'Hang On a Minute': Investigations on the Effects of Delayed Objective Functions in Multiobjective Optimization

Richard Allmendinger[1] and Joshua Knowles[2]

[1] Department of Biochemical Engineering, University College London, UK
r.allmendinger@ucl.ac.uk
[2] School of Computer Science, University of Manchester, UK
j.knowles@manchester.ac.uk

Abstract. We consider a multiobjective optimization scenario in which one or more objective functions may be subject to delays (or longer evaluation durations) relative to the other functions. We motivate this scenario from the viewpoint of experimental optimization problems, and derive several simple strategies for dealing with population and/or archive updates under these conditions. These are embedded in a ranking-based EMO algorithm and tested on the WFG test problems augmented with delayed objective(s). Results indicate that good performance can be achieved when the most recently generated solutions are submitted for evaluation on the delayed objective functions, and missing objective values are approximated using a fitness inheritance-based approach. Also, in general one should wait for all evaluations to complete before resuming search if the delay is short, while a non-waiting strategy should be preferred for longer delays.

1 Introduction

Evolutionary approaches to multiobjective optimization continue to find new applications in a diverse range of scientific and problem-solving contexts, even in areas where existing techniques have considerable history and traction. One area of potential exploitation of EMO methods that is beginning to take off (though is still under-represented) is in applications where the optimization loop involves an experimental (rather than a simulation) step, examples being [14, 20, 19, 6]. Motivated primarily by this area, this paper considers the problem of optimizing several objectives simultaneously in the case where *at least one of them* requires a relatively longer time to evaluate than the 'cheaper' or 'cheapest' of the objective functions. This kind of problem comes about when one objective, for example, involves a lengthy experimental process such as growth, fermentation or such, or perhaps the involvement of human expert(s) input, although we note that it also comes about in the more often considered case where computational simulations are used. We consider here that the objective functions can be evaluated in batches, as is often the case in experimental settings, but that there is not a possibility of speed-up on the expensive objective(s) in terms of the (relative) time taken to evaluate the batch of solutions.

Given this setting, we investigate what strategies one might employ in an EMO algorithm (EMOA) to deal with the delayed objective(s). We simulate the use of surrogate models, which may be appropriate in some contexts, but our main focus is on strategies that do not rely on estimating missing or delayed objective values. In the following section, we note some EA and EMO papers that have looked at similar or perhaps related problems, from which we can and do draw some inspiration. Section 3 defines our problem and reviews some basic properties of dominance relationships in the case of unknown objective values, and this prepares the way for some of the strategies that we go on to detail in Section 4. Experimental results on modified WFG functions are given in Section 5, and Section 6 is a discussion and conclusion.

2 Background and Related Work

All general-purpose approaches to optimization involve a "generate-and-test" loop that must be repeatedly applied in order to discover optimal or high-performing solutions. The iteration of the main loop means that the cost and feasibility of optimization, in any given context, will depend critically on how fast and cheaply solutions can be accurately evaluated. Often, in real applications, it is expensive or time-consuming to evaluate solutions accurately, so that there is much interest in the optimization community in topics around the subject of how to save function evaluations (i.e., designing better optimizers), and also around how to build or use surrogate models of the real evaluation function that are sufficiently accurate to allow optimization to occur but at reduced temporal or financial cost.

Here, we are concerned, in the context of expensive objective functions, with multiobjective optimization: in particular, finding a representative approximation of the entire true Pareto Front. Surrogate modeling is a viable and appropriate approach to tackling many expensive objective functions even in the context of multiple objectives (see [12, 9, 24]), but is a complicated area with many choices to consider for a proper study. Here, our focus is more on the basic design choices of the multiobjective optimizer.

Given this, the key questions are: how should fitness assignment, and population update occur to account for the fact that one or more objective functions are delayed, i.e. that the fitness estimates of some solutions, at any given time, may only be partial? Should we devise or employ selection and update techniques that can deal with partial fitness information, or should we just use a more standard EMOA, and simply wait for all evaluations to complete, no matter the delay, before going on to the next generation? A look at the literature provides useful clues and ideas to try out.

Some papers have considered the idea of finding 'minimal sets of objective functions' [5], such that the subset does not conflict with the full set, and is not redundant. The context of this work is generally different to ours — mostly the concern has been with reducing the number of objective functions down (online or prior to search) in the context of 'many-objective' optimization [16], to facilitate the optimization process. Nevertheless, clearly the effect of objective

function removal is closely related to the effect of objective function delay, where at least some solutions that we wish to rank, or to assign reproductive opportunities, might have a reduced number of objective values (at least temporarily). The difference is that our aim is not to identify which objectives we can neglect, but rather to estimate the effects of neglecting a specific expensive objective in order to determine whether it is worthwhile to use the objective.

Asynchronous evaluation in optimization in the context of grid computing was considered in [13, 17]. The problem overlaps but is distinct from ours in that the cloud computing resource is assumed to be heterogeneous and/or unreliable, and the asynchrony happens across the population rather than across objectives. (In contrast, we assume for the moment a rather reliable and homogeneous process for evaluating a whole population en masse, and are concerned only with the fact that some objectives can be evaluated faster than others.) Although the context is a bit different, we think that as Lewis *et al.*[13] found, a strategy based on a moderate amount of waiting for slower evaluations may be competitive in some settings, and we also consider the effect of diversity maintenance might be important (see below).

If we think of using algorithms that have solutions staying in a memory (i.e., an archive or secondary population) to allow that they can be waiting for their expensive objective function values to be computed, and that these solutions are potentially part of the present or future breeding pool, then we may have solutions from some number of generations in the past needing to take part in reproduction. In this context, the use of age-layered populations [10] may be a neat way to handle population update and selection matters. We consider an adaptation of this architecture in our work here.

We have learnt through the development of EMO over the last years that diversity preservation, or methods for ensuring objective space spread are very important to obtain good approximation sets (e.g., if we are considering sensible measures of performance such as hypervolume or epsilon-dominance, at least for external evaluation). But the diversity of solutions in the objective space is going to be difficult to estimate when many solutions are missing one or more objective values; in fact this problem may be more severe than the adaptations necessary to deal with Pareto ranking of solutions. Given this issue, it may be sensible to revert to the use of decision space diversity [18, 22] in place of objective space, as we assume that decision space information is quick and cheap to use.

Although we do not wish to cloud our initial studies by complex considerations concerning the use of surrogate modeling techniques, it does seem that one of the most direct approaches to handling delayed objective values is to use estimated objective values in their place (at least until true values become available). In this regard, we consider a few simple methods of estimating objective values, inspired in part by work on fitness inheritance [21, 15]. Simple methods of missing value imputation from machine learning might also be used (see [23]).

Finally, we note that the problem of delayed objective function values has some relationship to our recent work on *ephemeral resource constraints* (ERCs) in single-objective optimization [2, 3, 1, 4]. ERCs are temporary limitations in the capacity to evaluate certain otherwise feasible solutions *during the optimization process*. ERCs arise in experimental optimization settings due to external

factors such as machine breakdowns, limited availability of certain reagents or chemicals under test, or human experts with limited availabilities, and usually they only affect part of the feasible search region at any given time (often also as a function of previous actions). Delayed objectives, by contrast, prevent (immediate) evaluation of *all* the solutions of a batch, but in only some objectives. A key finding from our work with ERCs is that waiting for objective values is quite often the best thing to do, but it can depend on several other factors about the ERCs, such as how long they will last, how much of the search space is affected, and so on. We expect that also in the case of delayed objectives in EMO, simply applying a standard algorithm and waiting for objective functions to return may be the best thing to do in many cases. But we would like to discover the situations where this is not the case, and what other strategies may be sensible. As we found with ERCs, significant savings may sometimes be possible even with a minimal need for information about the problem [2].

3 Problem Formulation and Pareto Dominance Considerations

We augment the notion of delayed objective functions onto a multiobjective optimization problem as follows:

$$\begin{aligned} &\text{minimize } \ (f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))^T \\ &\text{subject to } \ \boldsymbol{x} \in X, \end{aligned} \tag{1}$$

where $\boldsymbol{x} = (x_1, ..., x_l)$ is a *solution vector* and X a *feasible search space*. The static *objective functions* $f_i, i = 1, ..., m$ are to be minimized and each function is associated with some *evaluation delay* of $\Delta t_i \geq 0$ time steps (e.g. hours or days) relative to the objective(s) that is (are) quickest to evaluate. That is, $\Delta t_i = 0$ means that function i is quickest to evaluate and thus has no delay, while $\Delta t_i > 0$ means that function i needs Δt_i time steps longer to be evaluated than the quickest objective. There is at least one function with delay, i.e. $\exists i \in \{1, ..., m\} : \Delta t_i > 0$.

In this study, each function f_i is evaluated in a batch of $k_i > 0$ solutions, and it takes one time step to evaluate this batch. If not otherwise stated, we assume an optimization scenario with exactly one delayed objective function; this will be always function f_m having a delay of $\Delta t_m > 0$, and we set $\Delta t_i = 0, i = 1, ..., m-1$ for the other functions. In this setup the following two Pareto dominance relations hold, which we will incorporate later in some of our strategies for dealing with delayed objective functions.

Lemma 1. *Let S be a set of points for which $m-1$ objective values are all known, and the mth objective values are all unknown. Then if all solutions in S are non-dominated with respect to the $m-1$ objectives, it follows that all solutions in S are non-dominated to each other irrespective of their mth objective values.*

Lemma 2. *Let S be a set of points for which $m-1$ objective values are all known, and the mth objective values are all unknown. Then (a) the minimum*

Algorithm 1 Ranking-based EMOA for optimizing subject to delayed objective functions

Require: $f_1, ..., f_m, \Delta t_m > 0, (\Delta t_1 = 0, ..., \Delta t_{m-1} = 0), \mu = \lambda$, T (time limit)

$t = 0$ (time counter), $Pop = \emptyset$
// Initialize Population:
Pop = random_generate_n_solutions($n = \mu$)
evaluate_pop($Pop, f_1, \dots f_{m-1}$), assign_pseudovalues_to_expobjective(Pop, f_m), $t = t + 1$ // evaluation of non-delayed objectives only, and assignment of pseudovalues to the delayed objective f_m
endtime = evaluate_pop_expensive(Pop, f_m, *currenttime* $= t$) // spawns parallel thread to evaluate Pop on delayed objective; immediately returns the projected end time for spawned process; sets Pop's mth objective value to 'pending'
// Main Loop:
while $t < T$ **do**
  rank(Pop, ranking_method) // ranking method must account for missing (delayed) objective values of some solutions
  $ParentPop$ = parental_selection(Pop)
  $OffPop$ = crossover_and_mutation($ParentPop$)
  evaluate_pop($OffPop, f_1, \dots f_{m-1}$), $t = t + 1$ // evaluation of non-delayed objectives only
  $Pop = Pop \cup OffPop$
  assign_pseudovalues_to_expobjective(Pop, f_m) // (re)assignment of pseudovalues to delayed objective f_m
  if ($t =$ *endtime*) **then**
    pending objective values are now updated
    $EvalPop$ = selection_for_expevaluation(Pop) // decides which μ solutions from Pop to evaluate on the delayed objective f_m; only selects from solutions that have no value for f_m
    endtime = evaluate_pop_expensive($EvalPop, f_m$, *currenttime* $= t$)
return (Pop)

number of different non-dominated sorting (NDS) ranks in S is 1, and (b) the maximum number of different ranks is the number of NDS ranks existing amongst the solutions in S in the $m - 1$ known objectives plus the number of points that are equal with respect to the known objectives.

4 Strategies for Dealing with Optimizations Problems Featuring Delayed Objective Functions

As the basis for our strategies we use a ranking-based EMOA as shown by Algorithm 1. Unlike standard EMOAs, the size of the population Pop in this EA is not fixed. This way we allow solutions with missing objective function values to influence the search direction and be evaluated at any point in time during the optimization. Assuming that the delayed objective function f_m is evaluated in a batch of $k_m = \mu$ solutions, the EA begins the optimization by generating a set of μ solutions at random, evaluating them on the non-delayed

objectives only and assigning pseudovalues to the delayed objective (Line 3). At $t = 0$, all solutions are submitted for evaluation on the delayed objective function f_m, and their mth objective values are set to 'pending' (Line 4). The projected end time *endtime* of the delayed objective represents the time step at which the pending objective values are updated (i.e. revealed) (Line 13), and a set of new μ solutions for evaluation on the delayed objective selected (Line 14). Each generation, the population is first ranked (Line 6), and then λ offspring generated by a process of selection, crossover and mutation (Line 7 and 8), and evaluated on the non-delayed objective functions (Line 9). Following this, all offspring are added to *Pop* and pseudovalues are (re)assigned to all solutions in *Pop* that have not been evaluated on f_m (Line 11); reassigning pseudovalues to solutions reduces the risk that these solutions take over the population and potentially misguide the search. Our EMOA ensures that the population *Pop* does not contain duplicate solutions, i.e. offspring are generated until we have a set that has not been evaluated yet.

In the following we describe various modifications to the EMOA we are going to investigate in the presence of delayed objective functions. In particular, we look at different methods for the assignment of pseudovalues to the delayed objective (Line 3 and 11), ranking (Line 6), parental selection (Line 7) and the selection of solutions for evaluation on the delayed objective function (Line 14).

Assignment of pseudovalues to delayed objectives. We investigate three assignment strategies. The first strategy, *random pseudovalue assignment*, assigns to each solution with a missing objective value a pseudovalue drawn at random from the interval $[\min_{Pop} f_m, \max_{Pop} f_m]$, where $\min_{Pop} f_m$ and $\max_{Pop} f_m$ is the minimum and maximum value of objective f_m of all solutions in *Pop* that have actually been evaluated on objective f_m. The second strategy, *noise-based pseudovalue assignment*, draws for each solution with a missing objective value a random solution from *Pop* that has been evaluated on all objectives (including the delayed objective), and adds a small amount of (Gaussian distributed) noise $\mathcal{N}(0, \sigma^2)$ to the value of the delayed objective; the resulting value is used as the pseudovalue for the delayed objective. The third strategy, *fitness inheritance-based pseudovalue assignment*, selects for each solution with a missing objective value a solution from *Pop* that is both closest to it in the decision space (in terms of normalized Euclidean distance) and has been evaluated on all objectives, and then simply takes over the delayed objective value of this solution.

Ranking subject to missing objective values. We investigate two ranking schemes. The first scheme, *performance ranking*, sorts all solutions in *Pop* according to their non-dominated sorting (NDS) ranks only. In contrast, the second ranking scheme, *performance+age ranking*, considers both the NDS rank and the time stamp at which a solution has been generated. More precisely, first the NDS ranks of all solutions in *Pop* are obtained (these ranks are used later as quality criterion in parental selection), and then the population is sorted based on the age of solutions whereby more recently generated solutions are favoured (this sorting affects the truncation selection only).

Determining a parent population *ParentPop*. Once the population *Pop* has been ranked we need to decide which solutions should be eligible for parental selection (as our population is not limited in size). In a standard EA setup with a fixed population size, this design choice would correspond to the reproduction scheme (or environmental selection mode). We investigate two schemes: parental selection among (i) the top ranked μ solutions in *Pop* (generational reproduction scheme) (denoted in future by GGA) and (ii) the top ranked $\mu \times 2$ solutions ($(\mu + \lambda)$-ES reproduction scheme) (denoted in future by $(\mu + \lambda)$-ES).

Selecting solutions for evaluation on the delayed objective function. We investigate two strategies to decide which μ solutions from *Pop* to evaluate on f_m. The first strategy, *sweep selection*, selects always the most recently generated μ solutions for evaluation on f_m. The second strategy, *priority-based selection*, assigns to each solution without a value for f_m, a score representing the solution's priority of being evaluated. We compute this score by first obtaining the NDS ranks, considering the objectives $f_1, ..., f_{m-1}$ only, of all completely evaluated solutions in *Pop*. Then, the priority score of a solution is estimated based on Lemma 1 and 2 (see Section 3), and also on the idea of counting the total amount the ranking of all (completely evaluated) solutions could be changed: If a solution with no value for f_m is dominating all solutions, then potentially it could demote all of these solutions by one rank after revealing the value of objective f_m (i.e. we have a priority score equal to the number of completely evaluated solutions in *Pop*). If a solution with no value for f_m is dominated by all solutions, then it cannot possibly dominate any solution (although it might be non-dominated if the value of f_m is very small) (i.e. we a have a priority score of zero). In all other cases, a solution with no value for f_m can potentially dominate some solutions in *Pop*; here we assign the solution a priority equal to the number of solutions having a lower rank.

The modifications described above enable an EMOA to deal with partial fitness information. We will investigate also an EMOA that waits for all evaluations to complete and thus prevents having to deal with solutions with missing objective values; hence for this algorithm it makes sense to look only at modifications related to the ranking of solutions (Line 6) and the determination of the parent population *ParentPop* (Line 7).

5 Experimental Study

This section describes the test functions and the parameter settings as used in the subsequent experimental analysis, which investigates the performance of the strategies described above when applied to problems with delayed objectives.

5.1 Experimental setup

Our aim in this study is to understand the effect of delayed objective functions on EA performance. To cover a wide range of problem characteristics, we use the Walking Fish Group (WFG) toolkit [11]. We use the toolkit with 4 distance

Table 1. EA parameter settings.

Parameter	*Setting*
Parent population size μ $(= k_m)$	50
Offspring population size λ	50
Per-variable mutation probability p_m	$1/l$
Crossover probability p_c	0.9
Distribution index (mutation and crossover)	20
Time limit T	40

parameters and 2 position parameters within the standard WFG1-WFG9 test problems; i.e. we have $l = 4 + 2 = 6$ continuous decision variables. If not otherwise stated we use the WFG problems with $m = 3$ objectives, with f_3 being the objective function delayed by $\Delta t_3 = 3$ time steps. We set the batch size associated with this objective function identical to the population size, i.e. $k_3 = \mu$; the batch sizes $k_i, i = 1, ..., m-1$ are irrelevant as there is no delay on the associated objective functions.

We augment the strategies described in Section 4 on a ranking-based EMOA (see Algorithm 1) that uses binary tournament selection (with replacement) for parental selection, simulated binary crossover (SBX) [7], and polynomial mutation [8]. The parameter settings of the EMOA are given in Table 1.

For the noise-based pseudovalue assignment strategy we use a noise level of $\sigma = \sqrt{(\max\limits_{Pop} f_m - \min\limits_{Pop} f_m) * 0.05}$, where, as in the case of the random pseudovalue assignment method, $\max\limits_{Pop} f_m$ and $\min\limits_{Pop} f_m$ is the maximum and minimum value of objective f_m of all solutions in Pop that have actually been evaluated on objective f_m so far. To reduce the risk of any search bias, the parameters $\max\limits_{Pop} f_m$ and $\min\limits_{Pop} f_m$ are set initially to large positive and negative numbers, here 1000 and -1000, respectively. We set the time limit to $T = 40$ time steps.

Any results shown are average results across 20 independent algorithm runs. We use paired comparison by employing a different seed for the random number generator for each EA run but the same seeds for all strategies described above.

5.2 Experimental results

Table 2 gives us an initial overview of the performance (hypervolume measurements) of some of the algorithm modifications on all WFG test problems.[3] Results were obtained using an EMOA with random pseudovalue assignment and

[3] The hypervolume measurements were obtained by normalizing the non-dominated front found by an EA at the end of an optimization run, and then taking the average of the hypervolume measurements across 20 runs. The normalization was done based on the extremal values of the estimated True Front, which is available online at `http://jmetal.sourceforge.net/problems.html`, and the reference point was set to the minimum and maximum values of the normalized front.

Table 2. Average hypervolume values obtained in an environment with and without (in parenthesis) delayed objective functions for different algorithm setups on the WFG test problems using $m = 3$ objectives. All EAs optimizing subject to delayed objective functions employed random pseudovalue assignment and sweep selection. For each problem instance and optimization environment (with delay vs without delay), we highlighted all algorithm setups in bold face that are not significantly worse than any other setup. A Friedman test revealed a significant difference between the search algorithm setups in general, but differences among the individual setups were tested for in a post-hoc analysis using (paired) Wilcoxon tests (significance level of 5%) with Bonferroni correction.

		GGA		$(\mu + \lambda)$-ES	
		waiting	no waiting	waiting	no waiting
WFG1	Performance ranking	0.1015	**0.1976** (**0.1939**)	0.0940	0.1682 (**0.1916**)
	Performance+age ranking	0.0885	0.1077 (0.0982)	0.0875	0.0992 (0.1016)
WFG2	Performance ranking	**0.6901**	0.6748 (0.8527)	0.6474	0.6655 (**0.8616**)
	Performance+age ranking	0.6028	0.5603 (0.6822)	0.6068	0.5819 (0.6958)
WFG3	Performance ranking	**0.4292**	0.4030 (**0.4639**)	**0.4214**	0.3915 (**0.4624**)
	Performance+age ranking	0.4129	0.3970 (0.4270)	**0.4260**	0.4000 (0.4396)
WFG4	Performance ranking	**0.3434**	0.2701 (**0.4172**)	0.3362	0.2503 (**0.4199**)
	Performance+age ranking	0.2935	0.2468 (0.3728)	0.3015	0.2561 (0.3598)
WFG5	Performance ranking	**0.3430**	0.2676 (**0.4020**)	0.3353	0.2925 (**0.4012**)
	Performance+age ranking	0.3284	0.2888 (0.3633)	0.3290	0.2996 (0.3667)
WFG6	Performance ranking	**0.3457**	0.2356 (**0.3943**)	0.3240	0.2646 (**0.3918**)
	Performance+age ranking	0.3214	0.2701 (0.3410)	0.3252	0.2888 (0.3464)
WFG7	Performance ranking	**0.3447**	0.2760 (0.4096)	**0.3408**	0.3004 (**0.4163**)
	Performance+age ranking	0.3298	0.2862 (0.3787)	0.3346	0.2975 (0.3722)
WFG8	Performance ranking	**0.3129**	0.2497 (**0.3721**)	0.3021	0.2550 (**0.3687**)
	Performance+age ranking	0.2900	0.2492 (0.3134)	0.3047	0.2556 (0.3172)
WFG9	Performance ranking	**0.3743**	0.3260 (**0.4330**)	0.3596	0.3402 (**0.4224**)
	Performance+age ranking	0.3242	0.2865 (0.3727)	0.3332	0.2912 (0.3858)

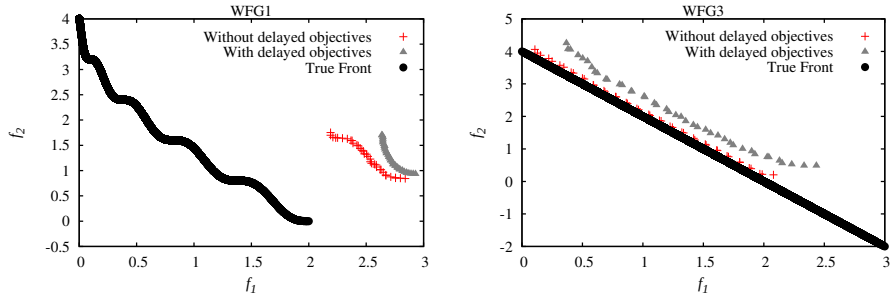

Fig. 1. Plots showing the true Pareto Front, and median attainment surface (across 20 runs) obtained on WFG1 (left) and WFG3 (right) with $m = 2$ objectives in an environment with delayed objective functions (objective function f_2 was subject to a delay of $\Delta t_2 = 3$, and $k_2 = \mu$) and without. The EMOA was equipped with a generational reproduction scheme (GGA), sweep selection, random pseudovalue assignment, performance ranking, and waited for all evaluations to complete before resuming search.

sweep selection. We can make several observations from the table: (i) optimizing subject to delayed objective function affects the performance negatively on all problems except WFG1; (ii) a generational reproduction scheme without elitism (GGA) tends to perform best in the presence of delayed objectives (when waiting is applied), while there is no clear winner between GGA and $(\mu+\lambda)$-ES in an environment without delays; (iii) waiting for all evaluations to complete performs best on all problems except WFG1 and WFG2; (iv) performance ranking generally performs better in an environment with and without delayed objectives.

When optimizing subject to delayed objective functions, the convergence speed seems to be reduced and a more diverse population maintained; these properties tend to be amplified as the delay Δt_m becomes larger, ultimately causing the performance to reduce (as will be seen later from Figure 4). For WFG1, however, the presence of a delayed objective can yield better results than obtainable in an unconstrained environment (observation (i)); in general, we observed that the experimental results obtained on WFG1 are different than on the other WFG problems, which may be due to the structure of this problem (WFG1 is separable, uni-modal and has a dissimilar weight structure [11]). For WFG2-WFG9, assigning pseudovalues to the delayed objective that are too far away from the true objective values can lead to misguidance when performing ranking and parental selection. The risk of misguidance can be reduced when employing an EMOA with a waiting strategy and a generational reproduction scheme (observation (ii) and (iii)). Observation (iv) implies that parental selection should not be limited to a subset (the most recently generated solutions) of the population.

Figure 1 and 2 show visually the performance impact of delayed objective functions on WFG1 and WFG3 using $m = 2$ and 3 objectives, respectively. Plots are showing the median attainment surface across 20 runs obtained by

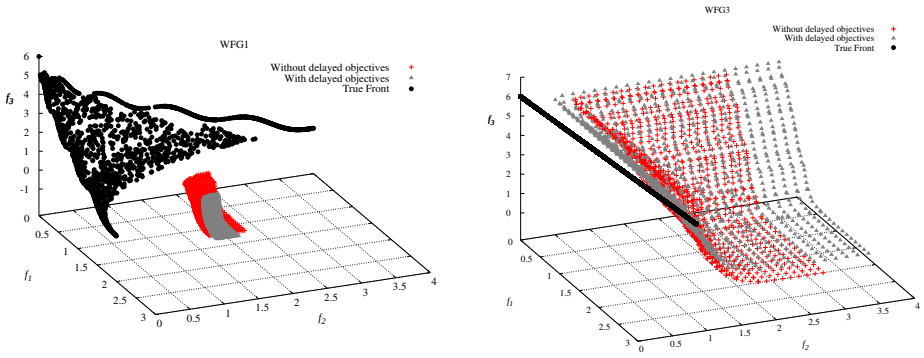

Fig. 2. Plots showing the true Pareto Front, and the median attainment surface (across 20 runs) obtained on WFG1 (left) and WFG3 (right) with $m = 3$ objectives in an environment with and without delayed objective functions. The EMOA was equipped with a generational reproduction scheme (GGA), sweep selection, random pseudovalue assignment, performance ranking, and waited for all evaluations to complete before resuming search.

the best performing EMOA from Table 2. The plots indicate that the impact of a delayed objective function on the performance of an optimizer depends on (i) the characteristics of the fitness landscape of a problem and (ii) the number of objectives to be optimized, an observation we will make again later.

Figure 3 investigates the effect of different delay lengths Δt_m on the performance of our strategies for WFG1 (top plots) and WFG2 (bottom plots); note that the setting $\Delta t_m = 0$ means there is no delay and thus refers to an unconstrained optimization scenario. The left and right plots show the performance impact for EMOAs employing sweep selection and priority-based selection, respectively. All results were obtained using a generational reproduction scheme (GGA), and performance ranking; this setup yielded best results as shown previously. We make the following observations from the figure.

- Generally, an increase in the delay length Δt_m affects search negatively, and algorithm choice is important. Interestingly, a performance improvement may be obtained for short delays (see range $0 < \Delta t_m < 7$ in the top left plot). It seems to be the case that this is due to an increase in the population diversity, although other factors may be responsible.
- Sweep selection clearly outperforms priority-based selection on both test problems (and the other WFG problems for which the results are not shown here) and for all values of $\Delta t_m > 0$. The reason is that priority-based selection ignores completely the values of the delayed objective f_m when selecting solutions for evaluation on the delayed objective. This may lead to misguidance in the selection and stagnation in the search.
- With respect to the delay length Δt_m, we observe for WFG2 and sweep selection (bottom left plot) that there is a value ($\Delta t_m = 5$) at which one should switch from a waiting strategy to a non-waiting one (when using sweep se-

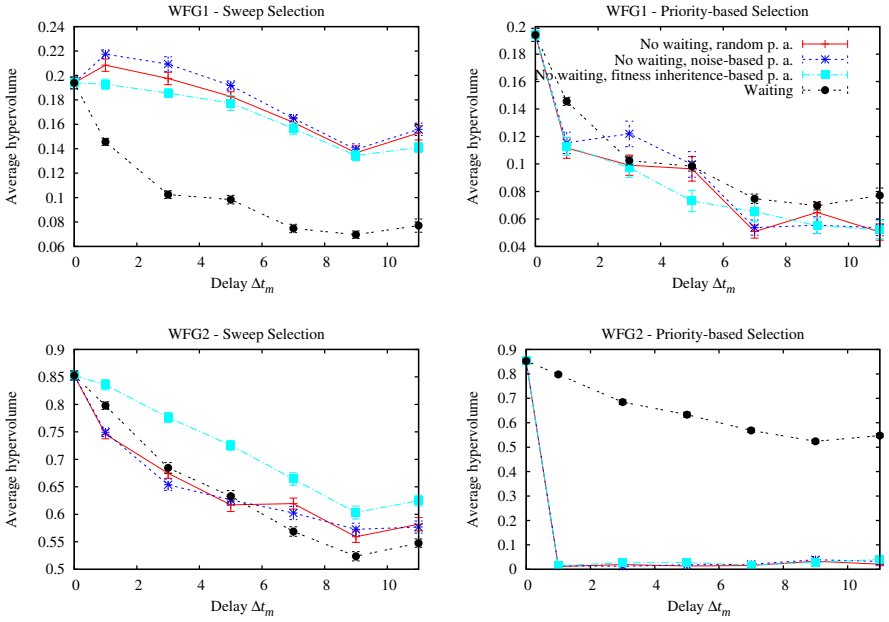

Fig. 3. Plots showing the average hypervolume (and its standard error, indicated by the error bars) obtained on WFG1 (top plots) and WFG2 (bottom plots) with $m = 3$ objectives using EA setups employing sweep selection (left plots) and priority-based selection (right plots). All results were obtained using an EMOA equipped with a generational reproduction scheme (GGA) and performance ranking.

lection and a random or noise-based pseudovalue assignment; this pattern was apparent for all WFG problems except WFG1). The larger Δt_m the slower is the search progress when employing a waiting strategy. The length of the delay Δt_m at which the switch from a waiting to a non-waiting strategy should be performed depends on the difficulty of the problem at hand, and the optimization time available. For WFG1 and sweep selection (top left plot) a non-waiting strategy should be employed for all values of Δt_m. When using priority-based selection, a waiting strategy should be preferred over a non-waiting one because it removes the risk of getting the priority scores wrong and thus submitting non-promising solutions for evaluation on the delayed objective.

- Fitness-inheritance based pseudovalue assignment combined with sweep selection, and no waiting yields best performance on WFG2 (and all the other WFG problems except WFG1). The reason is that the fitness inheritance-based method is able to approximate the value of the delayed objective f_m better than the other two pseudovalue assignment strategies, reducing the risk of misguidance in the search.

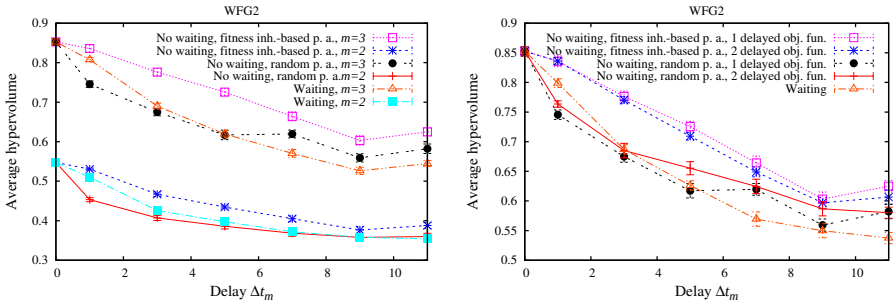

Fig. 4. Plots showing the average hypervolume (and its standard error, indicated by the error bars) obtained by different EA setups on WFG2 with $m = 2$ and 3 objectives (using 1 delayed objective function) (left) and $m = 3$ objectives using 1 and 2 delayed objective functions (in case of 2 delayed objective functions, a delay was on f_2 and f_3 with $k_2 = k_3 = \mu$). All results were obtained using an EMOA equipped with a generational reproduction scheme (GGA), sweep selection, and performance ranking.

Finally, Figure 4 shows some initial results on how the search performance is affected on WFG2 by the number of objectives m to be optimized (left) and the number of delayed objective functions (right) as a function of the delay length Δt_m. From the left plot we observe that whilst the performance is affected in similar way for $m = 2$ and 3 objectives, there is a smaller performance gap between different strategies for $m = 2$. This pattern was also apparent for the other WFG problems (results not shown) and may indicate that lower-dimensional problems are easier to deal with in the presence of delayed objectives.

From the right plot in Figure 4 we observe that having two instead of one delayed objective function (i.e. being more uncertain about the quality of a solution) improves the performance of strategies that approximate missing objective values poorly (random pseudovalue assignment) but degrades the performance of otherwise accurate approximation techniques (fitness inheritance-based pseudovalue assignment). This pattern was apparent for all WFG problems except WFG1, and may indicate that there is a trade-off between increasing the risk of misguidance (leading to reduction in the performance when using an accurate approximator) and increasing the probability that truly poor solutions are approximated poorly and thus not considered for evaluation on the delayed objective function (improvement in performance when using a poor approximator).

6 Summary and conclusion

In this paper we have considered a multiobjective optimization scenario in which at least one objective function may be subject to delays relative to the other functions. In other words, some objective functions take longer to be evaluated than others. This kind of problem can be encountered, for example, when the evaluation of an objective function involves a lengthy experimental process such as growth or fermentation, or the involvement of human expert(s) input. We have

proposed several strategies to deal with this kind of optimization scenario — concerning the pseudovalue assignment to delayed objectives, population ranking, reproduction scheme, and the selection of solutions for evaluation on the delayed objective functions — and assessed them on the (continuous) WFG test problems.

The experimental study revealed that delayed objective functions affect the search performance of an EA — in general, the longer the delay the poorer is the search performance — but that a well-tuned optimizer can damp the performance impact significantly. In particular, when optimizing subject to delays, we can tentatively conclude that one should: (i) employ a fitness inheritance-based pseudovalue assignment (i.e. fill missing objective values of a solution with the objective values of the genetically closest and fully evaluated solution), (ii) select parents for reproduction from a population that is sorted based on the solutions' non-dominated sorting ranks (without accounting for the time stamps at which solutions have been created), (iii) use a generational reproduction scheme (without elitism), and (iv) submit the most recently generated solutions for evaluation on the delayed objectives. Furthermore, we found that, in general, for short delays one should wait for all evaluations to complete before continuing with the next generation. When the delay is long, however, waiting slows down the search and should be avoided. Finally, we have seen that our observations hold for problems with two and three objectives, and that varying the number of delayed objective functions has interesting implications on the performance depending on the algorithm setup employed.

Our study has shown that EA performance crucially depends on the way pseudovalues are assigned and solutions for evaluation on the delayed objective functions selected. We believe there are still some performance improvements to gain by tuning these two aspects, which usually do not need to be considered in the design of an EA. Investigating the effect of delayed objective functions on many-objective optimization problems where several objectives are subject to delays of different durations is another important avenue to be pursued.

References

1. R. Allmendinger. *Tuning Evolutionary Search for Closed-Loop Optimization*. PhD thesis, School of Computer Science, The University of Manchester, 2012.
2. R. Allmendinger and J. Knowles. On-line purchasing strategies for an evolutionary algorithm performing resource-constrained optimization. *Parallel Problem Solving from Nature–PPSN XI*, pages 161–170, 2011.
3. R. Allmendinger and J. Knowles. Policy learning in resource-constrained optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2011)*, pages 1971–1978. ACM Press, 2011.
4. R. Allmendinger and J. Knowles. On handling ephemeral resource constraints in evolutionary search. *Evolutionary Computation*, 2013. (Accepted. In Press).
5. D. Brockhoff and E. Zitzler. Objective reduction in evolutionary multiobjective optimization: theory and applications. *Evolutionary Computation*, 17(2):135–166, 2009.
6. F. Caschera, M. Hanczyc, and S. Rasmussen. Machine learning for drug design, molecular machines and evolvable artificial cells. In *Proceedings of the 13th annual*

conference companion on Genetic and evolutionary computation, pages 831–832. ACM, 2011.
7. K. Deb and R. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1994.
8. K. Deb and M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.
9. M. Emmerich, K. Giannakoglou, and B. Naujoks. Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *Evolutionary Computation, IEEE Transactions on*, 10(4):421–439, 2006.
10. G. Hornby. ALPS: the age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 815–822. ACM, 2006.
11. S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.
12. J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.
13. A. Lewis, S. Mostaghim, and I. Scriven. Asynchronous multi-objective optimisation in unreliable distributed environments. *Biologically-Inspired Optimisation Methods*, pages 51–78, 2009.
14. S. O'Hagan, W. Dunn, M. Brown, J. Knowles, and D. Kell. Closed-loop, multiobjective optimization of analytical instrumentation: gas chromatography/time-of-flight mass spectrometry of the metabolomes of human serum and of yeast fermentations. *Analytical Chemistry*, 77(1):290–303, 2005.
15. T. P. Runarsson. Constrained evolutionary optimization by approximate ranking and surrogate models. *Parallel Problem Solving from Nature–PPSN VIII*, pages 401–410, 2004.
16. D. Saxena and K. Deb. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In *Evolutionary Multi-Criterion Optimization*, pages 772–787. Springer, 2007.
17. I. Scriven, D. Ireland, A. Lewis, S. Mostaghim, and J. Branke. Asynchronous multiple objective particle swarm optimisation in unreliable distributed environments. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2481–2486. IEEE, 2008.
18. O. Shir, M. Preuss, B. Naujoks, and M. Emmerich. Enhancing decision space diversity in evolutionary multiobjective algorithms. In *Evolutionary Multi-Criterion Optimization*, pages 95–109. Springer, 2009.
19. O. Shir, J. Roslund, Z. Leghtas, and H. Rabitz. Quantum control experiments as a testbed for evolutionary multi-objective algorithms. *Arxiv preprint arXiv:1112.5424*, 2011.
20. B. Small, B. McColl, R. Allmendinger, J. Pahle, G. López-Castejón, N. Rothwell, J. Knowles, P. Mendes, D. Brough, and D. Kell. Efficient discovery of anti-inflammatory small molecule combinations using evolutionary computing. *Nature Chemical Biology*, 2011. Published online, 23 October. DOI:10.1038/nchembio.689.
21. R. Smith, B. Dike, and S. Stegmann. Fitness inheritance in genetic algorithms. In *Proceedings of the 1995 ACM symposium on Applied computing*, pages 345–350. ACM, 1995.
22. T. Ulrich, J. Bader, and L. Thiele. Defining and optimizing indicator-based diversity measures in multiobjective search. *Parallel Problem Solving from Nature–PPSN XI*, pages 707–717, 2011.

23. S. van Buuren. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical methods in medical research*, 16(3):219–242, 2007.
24. I. Voutchkov and A. Keane. Multiobjective optimization using surrogates. In *Adaptive Computing in Design and Manufacture (ACDM 2006)*, pages 167–175. The M.C.Escher Company, Netherlands, 2006.