO2++. This is achieved by computing at the end of each odd-numbered (test) generation the corresponding *archive insertion ratio* of each paradigm – i.e.,

the percentage of individuals generated by each of the three strategies that warranted an archive insertion for improving the existing approximation of the PS.

The archive insertion ratios are extremely important for ensuring the robust and fast convergence behavior of DECMO2++ as they are used to:

1) actively pivot the search towards the more successful strategy. Thus, at each even-numbered generation, a number of B bonus individuals will be created by the strategy/strategies that achieved the highest insertion ratios in the previous test generation.
2) support a self-diagnostics logic that attempts to identify during the run the current stage of convergence: "early", "middle" (when the insertion ratios of both P and Q drop below 0.5), or "late" (when the insertion ratio of A constantly outperforms those of P and Q).

The search behavior of DECMO2++ is influenced by the estimated convergence state as: in the "early" stage, a specialized directional intensification operator is used when evolving individuals directly from A; in the "middle" stage, the number of bonus individuals is halved; in the "late" stage, the bonus individuals are created at every generation directly from A.

The main population-size parameter of DECMO2++ is the size of the archive (i.e., $|A|$) while the sizes of the various sets of individuals used inside the coevolutionary model are obtained using (2).

$$\begin{cases} |B| = \begin{cases} \frac{|A|}{5}, \text{ if convergence stage} = \text{“early”} \\ \frac{|A|}{10}, \text{ if convergence stage} \neq \text{“early”} \end{cases} \\ |P| = |Q| = \frac{|A|-|B|}{2} \end{cases} \quad (2)$$

As with most coevolutionary methods, efficient fitness sharing is crucial for enabling an effective integration of the coevolved components. The fitness sharing mechanism used by DECMO2++ at the end of each generation is highly elitist as it entails a two-fold application of the environmental selection operator. First, the operator is used to extract a total number of $|B|$ elite individuals, from the union of P, Q, and A. Afterwards, environmental selection is also used to integrate these elite individuals in sub-populations P and Q.

B. The dilemma of an asynchronous DECMO2++ variant

It is very important to note that the (generation-based) DECMO2++ model intrinsically determines three perfectly *synchronized coevolutionary interactions / constraints* between key parts of the optimization strategy, regardless of the level of time-wise variance related to fitness evaluation:

CI1 The fitness sharing stages are tightly linked with the archive insertion ratios as the latter are always computed between two consecutive fitness sharing stages. For example, the archive insertion ratios are reset directly before starting the computation of generation t, they are calculated using all the newly generated individuals of generation t, they are used by the self-diagnostics logic of generation t and, finally, for determining the bonus allocation of generation $t+1$.

CI2 The fitness sharing stages are equidistant with regard to the total number of evaluated individuals (i.e., they are performed after every batch of $|A|$ newly evaluated individuals).

CI3 The generation-based fitness sharing stages always allow for a predetermined number of individuals to be evaluated and used for determining archive insertion ratios for each DECMO2++ sub-population: e.g., between the fitness sharing stages of generations no. t and no. $t+1$, DECMO2++ will evaluate exactly L_P individuals generated using the SPEA2 evolutionary model, exactly L_Q individuals generated using the GDE3 / DEMO evolutionary model, and exactly L_A individuals evolved directly from the decomposition-based archive A.

Using a set of limited but systematic benchmark tests spread across 31 MOOPs, we determined that the synchronous coevolutionary interaction between the fitness sharing stages and computation and usage of archive insertion ratios (i.e., CI1) must be maintained when aiming for asynchronous evaluation in order to preserve the generally robust convergence behavior exhibited by DECMO2++. This should not come as a surprise since the orchestrated interaction between these two features defines the active run-time search adaptation mechanism that truly sets DECMO2++ apart from other coevolutionary multi-objective optimization approaches like [8] and [20].

The (steady-state) asynchronous evolutionary model makes it impossible to simultaneously maintain both the CI2 and CI3 coevolutionary interactions, especially when considering a large time-wise variance of the fitness evaluation function. This is because the strict enforcement of both CI2 and CI3 (e.g., by introducing waiting times on the master process) means that (3) must hold and this would transform the asynchronous evolutionary model into a fully synchronous one (i.e., generation-based with a generation size of $|A|$):

$$L_P + L_Q + L_A = |A| \quad (3)$$

However, it is possible to simultaneously couple either CI1 with CI2 (as described in Section II-C) or CI1 with a softer version of CI3 (as described in Section II-D) while preserving the basic (steady-state) asynchronous parallelization schema investigated in [10] and [6].

C. Equidistant fitness sharing

When opting to simultaneously enforce the CI1 and CI2 coevolutionary interactions described in Section II-B, we obtain an asynchronous DECMO2++ variant (**E-aDECMO2++**) that, similarly to the generation-based baseline, features equidistant fitness sharing stages that take place after every batch of $|A|$ individuals has been evaluated. Therefore, during any optimization run, E-aDECMO2++ will contain as many fitness sharing stages as DECMO2++.

During an E-aDECMO2++ optimization run, the sub-population archive insertion ratios are determined after a random number of individuals (marked by $R_P, R_Q,$ and R_A) have been evaluated for each corresponding sub-population. By enforcing CI2, we have that $R_P + R_Q + R_A = |A|$ meaning

that the three random numbers are reset to 0 immediately after a fitness sharing stage (i.e., after a new batch of $|A|$ individuals has been evaluated).

It should be noted that during an E-aDECMO2++ run, in some cases, $R_P, R_Q,$ and R_A can be much smaller than their generation-based correspondent (i.e., $L_P, L_Q,$ and L_A defined in Section II-B). This in turn impacts the accuracy of the associated archive insertion ratio. In an attempt to alleviate this effect, we allow the $|B|$ bonus individuals to improve[1] the archive insertion ratios of sub-populations P and Q. This is a change from the generation-based DECMO2++ model where bonus individuals cannot contribute to the insertion ratios of P and Q since the ratios are only computed at the end of (odd-numbered generations) and the bonus individuals are assigned to P and Q only during even-numbered generations.

D. Sub-population-centered fitness sharing

When choosing to simultaneously enforce CI1 and (a softer version of) CI3, the obtained asynchronous DECMO2++ variant (**S-aDECMO2++**) features fitness sharing stages that are sub-population-centered – i.e., only take place after a minimum predetermined number of individuals (i.e., $L_P, L_Q,$ and L_A) has been evaluated for each sub-population. In this case, the DECMO2++ constraint from (3) still holds.

In S-aDECMO2++, the sub-population archive insertion ratios are also determined after a random number of individuals (i.e., $R_P, R_Q,$ and R_A) have been evaluated for each sub-population. The difference is that, in light of the softer version of the CI3 coevolutionary interaction, (4) must hold and thus the insertion ratios associated with $R_P, R_Q,$ and R_A should be as accurate as those of DECMO2++.

$$\begin{cases} R_P & \geq L_P \\ R_Q & \geq L_Q \\ R_A & \geq L_A \end{cases} \tag{4}$$

The downside of S-aDECMO2++ is that from (3) and (4) it immediately follows that $R_P + R_Q + R_A \geq |A|$ and this means that given a fixed number of fitness evaluations that are to be computed during the optimization run, S-aDECMO2++ will benefit from fewer fitness sharing stages than DECMO2++.

To sum everything up, by not enforcing either CI2 or CI3 (even in a softer version), the search behavior of both asynchronous variants of DECMO2++ will be impacted when compared to the baseline coevolutionary strategy as:

- E-aDECMO2++ will have less accurate insertion ratios;
- S-aDECMO2++ will have fewer fitness sharing stages.

The main aim of the present work is to highlight which of these downsides is more serious , especially when considering different levels of variance in the time-wise distribution of the fitness evaluation function.

[1]When computing the ratios, the bonus individuals are counted in the numerator if successfully inserted into A, but are not counted in the denominator (i.e., R_P or R_Q).

III. Performance evaluation

In order to determine the convergence behavior of E-aDECMO2++ and S-aDECMO2++, we used a pseudoparallel computational environment similar to the ones described in [5] and [21] as this enables an exact control of the numerical experiment, especially concerning the time-wise distribution of the fitness function. The pseudoparallel computational environment has one master node and λ =200 slave nodes. While all the tasks required to generate one new individual on the master node (are simulated to) take $t_s = 1$ unit of time, the duration of fitness evaluations is sampled from a normal distribution with the mean $t_p = 1000$ and the standard deviation $c_v \times t_p$. By changing c_v – i.e., the *coefficient of variation* – such that $c_v \in \{0.0, 0.05, 0.1, 0.2\}$, the time-variance of the fitness evaluation function can be easily controlled while maintaining the parallelization ratio $r = \lceil \frac{t_p}{t_s} \rceil$ fixed.

We performed optimization runs on an extensive benchmark suite that aggregates 31 standard MOOPs:

- all five real-valued problems from the ZDT test set [22];
- Kursawe's function [23]: 10 variables and 2 objectives;
- all seven problems from the DTLZ test set [24];
- all nine problems proposed in the LZ09 problem set [25];
- all nine problems from the WFG test set [26].

We made 100 independent repeats for each MOEA-MOOP combination. We only report over averaged results and, in some cases, we apply statistical significance testing when analyzing the observed differences.

A. Evaluation criteria

In this work, the quality indicator used for comparing optimization results is the (normalized) hypervolume metric [27] (notation Ind_H). The theoretical proof of a monotonic convergence behavior [28] for Ind_H makes this unary indicator of Pareto convergence an attractive basis for more advanced application-specialized comparative indicators.

For instance, in both [5] and [21], when wishing to compare an asynchronous and a synchronous (i.e., generation based) version of a MOEA from a qualitative point of view, Ind_H-based indicators are defined for measuring the differences in nfe-count or evaluation time requirements needed to reach a certain level of the true hypervolume (i.e., the one associated with the PS). In the present experiments we adopt the $\Delta_{qual}(p)$ indicator from [5] that measures the relative percentage of individuals that must be computed by an asynchronous MOEA (when compared to its generation-based baseline) in order to reach a PN with a hypervolume that is $p\%$ of the true hypervolume for a given problem. Concretely, $\Delta_{qual}(p)$ is usually positive (as the asynchronous versions need to compute more individuals) and is calculated as follows:

$$\Delta_{qual}(p) = \left(\frac{nfe_{async}(p)}{nfe_{gen}(p)} - 1 \right) \times 100 \tag{5}$$

where $nfe_{async}(p)$ represents the number of fitness evaluations required by the asynchronous version of the MOEA to reach a PN of at least $p\%$ Ind_H-measured quality and

$nfe_{gen}(p)$ represents the corresponding number of fitness evaluation required by the generation-based variant.

B. Hypervolume-ranked performance curves

Another application-specialized comparison indicator comes in the form of *hypervolume-ranked performance curves* (HRPCs). They were proposed in [10] to facilitate the run-time comparison of the convergence behavior of (several) MOEAs across test sets containing many MOOPs.

The idea is to rank the algorithms tasked to solve a MOOP based on Ind_H-measured quality at various (pre-defined) stages during the run – e.g., after evaluating a new batch of 1000 individuals. Concretely, at a given comparison stage, when using the *basic ranking schema*, the worst performer among n_s algorithms will receive the rank n_s and the best performing algorithm will receive the rank 1. Keeping in line with [10], a bonus rank of 0 is awarded to a MOEA that is estimated to have fully converged on a problem (i.e., $Ind_H > 0.99$) while a penalty rank of n_s+1 is awarded while the MOEA has not discovered a relevant PS approximation for the given MOOP (i.e., $Ind_H < 0.01$). By simply averaging (MOEA-wise and stage-wise) the ranks achieved on individual MOOPs, one can easily illustrate the comparative convergence behavior across an entire benchmark problem set.

At each comparison stage, HRPCs rely on making two-by-two MOEA comparisons in increasing order of performance and, in order to highlight certain convergence bahaviors, apart from the basic ranking, we also apply:

- a *pessimistic ranking schema* that only awards different ranks when the stage-wise differences between the average performance of two MOEAs is larger than a predefined threshold[2] marked by th;
- a *statistical ranking schema* that only awards different ranks when the stage-wise differences between the average performance of two MOEAs is deemed statistically significant by a one-sided Mann-Whitney-Wilcoxon test [29] with a considered significance level of 0.025.

C. Tested algorithms and parameterizations

We analyzed the performance of E-aDECMO2++ and S-aDECMO2++ when comparing both to their DECMO2++ baseline and to other well-known generation-based MOEAs:

- SPEA2 and GDE3 – the original MOEAs that proposed the search paradigms emulated by the P and Q subpopulations of DECMO2++;
- NSGA-II [14] – the best known and most widely applied MOEA (when considering the number of citations);
- MOEA/D-DE with Dynamic Resource Allocation [30] – a state-of-the-art solver that delivers highly competitive solutions for many MOOPs.

Across all optimization runs, the tested MOEAs used their (fixed) literature-recommended parameterization. The population / archive size was set to 200 (except for MOEA/D-DE where a special archive size that depends on objective-count is the standard parameterization) and each MOEA was awarded a computational budget of nfe=50,000 per optimization, irrespective of MOOP.

[2]For example, a value of $th = 0.05$ would require a minimal 5% difference between average Ind_H values to award different ranks.

IV. Results and interpretation

In the left-hand plot from Figure 2 we present the average Ind_H-measured performance over the 31 benchmark MOOPs of the E-aDECMO2++ asynchronous variant[3] and of the DECMO2++ generation-based baseline. With regard to the convergence behavior of the two coevolutionary approaches, it is important to observe that the generation-based algorithm performs better in the early phases of the optimization runs (i.e., $nfe < 15.000$). When considering the latter part of the optimization runs (i.e., $nfe > 15.000$) the performance of E-aDECMO2++ and DECMO2++ appears to be largely similar.

In the right-hand plot from Figure 2 we use the $\Delta_{qual}(p)$ metric defined in (5) to better characterize the difference in performance between E-aDECMO2++ and DECMO2++ (especially when considering variance). Thus, the solid black line ($c_v = 0$) directly corresponds to the left-hand plot and highlights that (on average, across the 31 benchmark MOOPs) in order to obtain a PN with a hypervolume that is between $p = 30\%$ and $p = 70\%$ of the true hypervolume, E-aDECMO2++ must evaluate $\approx 30 - 40\%$ more individuals than DECMO2++. When aiming for a PN that has $p = 80\%$ of the true hypervolume, E-aDECMO2++ must evaluate only $\approx 20\%$ more individuals than its baseline. In order to generally reach the best possible average performance of DECMO2++ (i.e., $\geq 90\%$ of the true hypervolume), E-aDECMO2++ needs up to 10% less fitness evaluations than DECMO2++.

When considering a fairly limited amount of variance (i.e., $c_v = 0.02$) in the time-wise distribution of the fitness evaluation function, all the previous observations regarding E-aDECMO2++ still hold, as the impact on the associated $\Delta_{qual}(p)$ is barely noticeable. However, whenever there are high levels of variance (i.e., $c_v = 0.10$ and $c_v = 0.20$), the convergence behavior of E-aDECMO2++ is visibly improved as, on average, E-aDECMO2++ must evaluate:

- only $\approx 20 - 25\%$ more individuals than DECMO2++ in order to reach PNs that have between 30% and $p = 70\%$ of the true hypervolume;
- only $\approx 10\%$ more individuals to reach PNs that have 80% of the true hypervolume;
- still 10% less individuals to match DECMO2++ peak performance.

It is very important to note that when not allowing bonus individuals to positively contribute to (i.e., improve) the archive insertion ratios of E-aDECMO2++, the positive impact of high variance ($c_v = 0.10$ and $c_v = 0.20$) on the convergence performance of E-aDECMO2++ is completely canceled and the $c_v = 0$ convergence behavior becomes standard.

[3]When considering a constant time-wise distribution of the fitness evaluation function.

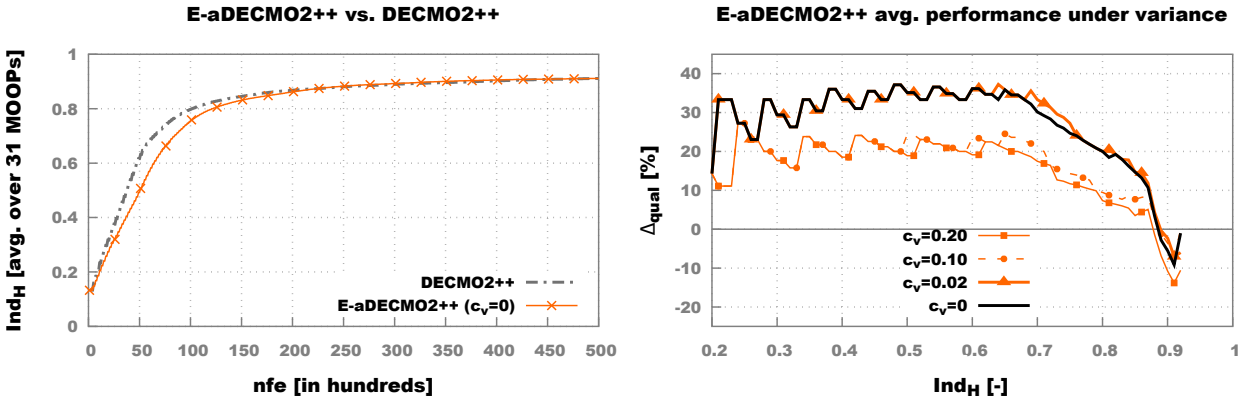

Fig. 2. Comparative E-aDECMO2++ run-time Ind_H and $\Delta_{qual}(p)$-measured performance averaged over 31 artificial benchmark MOOPs.

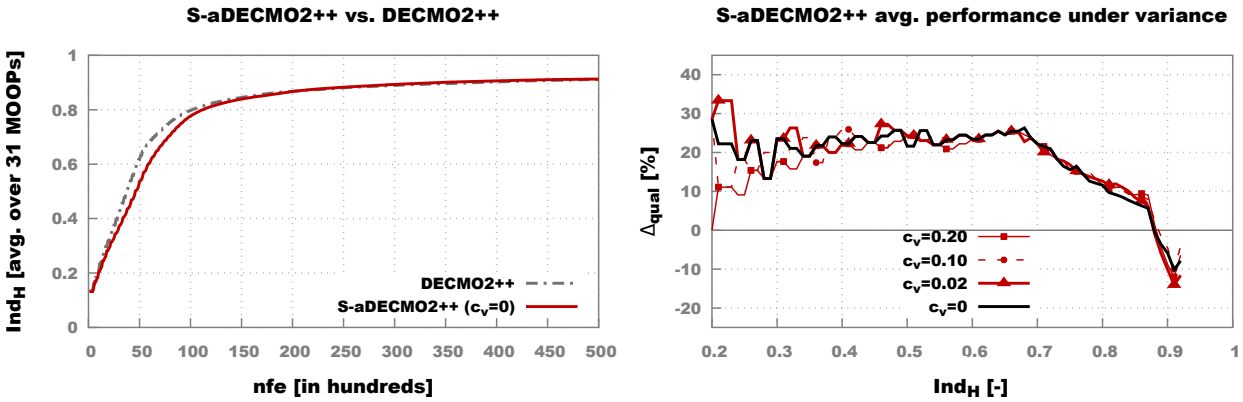

Fig. 3. Comparative S-aDECMO2++ run-time Ind_H and $\Delta_{qual}(p)$-measured performance averaged over 31 artificial benchmark MOOPs.

In the two plots from Figure 3 we present the average Ind_H and $\Delta_{qual}(p)$-measured performance of the S-aDECMO2++ asynchronous parallelization variant. Two observations are extremely important:

1) The average convergence behavior of S-aDECMO2++ across the 31 benchmark MOOPs is largely invariant with regard to the time-wise distribution of the fitness function (especially when aiming for PNs with hyprvolumes that are at least 30% of the true hypervolume).
2) The average convergence behavior of S-aDECMO2++ matches the one exhibited by E-aDECMO2++ under high variance (and, implicitly, is superior to that of E-aDECMO2++ under low variance in the time-wise distribution of the fitness evaluation function).

In light of all the aforementioned results and interpretations, S-aDECMO2++ is on average the better (i.e., more robust) asynchronous parallelization variant of DECMO2++. Furthermore, when also considering the reason behind the improved performance of E-aDECMO2++ when having large variance in the time-wise fitness distribution, it becomes obvious that, inside the baseline coevolutionary solver, maintaining even a softer version of the CI3 coevolutionary interaction (i.e., the accuracy of the insertion ratios) is more important than maintaining CI2 (i.e., the number of fitness sharing stages).

While all the performed experiments indicate that both asynchronous parallelization variants of DECMO2++ have higher run-time nfe requirements than their generation-based coevolutionary baseline, it is important to view this aspect in a proper context. Therefore, in Figure 4 we present the average Ind_H-measured convergence behavior of E-aDECMO2++ and S-aDECMO2++ when comparing with four other well-known generation-based MOEAs. The plots indicate that even when using an asynchronous parallelization paradigm, the coevolutionary approaches remain quite competitive (especially in the early phases of the optimization runs). Furthermore, the magnitude and statistical significance of the observed differences in convergence behavior are confirmed by the four complementary HRPCs from Figure 5.

The fact that S-aDECMO2++ is generally competitive with generation-based MOEAs when evaluating based on required nfe (i.e., qualitative aspect) is extremely important when also considering that in real-life optimization scenarios the asynchronous MOEA benefits from a structural computational speed-up that is likely to be improved by the presence of variance in the time-wise distribution of the fitness evaluation

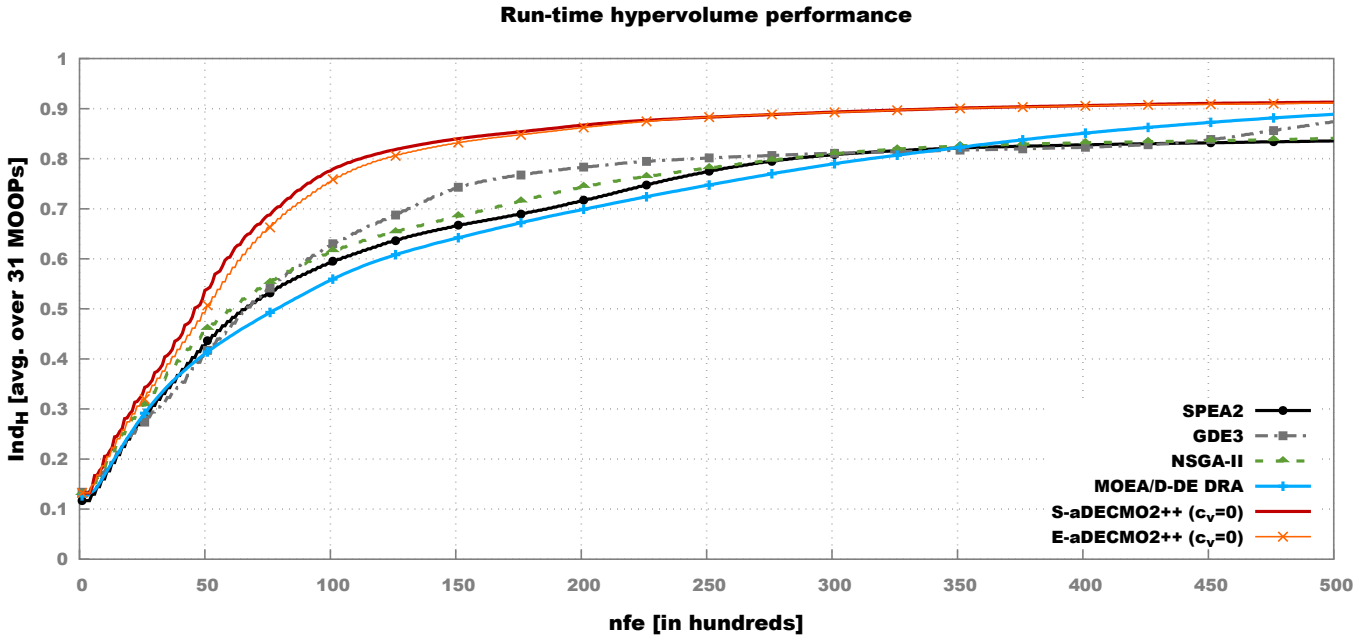

Fig. 4. Run-time Ind_H-measured performance averaged over 31 artificial benchmark MOOPs.

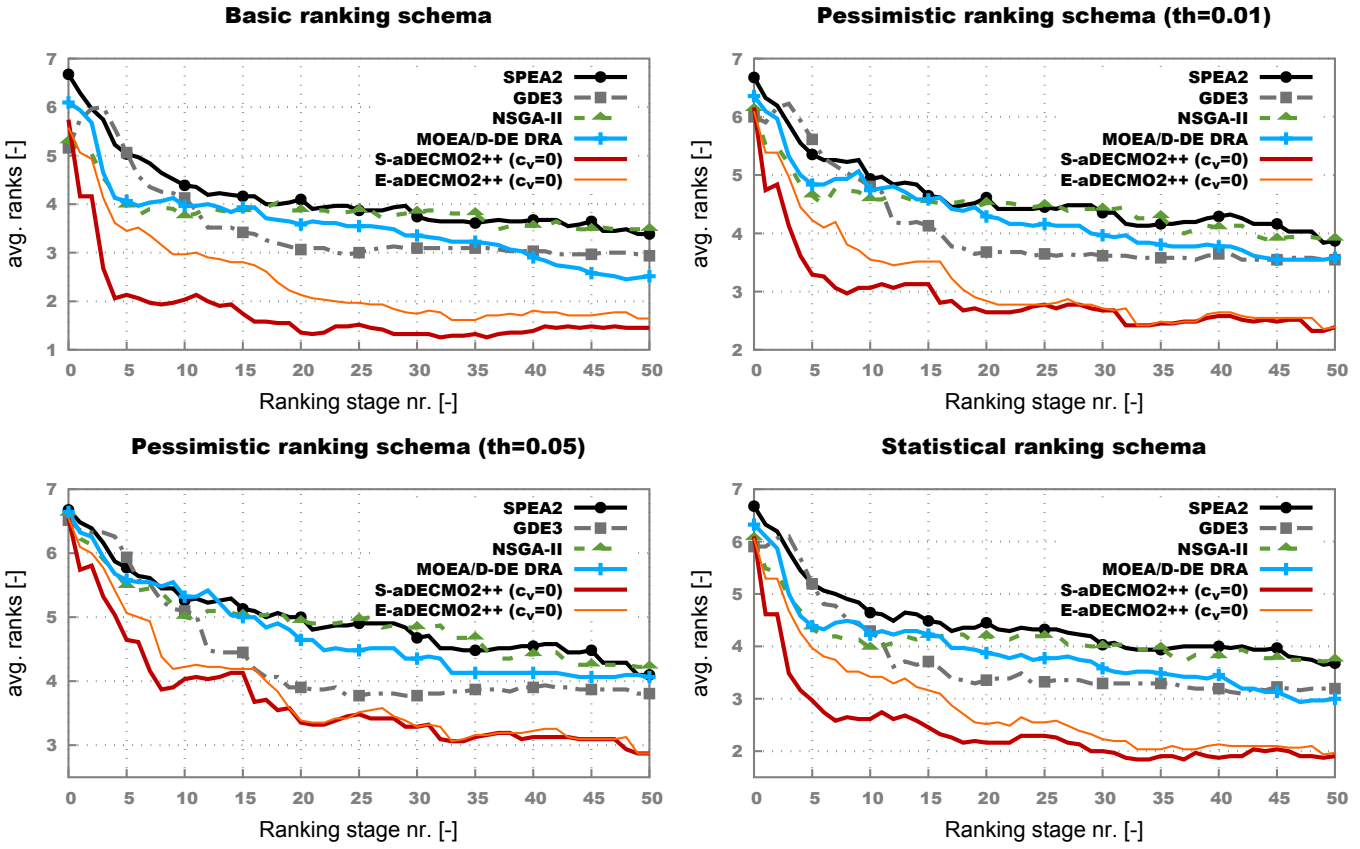

Fig. 5. HRPCs obtained when comparing E-aDECMO2++ and S-aDECMO2++ with four generation-based MOEAs across the 31 benchmark MOOPs. The nfe between two consecutive ranking stages is 1000.

function (i.e., quantitative aspect).

V. Conclusions and future work

In this paper we analyzed two asynchronous parallelization variants for the DECMO2++ multi-objective coevolutionary solver. Our analysis revealed that each parallelization option infringed on one of three key (synchronized) coevolutionary interactions enforced by the generation-based (baseline) model. When also considering the impact of having variance in the time-wise distribution of the fitness function, choosing to operate with a more accurate estimation of the comparative performance of the three coevolved sub-populations (i.e., the S-aDECMO2++ variant) proved more successful than the strict enforcement of equidistant fitness sharing stages (i.e., the E-aDECMO2++ variant) during the optimization run.

Future work will revolve on improving both the testing methodology and the range of considered asynchronous parallelization options suitable for coevolutionary solvers. Thus, we believe that the simulations performed on the pseudoparallel computational environment should also consider potential slave node failure, time-wise fitness distributions that are not normally distributed and, more importantly, scenarios where the time-wise and quality-wise distributions of the fitness evaluation functions are correlated (e.g., solutions with better fitness take longer to evaluate). From an algorithmic perspective, coupling DECMO2++ with more advanced parallelization options, like the adaptive semi-asynchronous strategy introduced in [21], is expected to deliver multi-objective solver variants that are even more competitive than S-aDECMO2++.

Acknowledgment

This work has been supported by the "LCM K2 Center for Symbiotic Mechatronics" within the framework of the Austrian COMET-K2 program.

References

[1] C. A. Coello Coello and G. B. Lamont, *Applications of multi-objective evolutionary algorithms*. World Scientific, 2004.

[2] A. Lewis, S. Mostaghim, and I. Scriven, "Asynchronous multi-objective optimisation in unreliable distributed environments," in *Biologically-Inspired Optimisation Methods*. Springer, 2009, pp. 51–78.

[3] M. Depolli, R. Trobec, and B. Filipič, "Asynchronous master-slave parallelization of differential evolution for multi-objective optimization," *Evolutionary Computation*, vol. 21, no. 2, pp. 261–291, 2013.

[4] S. Wessing, G. Rudolph, and D. A. Menges, "Comparing asynchronous and synchronous parallelization of the SMS-EMOA," in *International Conference on Parallel Problem Solving from Nature*, ser. LNCS, vol. 9921. Springer, 2016, pp. 558–567.

[5] A.-C. Zăvoianu, E. Lughofer, W. Koppelstätter, G. Weidenholzer, W. Amrhein, and E. P. Klement, "Performance comparison of generational and steady-state asynchronous multi-objective evolutionary algorithms for computationally-intensive problems," *Knowledge-Based Systems*, vol. 87, pp. 47 – 60, 2015.

[6] T. Harada and K. Takadama, "Performance comparison of parallel asynchronous multi-objective evolutionary algorithm with different asynchrony," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1215–1222.

[7] C. C. Coello and M. R. Sierra, "A coevolutionary multi-objective evolutionary algorithm," in *Evolutionary Computation (CEC), 2003 IEEE Congress on*, vol. 1. IEEE, 2003, pp. 482–489.

[8] J. A. Vrugt and B. A. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," *Proceedings of the National Academy of Sciences*, vol. 104, no. 3, pp. 708–711, 2007.

[9] A.-C. Zăvoianu, S. Saminger-Platz, E. Lughofer, and W. Amrhein, "Two enhancements for improving the convergence speed of a robust multi-objective coevolutionary algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 793–800.

[10] A.-C. Zăvoianu, E. Lughofer, G. Bramerdorfer, W. Amrhein, and E. P. Klement, "DECMO2: a robust hybrid and adaptive multi-objective evolutionary algorithm," *Soft Computing*, vol. 19, no. 12, pp. 3551–3569, 2014.

[11] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*. International Center for Numerical Methods in Engineering, 2002, pp. 95–100.

[12] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 1995.

[13] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[15] R. Storn and K. V. Price, "Differential evolution - a simple and effcient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, December 1997.

[16] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Evolutionary Computation (CEC), 2005 IEEE Congress on*. IEEE Press, 2005, pp. 443–450.

[17] T. Robič and B. Filipič, "DEMO: Differential evolution for multi-objective optimization," in *International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, Springer. Springer Berlin / Heidelberg, 2005, pp. 520–533.

[18] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - A comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.

[19] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, December 2007.

[20] A.-C. Zăvoianu, E. Lughofer, W. Amrhein, and E. P. Klement, "Efficient multi-objective optimization using 2-population cooperative coevolution," in *Computer Aided Systems Theory - EUROCAST 2013*, ser. LNCS, no. 8111. Springer Berlin Heidelberg, 2013, pp. 251–258.

[21] T. Harada, "Adaptive asynchrony in semi-asynchronous evolutionary algorithm based on performance prediction using search history," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 1031–1038.

[22] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[23] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Workshop on Parallel Problem Solving from Nature (PPSN I)*, ser. LNCS, vol. 496. Springer, 1991, pp. 193–197.

[24] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Evolutionary Computation (CEC), 2002 IEEE Congress on*. IEEE Press, 2002, pp. 825–830.

[25] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[26] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *Evolutionary Multi-Criterion Optimization (EMO 2005)*, ser. LNCS, vol. 3410, 2005, pp. 280–295.

[27] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Ph.D. dissertation, Swiss Federal Institute of Technology, 1999.

[28] M. Fleischer, "The measure of Pareto optima: Applications to multi-objective metaheuristics," in *Evolutionary Multi-Criterion Optimization (EMO 2003)*, ser. LNCS, vol. 2632. Springer, 2003, pp. 519–533.

[29] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, vol. 18, no. 1, pp. 50–60, 1947.

[30] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," School of CS & EE, University of Essex, Tech. Rep., February 2009.