HAL
open science

Asynchronous Master/Slave MOEAs and Heterogeneous Evaluation Costs

Mouadh Yagoubi, Marc Schoenauer

To cite this version:

Mouadh Yagoubi, Marc Schoenauer. Asynchronous Master/Slave MOEAs and Heterogeneous Evaluation Costs. Genetic and Evolutionary Computation Conference (GECCO 2012), Jul 2012, Philadelphia, United States. pp.1007-1014. hal-00689965

HAL Id: hal-00689965

https://hal.science/hal-00689965v1

Submitted on 20 Apr 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Asynchronous Master/Slave MOEAs and Heterogeneous Evaluation Costs

Mouadh Yagoubi[1,2]
1. PSA PEUGEOT CITROËN Centre technique de Vélizy-Villacoublay, France.
Mouadh.Yagoubi@inria.fr

Marc Schoenauer[2]
2. TAO INRIA Saclay Université Paris-Sud, Orsay, France.
Marc.Schoenauer@inria.fr

ABSTRACT

Parallel master-slave evolutionary algorithms easily lead to linear speed-ups in the case of a small number of nodes ... and homogeneous computational costs of the evaluations. However, modern computer now routinely have several hundreds of nodes – and in many real-world applications in which fitness computation involves heavy numerical simulations, the computational costs of these simulations can greatly vary from one individual to the next. A simple answer to the latter problem is to use asynchronous steady-state reproduction schemes. But the resulting algorithms then differ from the original sequential version, with two consequences: First, the linear speed-up does not hold any more; Second, the convergence might be hindered by the heterogeneity of the evaluation costs. The multi-objective optimization of a diesel engine is first presented, a real-world case study where evaluations require several hours of CPU, and are very heterogeneous in terms of CPU cost. Both the speed-up of asynchronous parallel master/slave algorithms in case of large number of nodes, and their convergence toward the Pareto Front in case of heterogeneous computation times, are then experimentally analyzed on artificial test functions. An alternative selection scheme involving the computational cost of the fitness evaluation is then proposed, that counteracts the effects of heterogeneity on convergence toward the Pareto Front.

Keywords

Parallellization, Heterogeneous Cost, Steady-State Algorithms

1. INTRODUCTION

Many real-world optimization problems involve heavy numerical simulations for computing a single value of the objective function and/or the constraints. Even during Moore years, the exponential increase of computer power could not match the ever-increasing demand for more and more complex simulations. But the hardware today seems to have reached the end of Moore law, and increases in computational power can only be brought now by the use of a large number of computing units, be it on multi-core CPUs, on GP-GPUs, on clusters of CPUs, on world-wide Computing Grids or on the Cloud. And taking advantage of such number of separate computing entities then becomes a software challenge.

At the same time, many (if not all) real-world problems are multi-objective by nature, involving the optimization of several contradictory objectives (e.g., typically, cost and quality). These objectives are generally not regular (e.g., not differentiable), noisy, and multi-modal, prohibiting the use of traditional optimization methods. Hence, heuristic and stochastic methods like Multi-Objective Evolutionary Algorithms (MOEAs) [8] are good candidates for tackling such problems, thanks to their robustness and their flexibility. Unfortunately, the price of this efficiency under such weak hypotheses is a high computational cost in terms of number of function evaluations required to reach a satisfactory solution.

Fortunately, whereas most traditional algorithms need to be almost completely rewritten to efficiently use widely distributed computational resources, Evolutionary Algorithms can be parallelized in several efficient ways [5, 1]. The master-slave model, in which a master node distributes fitness evaluations to slave nodes, and performs all evolutionary operations (selection, variation operators), is by far the simplest one to set up, and as such has been widely used in many application domains. Furthermore, because all evolutionary routines are performed in the master node, similarly to the sequential case, such algorithm can be extended to the multi-objective case without any additional algorithmic effort. Other approaches include the Island models, where each node runs a standard EA and the different nodes exchange "migrants" from time to time, and the totally distributed models, where each node contains one (or very few) individuals, and the selection and crossover operations are performed amongst neighboring nodes. From a parallel perspective, it is important to note that both the Island model and the totally distributed model are based on some underlying topology of the graph of nodes: matching this topology with that of the hardware allows the programmer to optimize the communication costs between nodes by restricting them to physically neighboring nodes. On the other hand, the master-slave model is based on a star-shaped topology, and all communications go through the master node: this has been considered a major limitation of such approaches on several specific highly parallel architectures. However,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'12, July 7-11, 2012, Philadelphia, Pennsylvania, USA.
Copyright 2012 ACM 978-1-4503-1177-9/12/07 ...$10.00.

when the cost of one fitness computation is of several orders of magnitude larger than the cost of the evolutionary operators (e.g. dozens of minutes vs milliseconds), such limitation doesn't hold any more.

Furthermore, the Island and the totally distributed models need to be specifically adapted to the multi-objective context. For instance, a Parallel approach of NSGA-II has been proposed in [11], that is based on the "guided domination approach": each of the participating processors is assigned the task of finding only a particular portion of the Pareto set. A geometrical approach (the cone separation) is proposed in [4], that subdivides the search space in several regions. In order to have each processor focus on a specific region, the borders of each region are treated as constraints. In [2], a complex mechanism is designed that handles multiple populations across heterogeneous processors, based on mobile computing agents. All these approaches, however, require large modifications of the algorithm, and none has been proven to significantly outperform the others on a wide range of problems. This is probably why many practical works using parallel MOEAs have focused on the master-slave model for its simplicity – and so does this paper.

The simplest implementation of master-slave evolutionary algorithm (be it single- or multi-objective) is the exact reproduction of the *generational* model, in which all individuals are sent out to slaves for evaluation, and evolutionary operators take place only when the whole population has been evaluated. In this model, the speed-up can be proved to be linear w.r.t. the number of processors, up to some limit when the communication time takes over the computing time of one evaluation [5]. Because it is also highly sensitive to failures (a major concern when the number of precessing units becomes very large), the generational model is only used as a baseline for comparisons, and in practice, the *asynchronous steady-state* model is used instead. The steady-state model was proposed in the sequential context [17] in order to eventually speed-up evolution by inserting back in the population each newborn offspring as soon as it has been evaluated. In the asynchronous parallel setting, each newborn offspring is sent to a slave for evaluation, and evaluated individuals are inserted back in the population on a first-come first-served basis. In case of failure of a slave (e.g., one individual never returns), the algorithm is undisturbed. Furthermore, and as importantly in real-world applications where different evaluations can take very different times, all available CPU time is used: as soon as a slave has completed an evaluation, the master sends it a new individual to evaluate, and thus no computing unit ever stays idle, at least in case the evaluation time is much larger than communication time.

The debate between steady-state and generational schemes has been going on for long. In a general real-world framework, [6] argues that steady-state performs very often better than generational, and even more so in a multi-objective optimization context. In [12], another comparison between steady-state and generational NSGA-II is proposed on a real case study. However, to the best of our knowledge, no systematic comparison has been made between the asynchronous steady-state and the generational algorithms in terms of speed-up: this will be a first contribution of this paper (Section 5), using standard benchmark functions.

Though efficient in terms of usage of the computational resources in case of heterogeneous evaluation times, asynchronous steady-state MOEAs can sometimes suffer from such heterogeneity in terms of quality of the solutions: indeed, when some part of the search space systematically takes longer to evaluate than all others, asynchronous steady-state algorithms might poorly sample that part. The second contribution of this paper is twofold: Firstly, to empirically demonstrate on both a real-world case-study and on artificially modified benchmark functions that such poor sampling indeed happens; Secondly, to propose a modified selection process for asynchronous steady-state multi-objective algorithms that will somehow correct the problem – while not slowing down too much the convergence when applied in homogeneous context. This selection scheme will be applied to two well-known MOEAs, namely NSGA-II [10] and MO-CMA-ES [14].

Next Section will present the real-world application (optimization of a diesel Engine) that motivated this work, first introducing the optimization problem itself, and analyzing the results obtained with NSGA-II (Section 3.1) in terms of both the computing time per fitness evaluation and the quality of the results. Section 2.2 will set the research questions that will be addressed in this work, and give the agenda of the rest of the paper.

2. OPTIMIZING DIESEL COMBUSTION

Environmental regulations are becoming very strict in Europe, setting hard constraints on the development of efficient automotive engines with low fuel consumption and low pollutant emissions. More specifically, when designing a diesel engine, three objectives are to be minimized, NOx, Soot, and Fuel consumption, that are naturally in conflict: this problem pertains to multi-objective optimization, and NSGA-II [10] was chosen to tackle it because of its well-known robust efficiency (Section 3.1).

All objectives can be computed using a multidisciplinary simulation of the combustion. However, the complexity of the phenomena that occur in the combustion chamber requires a very fine discretization in the numerical model: simulating the complete 3D model requires around 3 days on a recent single-core computer. Hence a simplified model was used here, that does not take into account the exact geometry of the combustion chamber, and thus only involves the 10 decisions variables that control the combustion process itself, i.e., the injection parameters, and the air/fuel mixing parameters (more details can be found in [20]). The simulation time is reduced to ... a few hours. However, highly non-linear phenomena are involved in combustion models, and the actual simulation run-times vary a lot depending on the values of the variables, from 1.5 to 24 hours, as can be seen on Figure 1-right for 40 random instances of the parameters.

In order to cope with this complexity, it was decided to use a 40-CPUs cluster, and hence to use parallel versions of NSGA-II. Furthermore, in order to keep all experiments within a reasonable range of CPU costs, all single fitness evaluations were eventually stopped after 8 hours, based on a human decision: the simulation is an iterative procedure, and unpromising fitness computations with respect to convergence of this iterative procedure (based on the expert's estimation) were stopped, while more promising runs were continued up to a maximum of 12 hours.

2.1 Experimental Results

Two variants of parallel NSGA-II were experimented with:

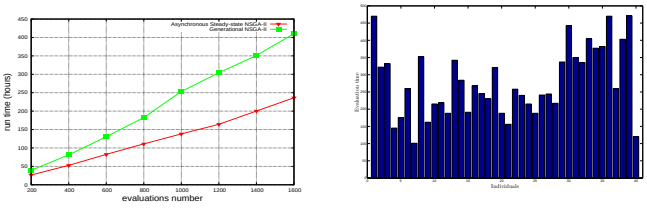

Figure 1: Left: Comparison of total run times of both generational and asynchronous steady-state variants. Right: Run-time of individual fitness computations for 40 random individuals.

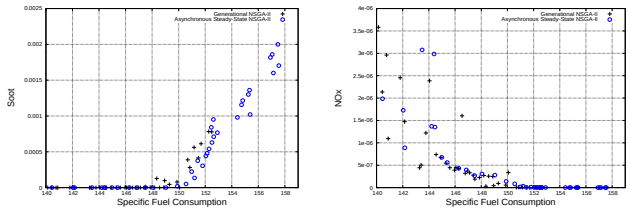

Figure 2: Two 2D projections of the Pareto Front. Soot (Left) and NOx (Right) vs Fuel Consumption.

the standard parallel generational algorithm, and the asynchronous steady-state version (Section 3.3), both with population size 40. Both algorithms were run for 1600 evaluations, regardless of their actual durations (i.e. 40 generations for the generational variant). The results of a single run are reported here, as an illustration of the effects of heterogeneity on multi-objective CPU costs . . . and results.

Figure 1-left shows the run time required along evolution, at different numbers of function evaluations, for both generational and asynchronous algorithms. The total optimization cost for the generational NSGA-II is above 17 days, while that of the asynchronous NSGA-II is less than 10 days. As expected, the usage of CPU is much more efficient with the asynchronous steady-state algorithm than for the generational one.

More interestingly, Figure 2 displays some 2D projections of the Pareto fronts in the space "Soot-Fuel consumption" vs "NOx-Fuel consumption". Both optimizations are comparable in terms of quality of convergence. However, some very efficient solutions in terms of NOx values (with of course poor Soot values) have been discovered by the asynchronous optimization, and not in the generational case. A closer look revealed that these solutions represent individuals with large evaluation time, close or over the 8 hours threshold, that were likely to be killed in the generational approach.

2.2 Discussion

These experiments, beside having provided interesting solutions in terms of Engine Optimization [20], also give very strong arguments in favor of using asynchronous steady-state algorithms in case of heterogeneous fitness computational costs (if at all necessary), in spite of the guaranteed linear parallel speed-up of the generational model in the homogeneous case: beside the obvious more efficient use of computational resources, and though not visible in the above results, the fault-tolerance property of the asynchronous approaches makes the case clear. However, the results obtained by both algorithms differ. It is the case here because the artificially-set 8-hours threshold, but it also clearly indicates that indeed some parts of the search space (and hence of the Pareto set) might exhibit very long computing times – and this can certainly have some consequences on the convergence of any MOEA.

Based on these considerations, the rest of this paper will try to address the following issues: i- what speedups can be expected from the asynchronous steady-state algorithms compared to their sequential version, even in the homogeneous case (Section 5); ii- how bad can be the consequences of highly heterogeneous evaluation times (Section 6.1); iii- can we circumvent this difficulty. This will lead to proposing the *Duration-Based Selection* in Section 6.2, that will be validated on artificial test functions.

3. PARALLEL MASTER-SLAVE MOEAS

This Section briefly describes the two MOEAs that will be concerned in this work, namely NSGA-II and MO-CMA-ES. Many different MOEAs have been proposed since the mid 80s, and the most popular today use the idea of Pareto dominance: a solution x Pareto-dominates a solution y if x is better than y on all objectives, and strictly better on at least one objective. The algorithm then proceeds almost like a single-objective EA, except for the selection procedures (both parental and survival), that are replaced by Pareto-based selections using two hierarchical criteria: the first one is based on Pareto dominance, and the secondary criterion enforces the dispersion of the solutions over the Pareto front (see [8, 7] for details).

Among the many MOEAs that have been proposed in the literature, only two have been considered in this work: NSGA-II (Non-dominated Sorting Genetic Algorithm), proposed in the early 90s [10], still considered today as one of the state-of-the-art MOEA for its robustness across a variety of application domains, and the more recent MO-CMA-ES (Multi-Objective Covariance Matrix Adaptation Evolution Strategy) [14], that transposes to the multi-objective context the qualities (and in particular the invariance properties) of the single-objective CMA-ES [13]. Both will be very briefly described in turn, in their original sequential versions. Their parallelization will then be discussed.

3.1 NSGA-II

The original (generational) NSGA-II algorithm uses Non-Dominated Sorting as a first selection criterion, and Crowding Distance as secondary diversity-preserving criterion (though some recent work suggests that the hypervolume might be a better choice as diversity-preserving criterion [3]).

Non-Dominated Sorting is based on the "Pareto rank" of individuals in a given population: the non-dominated individuals are given rank 1 and removed from the population. The non-dominated individuals of the remaining of the population are given rank 2, and the process continues until all individuals have a Pareto rank. Crowding Distance considers the objectives sequentially. The individuals are sorted according to objective i, and the partial crowding distance of the individual that has rank r with respect to objective i is the difference between the values of objective i of individuals with ranks $r-1$ and $r+1$. The Crowding Distance of an individual is then the sum over all objectives of its partial crowding distances. The comparison between two

individuals goes as follows: if their Pareto ranks are different, the smallest one is preferred, favoring progress toward the Pareto front; If they are equal, the one with largest Crowding Distance is preferred, favoring diversity.

Using this comparison procedure, both generational and steady-state [17] versions of NSGA-II can be easily described. Both use a population of size P, and deterministic tournament of user-defined size T as parental selection. The generational NSGA-II uses a standard $P + P$ scheme: P offspring are generated, and the best of the P parents + P offspring, according to the comparison described above become the parents of next generation. The steady-state version of NSGA-II uses a standard $P + 1$ steady-state scheme: 1 offspring is generated, and replaces in the population the worst individual (according to the comparison procedure at hand).

3.2 MO-CMA-ES

MO-CMA-ES, proposed by Igel et al [14], extends the single-objective CMA-ES [13] to the multi-objective context. Like NSGA-II, it uses the Non-Dominated Sorting as main comparison criterion between solutions. The hypervolume-indicator [22] is used as secondary comparison criterion.

All versions of the algorithm consider μ parents, and each individual is as (1+1)-CMA-ES algorithm. The generational version is a $(\mu+\mu)$ algorithm without parental selection and using only Gaussian mutation: each of the μ parents generates one offspring, and the μ best (according to the hierarchical sorting criterion described above) of the $2 * \mu$ parents plus offspring become the parents of the next generation. The mutation parameters of the selected offspring are updated following standard rank-one update for the covariance matrix, and a specific update rule close to the well-known 1-fifth rule for the step-size.

A steady-state version of MO-CMA-ES can be easily obtained by generating only one offspring, and inserting it in the population immediately after evaluation, i.e., as for NSGA-II, using a $\mu + 1$ steady-state scheme. Two possible variants were proposed in [15]: in the $(\mu + 1)$ algorithm, the parent is selected uniformly in the whole population, while in the $(\mu_{\prec} + 1)$ version, the parent is selected only amongst non-dominated individuals in the population. Further improvement were recently obtained by considering a tournament for parental selection rather than uniform selection [16]. But only the $(\mu_{\prec} + 1)$ will be considered here.

3.3 Asynchronous Steady-State MOEAs

The asynchronous parallel master-slave versions of the steady-state versions described above for both NSGA-II and MO-CMA-ES are straightforward to derive from the sequential versions: offspring are sent out to slaves for evaluation, and inserted back in the population on a first-come first-served basis [19]. The resulting algorithms are termed *AS-NSGA-II* and *AS-MO-CMA-ES* respectively.

Note that steady-state selection scheme has been applied in several MOEAs in the literature: the ϵMOEA [9] is based on the ϵ-dominance concept and uses steady-state selection scheme and archive update strategy. SMS-EMOA proposed by Beume et al. [3] is a steady-state MOEA that uses the hypervolume criterion as the secondary selection criteria – but these works address only the sequential context. The parallel versions of these algorithms have been studied in detail in [12], on a real-world application, and indeed the asynchronous versions were demonstrated to perform better. However, the main concern in this work was that of the total elapsed time for large hardware platform (e.g., peer-to-peer networks). And heterogeneity in fitness computations, when addressed, mostly came from the hardware, and did not depend on the individual being evaluated.

4. EXPERIMENTAL SETTINGS

All experiments presented in the rest of the paper use well known analytical test functions, more precisely, ZDT [21] and IHR [14] test suites, that will not be detailed any more here. The evaluations of these functions are very fast, and homogeneous in terms of computing time. However, in order to simulate asynchronicity, the algorithms maintain an additional queue that contains the evaluated offspring in the order they will be inserted back in the population at the end of the steady-state loop. This allows us to simulate any given kind of heterogeneity when adding newcomers in the queue, and selecting which one will then be inserted in the population (using the survival selection at hand).

For instance, offspring can be appened at the end of the queue, and either retrieved from the beginning of the queue (to simulate a strictly synchronous context), or uniformly (to simulate some heterogeneity uniformly distributed in the search space, or caused by the use of different types of hardware). The latter case applied to test function FOO will be refered to as *Rand-VC-FOO* (VC stands for Variable Cost).

Another possibility is to compute some "duration" based on the values of the objectives, and to insert the offspring in the queue based on such duration (ties are randomly broken), thus simulating the heterogeneous context in which some region of the search space are more costly to evaluate than others, as seen for instance in Section 2. The offspring that is sitting at the beginning of the queue is then inserted in the population. This case, applied to function FOO, will be refered to as *Region-VC-FOO*. However, in the experiements presented in Sections 6.1 and 6.2, a simple cost model is used: the durations can take only 2 values, the small one in most of the search space, and the large one in the region of interest.

Another important remark is that the size of the queue corresponds to the number of available processing units. Varying this parameter will allow us to simulate systems of different sizes (see next Section 5).

All results reported in the following are averages overs 30 independent runs. However, only averages or medians will be presented, as the standard deviations were very small indeed, and error bars would have made all plots unreadable. Unless otherwise stated, all runs were stopped after 50000 evaluation. The size of the population was set to 100 for both NSGA-II and MO-CMA-ES, and tournament size for NSGA-II was set to 2. The performance measurement considered in this experiments is the hypervolume indicator [22] which assesses both convergence and diversity.

5. ASYNCHRONOUS SPEED-UPS

Before addressing the issue of heterogeneous costs of fitness evaluation, we will first take an empirical look at the speed-ups that can be obtained by both asynchronous algorithms presented in Section 3.3 in some loosely homogeneous context, i.e., where heterogeneity is uniformly distributed (e.g. when it is due to hardware). These experi-

ments involve the artficial test functions *Rand-VC-ZDT* and *Rand-VC-IHR* defined in Section 4. Different sizes of the waiting queue will be used, and compared to the baseline, the single-processor asynchronous steady-state algorithm, i.e. the same algorithm with a queue-size 1.

Figures 3 and 4 show the evolution of the median hypervolume indicator for different values of queue-size, for AS-NSGA-II and AS-MOCMA-ES algorithms, with Rand-VC-ZDT3 test function: The left plots display this evolution in terms of total number of evaluations (regardless of parallelization), whereas the right plots show this evolution in terms of elapsed time. For the latter plot, the x-axis has been arbitrarily scaled by considering that the 50 000 evaluations performed for the 1-CPU algorithm (the black up-triangle curve on top) correspond to 1000 units (the 1-CPU line is only partially displayed to make the other lines readable).

Table 1 summarizes the results obtained on functions *Rand-VC-ZDT1*:3 and *Rand-VC-IHR1*:3 using different queue-sizes. For each column (hypervolume value given in lines labeled "Level"), the figure on the $nProc$ line is the ratio between the median elapsed times of the asynchronous algorithms using $nProc$ and 1 processing units respectively.

From these results, it appears that the speed-ups in terms of elapsed time are, for these test functions, clearly sub-linear. Furthermore, comparing the queue-sizes 50 and 100, a gain of a factor near to 2 is globally observed on the speed-up. However, this factor does not exceed 1.5 comparing high values of queue-size (500 and 1000).

But another observation can be made from Figures 3 and 4: the total number of evaluations (regardless of parallelization) that is needed to reach a given value of hypervolume decreases as the number of processors increases. In particular, if one has to pay for the total CPU time used (in case of commercial clusters for instance), then the best setting might still be the sequential one (at least if a large elapsed time is not a problem). This phenomenon can be explained as follows: Because of the steady-state selection scheme, one offspring is created and inserted into the waiting queue at each generation. Assuming that newborn individuals are better than their parents with a given probability, the quality of the offspring sitting in the waiting queue is on average proportional to their genetic age : young individuals are better than old ones. In such context, the larger the queue-size, the lower its average quality.

Furthermore, we also see a difference in the behaviors of MO-CMA-ES and NSGA-II, especially on ZDT functions: For AS-MO-CMA-ES, the "delay" in convergence discussed in the previous paragraph is recovered relatively faster than with AS-NSGA-II (Figure 3-left and 4-left). This phenomenon is also illustrated in table 1 where the speed-up to reach small levels of hypervolume indicator is higher for AS-MO-CMA-ES than for AS-NSGA-II. This can be explained by the higher selection pressure of NSGA-II algorithms, that select both parents for crossover by tournament selection, whereas MO-CMA-ES algorithms randomly choose the parent before applying mutation: the resulting offspring in the queue are on average of better quality for NSGA-II than for CMA-ES, and thus the latter algorithms suffer less if the inserted individuals are of lower quality.

Two other series of experiments were performed: In the first one, the population size was set to 500. In the second one, a sequential algorithm was used instead of the asynchronous steady-state version: offspring are ranked in the waiting queue in the same order than they were created. However, the analysis of these experiments gives similar results than the ones above. Those results are not given here due to space limitations, but can be found in [18] .

Table 1: Speed-ups for the convergence to given hyper-volume levels (ratio of elapsed times of asynchronous algorithms with $nProc$ and 1 processor).

Algo.	AS-NSGAII			AS-MOCMA-ES		
Level	0.1	0.01	0.001	0.1	0.01	0.001
Rand-VC-ZDT1						
50	41.91	43.08	45	45.65	44.27	45.98
100	74.50	77.94	81.61	82.89	84.15	90.35
200	120	132.91	137.6	168	177.08	183.92
500	276.92	244.23	263.86	318.18	345.52	360.13
1000	343.37	378.57	410.6	414.47	461.95	507.39
Rand-VC-ZDT2						
50	48.58	42.75	45.63	52.4	44.03	44.3
100	82.69	75.84	81.13	98.86	84.95	87.2
200	141.56	129.16	143.33	187.09	162.7	161.48
500	259.81	240.77	263.62	365.54	335.66	340.62
1000	391.79	366.32	411.7	514.79	480	500
Rand-VC-ZDT3						
50	42.15	43.07	44.3	44.56	48.13	47.93
100	74.13	76.71	78.8	89.13	94.51	93.93
200	126.47	129.85	136.8	169.65	184.52	190.76
500	228.72	247.24	262.26	339.77	363.84	356.32
1000	353.42	379.01	404.73	518.98	567.76	613.86
Level	10	5	1	10	5	1
Rand-VC-IHR1						
50	40	36.36	-	35.71	36.76	24.06
100	66.66	61.53	-	78.12	85.22	52.95
200	100	94.11	-	172.41	178.57	83.73
500	250	190.47	-	409.83	407.6	206
1000	307.69	258.06	-	735.29	728.15	254.32
Rand-VC-IHR2						
50	54.97	-	-			
100	242.01	61.53	-	-	-	-
200	361.94	229	-	-	-	-
500	1294	1178	-	-	-	-
1000	3029	2692	-	-	-	-
Level	10	5	min	10	5	min
Rand-VC-IHR3						
50	22.72	25.86	25	27.17	29.27	29.62
100	41.66	42.85	35.89	60.24	63.57	60.37
200	62.5	69.76	71.79	104.16	124.47	118.51
500	131.57	144.23	155.55	268.81	313.38	283.68
1000	185.18	283.01	224	520.83	589.4	544.21

6. HETEROGENEOUS CONTEXT

6.1 Deleterious Asynchronicity

The second part of the experimental work presented in this paper deals with the heterogeneity of evaluation costs. First, this Section presents experimental evidence that indeed asynchronous algorithms can be hindered by heterogeneity and completely miss a complete region of the Pareto Front (and hence of the Pareto Set) when this region requires higher computational cost than other parts of the search space.

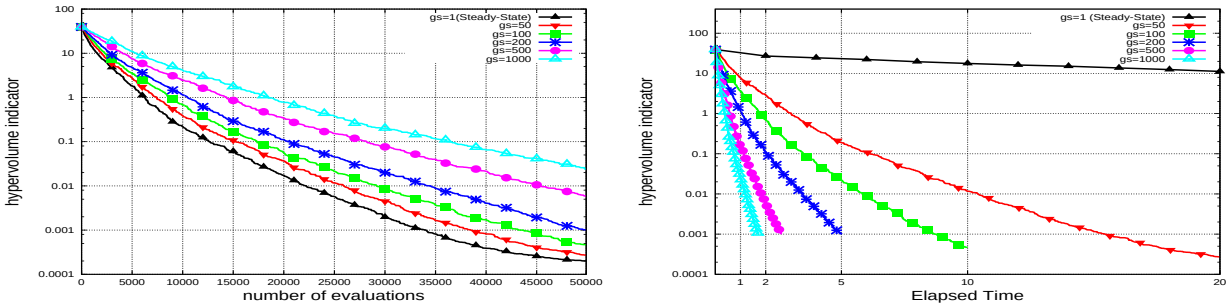

Figure 3: Evolution of average hypervolume indicator for AS-NSGA-II on Rand-VC-ZDT3 for different queue-sizes. Left: in terms of # evaluations. Right: in terms of elapsed time.

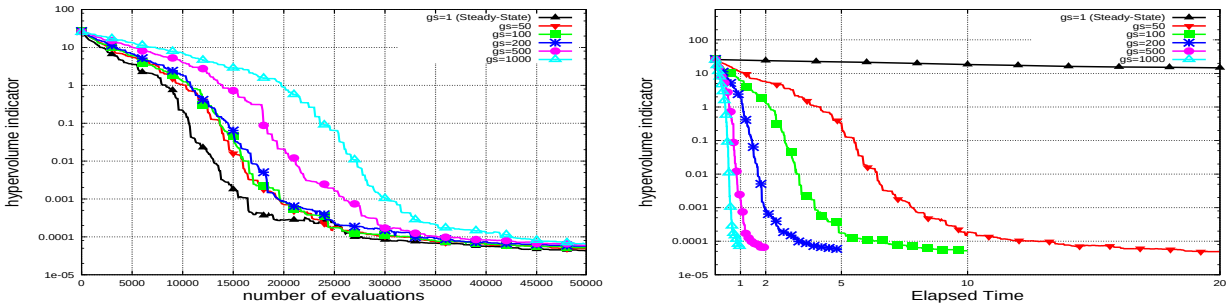

Figure 4: Evolution of average hypervolume indicator for AS-MO-CMA-ES on Rand-VC-ZDT3 for different queue-sizes. Left: in terms of # evaluations. Right: in terms of elapsed time.

To this aim, the test function *Region-VC-ZDT3* is used: it has a discontinuous Pareto Front (clearly visible on Figure 5-right), and the costly part of the Pareto Front (defined by $f_1 \in [0.3, 0.5]$ – a rectangle in objective space) contains one connected component of the true Pareto front. Using AS-NSGA-II, the costly region is not discovered after 50000 evaluations while other sub-parts of the Pareto front are reached (Figure 5-left). A similar though less deleterious effect is visible for AS-MO-CMA-ES ((Figure 5-right): significantly less points are visited in the costly region. However, the algorithm ultimately succeeds in identifying the costly component of the Pareto Front.

6.2 Duration-Based Selection

As seen in previous Section, heterogeneous evaluation costs might have dramatic consequences on the identification of the Pareto Front – and such situations do happen in real applications, as seen in Section 2. In such context, a natural idea to try to counterbalance the slow evaluations is to modify the selection procedure and augment it with some component that takes into account the duration of an evaluation (that the master node can easily monitor).

The basic idea of the *Duration-Based Selection* (DBS) is to use a user-defined probability $P_s \in [0, 1]$ within the (parental) selection to choose between the standard Pareto-based selection of the algorithm at hand, and a tournament selection solely based on the computational cost (the duration in terms of elapsed time) of the last evaluation of the individual. A critical issue will of course be to adjust P_s such that the costly individuals are not favored too much, i.e., that P_s is not too small. It is expected, however, that the individuals that are very costly to evaluate will nevertheless undergo Pareto-based selection at some point, and be eliminated at some point if they represent poor solutions of the multi-objective problem at hand.

Note that only the parental selection is modified. For AS-NSGA-II, parental selection is a tournament (of size 2 in all experiments presented here), and it is replaced with proba. $1 - P_s$ by a tournament of same size but based on evaluation duration. For AS-MO-CMA-ES, there is no parental selection per se. So with probabiliy P_s, the parent is randomly chosen, otherwise, a tournament (of size 2 here, too), is performed based on evaluation durations.

6.3 Experiments with DBS

The first experiment is concerned with the validation of DBS as a repair mechanism for the loss of some parts of the Pareto Front that lie in the costly regions, that was demonstrated in Section 5. Indeed, Figure 6-left illustrates that the use of the DBS-AS-NSGA-II with $P_s = 0.5$ (but similar results have been obtained with other values of P_s) allows the algorithm to find a few points in the costly part of the Pareto Front after 25000 evaluations (the small circles) and to further complete the front of optimal solutions (the crosses fill the Pareto Front after 50000 evaluations). This observation is generalized in figure 6-right which displays the evolution of average hypervolume indicator using 3 different values of P_s: increasing the value of P_s is equivalent to use more the classical dominance based selection, and this clearly delays the discovery of the costly region.

Further experiments were conducted with DBS, in particular in order to check that it does not harm too much the convergence toward the Pareto Front in case there is no heterogeneity in the evaluation costs. The results show that the

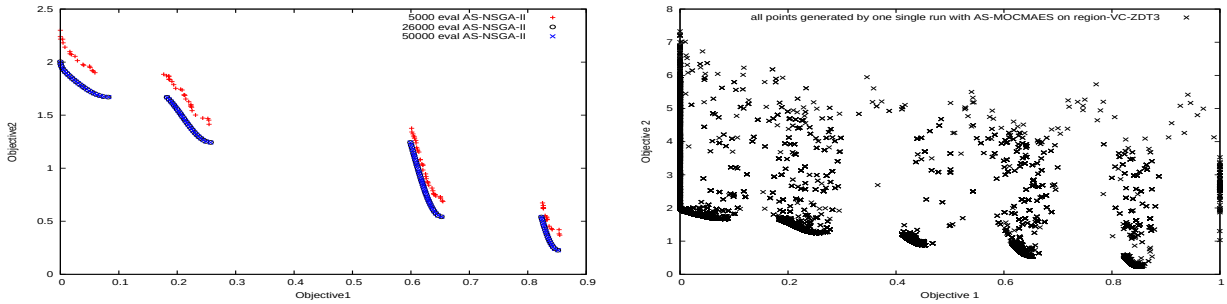

Figure 5: Deleterious effects of heterogenity on Region-VC-ZDT3 (costly region: $f_1 \in [0.3, 0.5]$). Left: Snapshots after 5000, 25000 and 50000 evaluations with AS-NSGA-II. Right: All idividuals visited by one run with AS-MOCMA-ES.

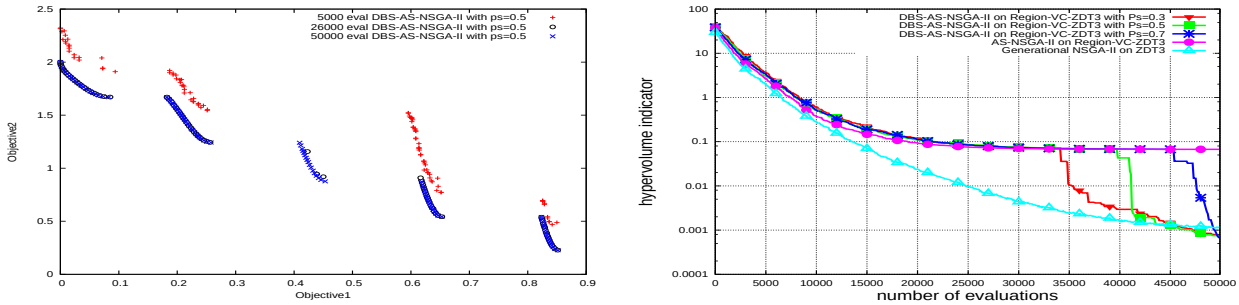

Figure 6: Duration-based selection at work: results of DBS-AS-NSGA-II on Region-VC-ZDT3 (costly region: $f_1 \in [0.3, 0.5]$. Left: Snapshots after 5000, 25000 and 50000 evaluations for $P_s = 0.5$. Right: Evolution of the average hypervolume indicator for different values of P_s

price to pay for the use of duration based selection is rather small, as depicted on Figure 7, comparing different variants of DBS-NSGA-II on *Rand-VC-ZDT3*. Indeed, the DBS variants are slightly outperformed by the standard AS-NSGA-II, but they are still clearly outperform the generational version in terms of quality.

A surprising result has been obtained on function *Region-VC-ZDT6* with costly region defined by $f_1 \in [0.8, 1]$ (see Figure 8-left). In this case, the use of the duration based selection operator delays the convergence instead of accelerating it. The snapshots after different numbers of evaluations (Figure 8-right) confirm that the AS-NSGA-II (equivalent to DBS-AS-NSGA-II with $P_s = 1$) goes faster than DBS-AS-NSGA-II variants, because the DBS selection delays the convergence trying to explore the costly region. The final results, however, are of the same quality for all algorithms in terms of hypervolumes and Pareto Front covering, even for small values of P_s.

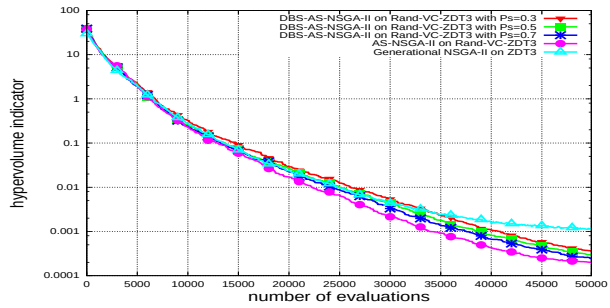

Figure 7: Evolution of average hypervolume indicator for DBS-AS-NSGA-II with different values of P_s on (plain) function ZDT3.

7. CONCLUSION

Motivated by a real-world case study tackling the optimization of the combustion in a Diesel engine, this paper has empirically investigated the behavior of some parallel master-slave asynchronous steady-state MOEAs on artificial test functions. Investigations have been conducted for two well-known state-of-the-art algorithms, NSGA-II and MO-CMA-ES, using the classical test functions ZDT and IHR that were artificially made heterogeneous, simulating different models of heterogeneity.

Sub-linear speed-ups in term of elapsed time have been assessed by considering different numbers of computing units (e.g., grid or cluster sizes). Furthermore, experiments with artificial heterogeneity have first demonstrated the possible negative impact heterogeneity can have on such asynchronous steady-state algorithms: some costly regions of the Pareto Front might be completely missed by the algorithm due to the high computational cost of evaluations – a situation that does take place in real application. In order to tackle this issue, DBS, a specific selection operator, has been proposed, that uses, with a user-defined probability, the evaluation duration instead of Pareto dominance as main comparison criterion. The first results in this direction have demonstrated the ability of MOEAs using DBS to discover

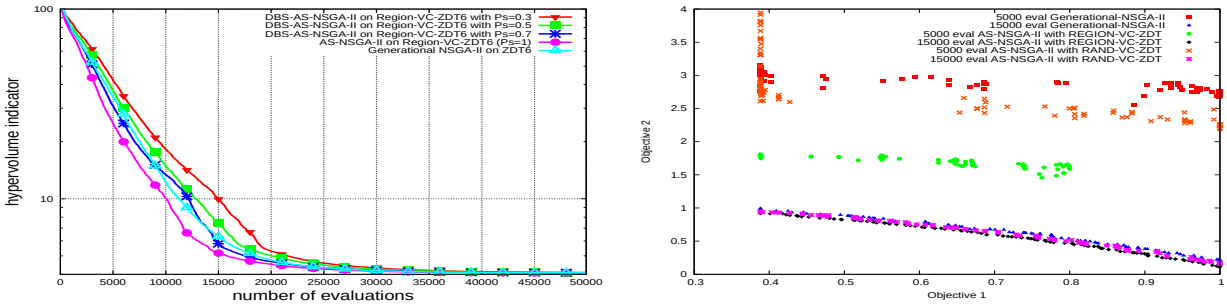

Figure 8: Results for variants of DBS-AS-NSGA-II with different values of P_s on Region-VC-ZDT6 with costly region $f_1 \in [0.8, 1]$. Left: Evolution of average hypervolume indicator. Right: Snapshots after 5000, 15000 and 50000 evaluations of typical runs.

the whole Pareto Front, including the parts in the costly regions. Furthermore, even in cases where DBS slightly slows down the convergence of the underlying MOEA, the quality of the discovered Pareto Front is unchanged.
On-going and future work is to allow a self-adaptation of convergence and coverage by the algorithm. Another perspective is to apply DBS on more realistic models of evaluation costs, for instance on the real-world application of Diesel Combustion that motivated it.

8. REFERENCES

[1] E. Alba and J. Troya. A Survey of Parallel Distributed Genetic Algorithms. *Complexity*, 4(4):31–52, 1999.

[2] V. G. Asouti and K. C. Giannakoglou. Aerodynamic optimization using a parallel asynchronous evolutionary algorithm controlled by strongly interacting demes. *Engineering Optimization*, 41(3):241–257, March 2009.

[3] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[4] J. Branke., H. Schmeck, and K. Deb. Parallelizing multi-objective evolutionary algorithms: Cone separation. In *IEEE CEC*, pages 1952–1957, 2004.

[5] E. Cantu-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Pub, 2000.

[6] D. Chafekar, J. Xuan, and K. Rasheed. Constrained multi-objective optimization using steady state genetic algorithms. In *GECCO*, pages 813–824. 2003.

[7] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[8] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.

[9] K. Deb, M. Mohan, and S. Mishra. Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In C. Fonseca et al., editor, *EMO'03*, pages 222–236. LNCS 2632, Springer Verlag, 2003.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[11] K. Deb., P. Zope, and A. Jain. Distributed computing of pareto-optimal solutions with evolutionary algorithms. In *EMO*, pages 534–549, 2003.

[12] J. Durillo, A. Nebro, F. Luna, and E. Alba. A Study of Master-slave Approaches to parallelize NSGA-II. In *Proc. IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, 2008.

[13] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[14] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[15] C. Igel, T. Suttorp, and N. Hansen. Steady-state selection and efficient covariance matrix update in MO-CMA-ES. In S. Obayashi et al., editor, *EMO'07*, pages pp.171–185. LNCS 2632, Springer Verlag, 2007.

[16] I. Loshchilov, M. Schoenauer, and M. Sebag. Not all parents are equal for mo-cma-es. In Ricardo H. C. Takahashi et al., editor, *Proc. EMO'2011*, LNCS, pages 31–45. Springer Verlag, 2011.

[17] G. Syswerda. A study of reproduction in generational and steady state genetic algorithm. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

[18] M. Yagoubi. *Optimisation évolutionnaire multi-objectif et parallèle: application à la combustion Diesel*. PhD thesis, Université Paris Sud, To appear.

[19] M. Yagoubi, L. Thobois, and M.Schoenauer. Asynchronous evolutionary multi-objective algorithms with heterogeneous evaluation costs. In *IEEE CEC*, pages 21–28, 2011.

[20] M. Yagoubi, L. Thobois, and M. Schoenauer. An Asynchronous Steady-state NSGA-II Algorithm for Multi-Objective Optimization of Diesel Combustion. In H. Rodrigues et al., editor, *Proc. 2nd Intl Conf. on Engineering Optimization – EngOpt'2010*, 2010.

[21] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[22] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, 1999.