Multiobjective Optimization: When Objectives Exhibit Non-Uniform Latencies

Richard Allmendinger[a,*], Julia Handl[b], Joshua Knowles[c]

[a]*Department of Biochemical Engineering, University College London, United Kingdom*
[b]*Manchester Business School, University of Manchester, United Kingdom*
[c]*School of Computer Science, University of Manchester, United Kingdom*

Abstract

Building on recent work by the authors, we consider the problem of performing multiobjective optimization when the objective functions of a problem have differing evaluation times (or latencies). This has general relevance to applications since objective functions do vary greatly in their latency, and there is no reason to expect equal latencies for the objectives in a single problem. To deal with this issue, we provide a general problem definition and suitable notation for describing algorithm schemes that can use different evaluation budgets for each objective. We propose three schemes for the bi-objective version of the problem, including methods that interleave the evaluations of different objectives. All of these can be instantiated with existing multiobjective evolutionary algorithms (MOEAs). In an empirical study we use an indicator-based evolutionary algorithm (IBEA) as the MOEA platform to study performance on several benchmark test functions. Our findings generally show that the default approach of going at the rate of the slow objective is not competitive with our more advanced ones (interleaving evaluations) for most scenarios.

Keywords: Multiobjective optimization, evolutionary computation, delayed objective functions, closed-loop optimization, budgeted optimization.

1. Introduction

Multiobjective optimization problems require the simultaneous optimization of multiple (often conflicting) objectives over a given space of candidate solutions. These problems occur in many practical applications, rather often as bi-objective problems, with typical pairs of objectives as quality vs cost, strength vs weight, or accuracy vs complexity (e.g., of a model). A concrete example of a multiobjective problem is the design of a bridge[1], such that it passes one or more strength tests, and such that its cost of construction is not too high.

Although in tackling such problems, it is possible to treat one objective as a constraint, or to weight or prioritize objectives to form a scalar optimization problem, a more general approach to multiobjective optimization (and the one we follow here) is to postpone or avoid the assignment of weights or priorities, and instead to seek a representation of all the optimal trade-offs of the objectives (the Pareto optimal front) to allow posterior exploration of the optimal choices, and a final solution to be selected (see Figure 1).

The problem that we identify and tackle in this paper stems from the fact that, to date, almost all such methods for multiobjective optimization, including the many methods based on evolutionary algorithms, assume that each candidate solution is evaluated on all the objectives *simultaneously*. Thus, every candidate solution explored is associated with its vector of objective values, it can be plotted in the objective space (as in Figure 1), and more importantly it can take part in relative assessments of its multiobjective fitness (or utility) so that fitness-biased selection (particularly) can be carried out. However, given that the objective functions to be evaluated could be of quite different character, this means these algorithms are somewhat overly-restricted, and could be inappropriate or inefficient for cases, under a time-budgeted optimization (see Jansen & Zarges (2013)), where the different objective functions vary in evaluation times (or latencies).

Consider an extreme example. We wish to optimize the formulation of a washing powder, and our two objectives are washing excellence and cost. In this case, it is easy to imagine that assessing washing excellence

*Corresponding author. Tel.: +44 (0)20 7679 7745
Email address: r.allmendinger@ucl.ac.uk (Richard Allmendinger)

[1]Which may indeed be made of concrete — no pun intended.

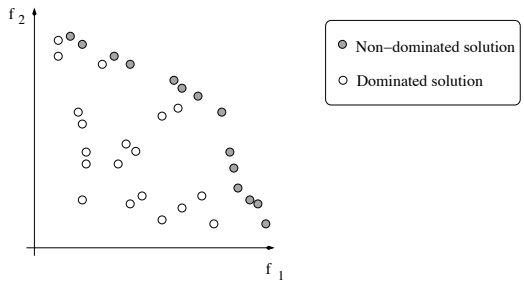

Figure 1: Solutions to a discrete optimization problem plotted in bi-objective space. Assuming only that the decision maker is consistent in her preference for larger values of each objective over smaller values, she must prefer the non-dominated solutions (shaded). If the set shown is the whole solution space, then the non-dominated solutions form the Pareto Front and are the optimal tradeoff solutions to the problem.

may be a laborious process involving testing the powder, perhaps on different materials and at different temperatures. By contrast, the cost of the particular formulation can be computed very quickly by simply looking up the amounts and costs of constituent ingredients and performing the appropriate summation. (This scenario is rather typical of the whole area of closed-loop optimization, where optimization algorithms direct and obtain results from real physical experiments, as described elsewhere (Box, 1957; Rechenberg, 2000; Caschera et al., 2010; Small et al., 2011), but, equally, objective functions may all be computational but still have widely differing latency (or delay).)

It is not obvious, we think, what a good strategy for the above scenario would be, given current multiobjective evolutionary algorithms (MOEAs) and the state of the field. Nor is it clear how much potential loss of performance comes about by using a basic MOEA going at the speed of the slowest objective (a default approach to the problem). These are the two main questions we seek to answer in the remainder of this paper. For simplicity, we focus mostly on the bi-objective case, although some of our definitions are general, and we include a discussion of extensions to $m > 2$ objectives in a later section.

2. Problem Definition

We define the problem more formally in the following. The notation we introduce here allows us to describe MOEAs that are suitable for evaluations that are not necessarily performed on all objectives at the same time. In the next section on algorithms we will present some such general schemes and indicate how they are instantiated on some basic, well-known MOEAs.

2.1. Basic Definitions

Definition 2.1 (Multiobjective Optimization Problem)**.** The general formulation of the problem is: "maximize" $\mathbf{f}(x)$ subject to $x \in X$, where x is an n-dimensional candidate solution vector, X is the search domain and $\mathbf{f} = (f_1, \ldots, f_m)$ is a vector objective function $\mathbf{f} : X \to \mathbb{R}^m$ mapping solutions to an m dimensional real-valued objective space. The term 'maximize' is written in quotes in order to indicate that there are not unique maxima to such a problem in general, and a further definition is needed to define an ordering on candidate solutions (see below).

Definition 2.2 (Pareto dominance)**.** Consider two solutions x_1 and x_2. We say that x_1 dominates x_2, also written as $x_1 \succ x_2$, if and only if $\exists i$ such that $f_i(x_1) > f_i(x_2)$ and $\forall j, f_j(x_1) \geq f_j(x_2)$.

Definition 2.3 (Pareto optimal front)**.** The Pareto optimal front, also denoted PF_{true}, is the set of points $\{\mathbf{f}(x) | x \in X, \nexists y \in X, y \succ x\}$.

Definition 2.4 (Approximation Sets and Performance)**.** Any set of points in the objective space with elements that are all non-dominated within the set is called an approximation set. Such sets can be partially ordered according to the *better* relation (Zitzler et al., 2003), analogously to the dominance order on points. The aim of multiobjective optimization can be defined as finding the best possible approximation set, where best is determined by this order. As a proxy method for assessing approximation sets, we use the hypervolume and attainment surfaces, as recommended by Zitzler et al. (2008).

2.2. Budgeted Optimization Definitions

Definition 2.5 (Total Budget)**.** The total budget for solving an optimization problem is the total number of time steps B available for solving it, under the assumption that only solution evaluations consume any time.

Definition 2.6 (Limited-Capacity Parallel Evaluation Model)**.** We assume parallelization of the evaluation of solutions is available, in two senses. First, a solution may (but need not) be evaluated on one objective in parallel to its being evaluated on another objective. Secondly, a number of (at most λ) solutions may be evaluated at the same time (i.e., as a batch or population) on any objective, provided their evaluation is started at the same time step, and finishes at the same timestep

(i.e. batches cannot be interrupted, added to, etc., during evaluation). For sake of simplicity, we assume λ is the same for all objectives.

Definition 2.7 (Per-Objective Latency)**.** Assume that each objective i can be evaluated in $k_i \in \mathbb{Z}^+$ timesteps (for a whole batch). Here, we consider a bi-objective case, and for simplicity, we define $k_1 = 1$ and $k_2 = k_{slow} > 1$, so that the slower objective is k_{slow} times slower than the faster one.[2]

Lemma 2.1 (Per-Objective Budgets)**.** From definitions 2.5, 2.6, 2.7, it follows that the total budget of evaluations *per objective* is different. The budget for f_1 is λB, whereas the budget for f_2 is $\lambda\lfloor B/k_{slow}\rfloor$.

Note, these per-objective budgets are derived and are the best possible, assuming that solutions are always evaluated in parallel batches of size λ, and new batches are evaluated immediately after the one just finished with no timestep unused (both objectives). Assuming a standard MOEA applied to the problem, by contrast, one would obtain only $\lambda\lfloor B/k_{slow}\rfloor$ evaluations on both objectives.

3. Algorithms

In this section, we will describe a number of algorithmic schemes that are able to operate in the model of budgeted multiobjective optimization defined above. One approach to the problem, as posed, would be to run a standard MOEA at the speed of the fast objective but to use fitness *approximation* (or "inheritance") whenever the slow objective function is "busy" with evaluating an earlier batch. Such an approach relies rather heavily on the fitness approximation scheme, and its performance will certainly depend on it closely. We briefly consider methods such as this one, which use approximation, in Section 5 (and these types of scheme were also the focus of our previous work (Allmendinger & Knowles, 2013a)). Our main original contribution in this paper, in contrast, is to propose and analyze a number of schemes that do not use approximation for the slow objective. We define the schemes in general terms, then provide four concrete strategies (there are two variants of Scheme 3) based on them. Finally, we indicate how these can be instantiated on existing generational MOEAs (and also explain, later in Section 6, how the schemes can be applied within steady-state MOEAs).

[2]In reality, the objectives of a problem may not have latencies that are exact multiples of each other, of course, and we will discuss this scenario in more detail in Section 6.

3.1. Algorithm Schemes

The basis of the proposed schemes is the observation that the slower objective is only evaluated every k_{slow} time steps. Then three distinct approaches can be identified for using the faster objective(s)[3]:

- Scheme I: To go at the rate of the slower objective (and go at the same rate on the faster objective, skipping time steps) using a standard MOEA. This approach uses the full budget of evaluations on the slow objective, but it does not fully utilize the evaluation budgets available for the faster objective.
- Scheme II: To go at the rate of the faster objective using a standard single-objective evolutionary algorithm (EA), for part of the optimization, and then switch to a final, evaluation of some selected solutions on the slower objective at some timestep t_{swap}. Until the time point t_{swap}, this approach fully exploits the budget of evaluations for the fast evaluation, but it does not utilize the budget of evaluations available for the slower objective.
- Scheme III: Interleaving the evaluation of the objectives so that both per-objective budgets are used to their full extent. Figure 2 illustrates this for the two-objective case.

Note that all three of these schemes result in a final population that has been evaluated on both objectives. It may seem intuitive that Scheme III is likely to be the most desirable, as more evaluations are done per unit time (and time is budgeted), but there remains the question of how to co-ordinate the additional evaluations done on the fast objective only, with a population of points that is evaluated on both objectives.

Four strategies based on the above schemes are investigated in this work: Waiting, Fast-First, Brood Interleaving, and Speculative Interleaving. In the following each strategy is explained in more detail.

In order to define concrete instantiations of the schemes, the following notation is introduced: The set G_i denotes the EA population at time step i. In contrast to this, the search trace S_i comprises the entire set of solutions generated by the EA up to time step i. The sets G_i^{fast} and G_i^{slow} denote the sets of solutions to be evaluated on the fast and slow objective at time step i,

[3]Note that the schemes are defined for two objectives only, but could also work for the case where $m > 2$ with just one slower objective, and the set of faster ones all evaluating at the same speed. For $m > 2$ and all objectives having different latencies (or delays), we have not designed a scheme yet.

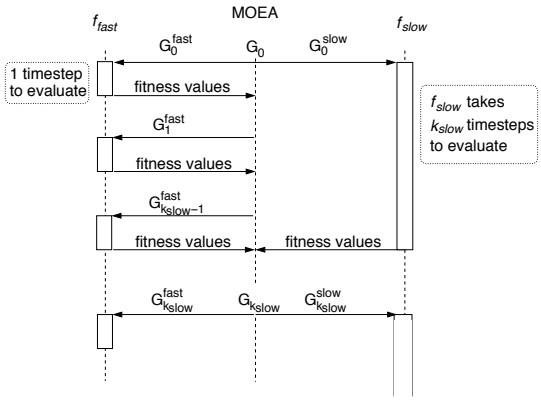

Figure 2: A sequence diagram of an MOEA using two objective functions with different latencies. (Time increases downwards in the diagram.)

respectively. Note that these can be distinct from each other and from the current offspring population — the need for this will become clear later in this section. The sets G'_i, G''_i, and G'''_i represent auxiliary sets of solutions at time step i, such as the offspring population or parents selected for reproduction, and K_i and K'_j the current and offspring interleaving population devoted for optimizing the fast objective function by Speculative Interleaving, respectively.

3.1.1. Waiting

The most straightforward strategy to deal with delayed objectives is an implementation of Scheme I, as shown in Algorithm 1. This strategy deals with delayed objectives by maintaining a single population (for both objectives) that goes at the rate of the slow objective on both objectives. In other words, evolution waits for all evaluations to be completed before continuing, thus losing evaluation time on the fast objective (Lines 4 and 11).

We call this strategy *Waiting*, and it can be easily embedded into a standard MOEA. This approach can be seen as a representative of the standard approach used in the MOEA literature, where possible differences between the delay of objectives have not been considered.

3.1.2. Fast-First

A simple strategy for the implementation of Scheme II is as follows (see Algorithm 2): The strategy *Fast-First* aims to maximize the number of fast evaluations, while ensuring that a final set of λ solutions has been evaluated on all objectives.

Algorithm 1 Scheme I: Waiting

1: initialize $G_0^{\text{fast}} := G_0^{\text{slow}} := G_0$
2: **in parallel, do** (a) **and** (b):
3: (a): evaluate G_0^{slow}
4: (b): evaluate G_0^{fast}
5: $i := k_{\text{slow}}$; $S_i := G_0$
6: **while** $i < B - k_{\text{slow}}$ **do**
7: apply multiobjective (parental) selection with variation to $G_{i-k_{\text{slow}}}$ to give G'_i
8: $G_i^{\text{fast}} := G_i^{\text{slow}} := G'_i$
9: **in parallel, do** (a) **and** (b):
10: (a): evaluate G_i^{slow}
11: (b): evaluate G_i^{fast}
12: $S_{i+k_{\text{slow}}} := S_i \cup G'_i$
13: apply environmental selection to $G_{i-k_{\text{slow}}}$ and G'_i to give G_i
14: $i := i + k_{\text{slow}}$; $S_i := S_{i-k_{\text{slow}}} \cup G'_{i-k_{\text{slow}}}$
15: identify and output the set of non-dominated solutions in S_i

This can be achieved by going, for most of the optimization duration, at the rate of the faster objective so as to optimize the faster objective only, and then by switching to the slower objective at the latest possible point (i.e. $t_{swap} = B - k_{slow}$, which allows for a final round of evaluations on the slow objective) (Line 5). The faster objective can be optimized using a standard single-objective EA. In the simplest case (considered here), the subsequent switch to the slower objective would involve the evaluation of a diverse set of solutions (here the best, distinct λ solutions found with respect to the fast objective) on the slower objective.

Again, it is clear that this strategy can be easily embedded into a standard single-objective EA. The main modification required is the maintenance of the search trace S_i required for the final filtering step (Line 12). The approach bears some resemblance to the previous use of diversity-preserving MOEAs in applications where additional objectives become available at the end of the optimization only (Chmielewski, 2013).

3.1.3. Interleaving strategies

The final two strategies, Brood and Speculative Interleaving, are implementations of Scheme III, and are less straightforward than the Waiting and the Fast-First strategies discussed above. Unlike Waiting and Fast-First, strategies in Scheme III attempt to utilize the full budget of evaluations available under both the fast and the slow objectives. The two strategies differ between each other in the way the results from fast and slow evaluations are integrated. Let us first present the general idea of strategies in Scheme III before explaining the strategies Brood and Speculative Interleaving in more detail.

The pseudocode given in Algorithm 3 presents a gen-

Algorithm 2 Scheme II: Fast-First

1: initialize $G_0^{\text{fast}} := G_0$
2: evaluate G_0^{fast}
3: update objective values in G_0 based on the values obtained for G_0^{fast} only
4: i:=1; $S_i := G_0$
5: **while** $i < B - k_{\text{slow}}$ **do**
6: apply single-objective (parental) selection with variation to G_{i-1} to give G'_i
7: $G_i^{\text{fast}} := G'_i$
8: evaluate G_i^{fast}
9: update objective values in G'_i based on the values obtained for G_i^{fast} only
10: apply environmental selection to G_{i-1} and G'_i to give G_i
11: i:=$i + 1$; $S_i := S_{i-1} \cup G'_{i-1}$
12: select the best, distinct λ solutions in S_i (wrt the fast objective) and assign these to G_i^{slow}
13: evaluate and output G_i^{slow}

Algorithm 3 Scheme III: A general template

1: // *Initialization:*
2: initialize a population G_0 of solutions and set $G_0^{\text{fast}} := G_0^{\text{slow}} := G_0$
3: **in parallel, do** (a) **and** (b):
4: (a): evaluate G_0^{slow}
5: (b): evaluate G_0^{fast}, followed by an *inner (single-objective) EA*: do $k_{\text{slow}} - 1$ generations of an EA initialized with G_0^{fast} on the fast objective
6: increment time counter i:=k_{slow}, and update S_i
7:
8: // *Main loop:*
9: **while** $i < B - k_{\text{slow}}$ **do**
10: apply multiobjective (parental) selection on $G_{i-k_{\text{slow}}}$ to produce new parent population G'_i
11:
12: // *Interleaving module:*
13: This module creates the two new populations, G_i^{slow} and G_i^{fast}, as follows:
14: G_i^{slow} is composed of solutions from the last inner EA run that improved on their parents G'_i on the fast objective plus, if insufficiently many, some new offspring of the parent population G'_i
15: G_i^{fast} is composed of offspring of G_i^{slow}
16:
17: **in parallel, do** (a) **and** (b):
18: (a): evaluate G_i^{slow}
19: (b): evaluate G_i^{fast}, followed by an *inner (single-objective) EA*: do $k_{\text{slow}} - 1$ generations of a EA initialized with G_i^{fast} on the fast objective
20: apply environmental selection to $G_{i-k_{\text{slow}}}$ and G_i^{slow} to give new population G_i
21: increment time counter i:=$i + k_{\text{slow}}$, and update S_i
22: identify and output the set of non-dominated solutions in S_i

eral Scheme III strategy, which we divided into several modules: After *initializing* the population G_0 and evaluating it in parallel on the slow and fast objective, (a) and (b), an *inner (single-objective) EA* is initialized with the current generation on the fast objective, G_0^{fast}, and applied to optimize the fast objective for the remainder of the interleaving period (i.e. $k_{\text{slow}} - 1$ generations). The *main loop* first creates a parent population based on multiobjective selection applied to the current population $G_{i-k_{\text{slow}}}$, and then enters the *Interleaving module* to create the new populations G_i^{slow} and G_i^{fast}. These populations are then evaluated in parallel, and an *inner (single-objective) EA* is used to optimize the fast objective for the remainder of the interleaving period. Finally, environmental selection is applied to the current population $G_{i-k_{\text{slow}}}$ and the current generation on the slow objective G_i^{slow} to give the new population G_i. The main loop is repeated until the budget B is used before non-dominated solutions on both objectives are identified and returned to the user.

Brood Interleaving: The idea of *Brood Interleaving* is to use the fast objective function to look ahead at possible offspring of the current generation being evaluated on the slow objective. This looking ahead can then be used to bias the next generation on the slow objective (hopefully in a positive way) so that the slow objective is used as efficiently as possible.

The way in which the look ahead is achieved is via something reminiscent of (soft) brood selection (Altenberg, 1994; Tackett & Carmi, 1994; Walters, 1998). In brood selection, more offspring are created by variation than will pass into the next generation, and this surplus (or brood) is then reduced by performing some filtering on the brood. In the case of *Brood Interleaving*, the fast objective is used to filter out prospective offspring of the generation currently under evaluation for the slow objective that perform worse than their parent(s) on the fast objective. This filtering should have the effect of avoiding the evaluation of offspring dominated by its parents. However, it also prevents the evaluation of those solutions that outperform their parents on the slow objective only, and it is unclear to what extent this biases the search.

Pseudocode for Brood Interleaving is given in Algorithm 4. A population of individuals, G_0, is initialized and this becomes the generation zero for both the slow and fast objectives. Evaluations are done in parallel (interleaved) on the slow and fast objectives (Lines 3 to 6). While the slow one is evaluating the zeroth generation, the fast one finishes, and subsequent generations for the fast objective are constructed and evaluated. These subsequent generations are offspring (i.e., genetic variants) of generation zero (Line 6).

When the slow objective function finishes, it is time to construct the next generation for the slow objective,

Algorithm 4 Scheme III(i): Brood Interleaving

1: initialize $G_0^{\text{fast}} := G_0^{\text{slow}} := G_0$
2: **in parallel, do** (a) **and** (b):
3: (a): evaluate G_0^{slow}
4: (b): evaluate G_0^{fast}
5: **for** $j = 1$ to $k_{\text{slow}} - 1$ **do**
6: apply uniform selection with variation to G_0^{slow} to generate G_j^{fast}, and evaluate G_j^{fast}
7: $i := k_{\text{slow}}$; $S_i := G_0 \cup G_1^{\text{fast}} \cup \ldots \cup G_{k_{\text{slow}}-1}^{\text{fast}}$
8: **while** $i < B - k_{\text{slow}}$ **do**
9: apply multiobjective (parental) selection without variation to $G_{i-k_{\text{slow}}}$ to give G_i'
10: find offspring of G_i' in $G_{i-k_{\text{slow}}+1}^{\text{fast}}, \ldots, G_{i-1}^{\text{fast}}$ and assign to G_i''
11: select from G_i'' those offspring that improved on one of their parents' (in G_i') evaluation on the fast objective function and assign to G_i'''
12: $G_i^{\text{fast}} := \emptyset$
13: **while** $|G_i'''| > \lambda$ **do**
14: remove a solution selected at random from G_i'''
15: **while** $|G_i'''| < \lambda$ **do**
16: create offspring o from uniform selection and variation applied to G_i'; $G_i''' := G_i''' \cup \{o\}$; $G_i^{\text{fast}} := G_i^{\text{fast}} \cup \{o\}$
17: $G_i^{\text{slow}} := G_i'''$
18: **while** $|G_i^{\text{fast}}| < \lambda$ **do**
19: create offspring o from uniform selection and variation applied to G_i^{slow}; $G_i^{\text{fast}} := G_i^{\text{fast}} \cup \{o\}$
20: **in parallel, do** (a) **and** (b):
21: (a): evaluate G_i^{slow}
22: (b): evaluate G_i^{fast}
23: **for** $j = 1$ to $k_{\text{slow}} - 1$ **do**
24: apply uniform selection with variation to G_i^{slow} to generate G_{i+j}^{fast}, and evaluate G_{i+j}^{fast}
25: apply environmental selection to $G_{i-k_{\text{slow}}}$ and G_i^{slow} to give G_i
26: $i := i + k_{\text{slow}}$; $S_i := S_{i-k_{\text{slow}}} \cup G_{i-k_{\text{slow}}}''' \cup G_{i-k_{\text{slow}}+1}^{\text{fast}} \cup \ldots \cup G_{i-1}^{\text{fast}}$
27: identify and output the set of non-dominated solutions in S_i

G_i^{slow}, upon performing multiobjective parental selection on the current EA population $G_{i-k_{\text{slow}}}$ to give (the parent population) G_i' (Line 9). The next generation on the slow objective, G_i^{slow}, is then made up (partially) of those offspring of G_i' (already evaluated on the fast objective) whose parents (one or both of them) improved over at least one of their parents' fast objective function evaluations (Line 10 and 11).

At this point the next generation can be smaller or larger than the batch size λ (e.g. if only few or many offspring improved upon their parents, respectively). In the latter case, solutions are removed at random from the generation (Line 14). In the former case, the generation is filled with solutions resulting from uniform selection from G_i' and subsequently applying variation (Line 16); note that these solutions need to be evaluated on both the fast and slow objective, occupying some of the spots available in the next generation for the fast objective, G_i^{fast}. The remaining spots for evaluation under G_i^{fast} are filled with solutions resulting from uniform selection from G_i^{slow} and subsequently applying variation (Line 19). After evaluating the generations for the slow and fast objective, and the interleaving populations (Lines 21 to 24), environmental selection is applied to the current EA population $G_{i-k_{\text{slow}}}$ and the set of solutions evaluated on the slow objective, G_i^{slow}, to give the EA population of the next iteration, G_i (Line 25).

Speculative Interleaving: The strategy of *Speculative Interleaving* (see Algorithm 5) is similar to Brood Interleaving except that it aims to maintain selection pressure at all time steps. Prior to the return of slow evaluations, only partial information about solutions is available. This means that a single-objective selection scheme needs to be used during the wait for slow evaluations, which will only account for solution evaluations under the fast objective. This single-objective scheme is embedded into a single-objective EA that has the task to drive the evolution of the generations on the fast objective. It is clear that this may introduce a bias towards the optimization of the fast objective, yet it is unclear to what extent this will outweigh the advantages gained from increased selection pressure.

Increased selection pressure is achieved by creating the interleaving populations, G_{i+j}^{fast}, $j = 1, \ldots, k_{\text{slow}} - 1$, by applying a single-objective EA (i.e. single-objective selection, variation, and potentially elitism) to K_j, $j = 1, \ldots, k_{\text{slow}} - 1$ (Lines 6 to 8, and Lines 26 to 28). Since the interleaving populations evolve whilst the slow objective is evaluated, we need to record all the solutions or ancestors (in G_i^{slow}) used to create a solution in G_{i+j}^{fast}, $j = 1, \ldots, k_{\text{slow}} - 1$; note, assuming that a solution is created from two parents, then the number of distinct ancestors associated with a solution may vary between 2 and $2^{k_{\text{slow}}-1}$. Subsequently, an offspring is included into G_i^{slow} if its ancestors (one or multiple) are selected by multiobjective parental selection (Lines 11 and 12) and it outperforms at least one of them with respect to the fast objective (Line 13).

This strategy resembles principles of speculative parallelization used in some parallel simulated annealing methods (Marchesi et al., 1994).

3.2. Instantiations on Generational-Based MOEAs

When augmenting the algorithm schemes introduced in the previous section on a generational-based MOEA, then the environmental and multiobjective parental selection scheme, and the variation mechanism (crossover and/or mutation), need to be replaced with the ones used

Algorithm 5 Scheme III(ii): Speculative Interleaving

1: initialize $G_0^{\text{fast}} := G_0^{\text{slow}} := G_0$
2: **in parallel, do** (a) **and** (b):
3: (a): evaluate G_0^{slow}
4: (b): evaluate G_0^{fast}
5: $K_1 := G_0^{\text{fast}}$
6: **for** $j = 1$ to $k_{\text{slow}} - 1$ **do**
7: apply single-objective (parental) selection with variation to K_j to generate G_j^{fast}, and evaluate G_j^{fast}
8: apply environmental selection to K_j and G_j^{fast} to give K_{j+1}
9: $i := k_{\text{slow}}$; $S_i := G_0 \cup G_1^{\text{fast}} \cup \ldots \cup G_{k_{\text{slow}}-1}^{\text{fast}}$
10: **while** $i < B - k_{\text{slow}}$ **do**
11: apply multiobjective (parental) selection without variation to $G_{i-k_{\text{slow}}}$ to give G'_i
12: find offspring of G'_i in $G_{i-k_{\text{slow}}+1}^{\text{fast}}, \ldots, G_{i-1}^{\text{fast}}$ and assign to G''_i
13: select from G''_i those offspring that improved on one of their parents' (in G'_i) evaluation on the fast objective function and assign to G'''_i
14: $G_i^{\text{fast}} := \emptyset$
15: **while** $|G'''_i| > \lambda$ **do**
16: remove a solution selected at random from G'''_i
17: **while** $|G'''_i| < \lambda$ **do**
18: create offspring o from uniform selection and variation applied to G'_i; $G'''_i := G'''_i \cup \{o\}$; $G_i^{\text{fast}} := G_i^{\text{fast}} \cup \{o\}$
19: $G_i^{\text{slow}} := G'''_i$
20: **while** $|G_i^{\text{fast}}| < \lambda$ **do**
21: create offspring o from uniform selection and variation applied to G_i^{slow}; $G_i^{\text{fast}} := G_i^{\text{fast}} \cup \{o\}$
22: **in parallel, do** (a) **and** (b):
23: (a): evaluate G_i^{slow}
24: (b): evaluate G_i^{fast}
25: $K_1 := G_0^{\text{fast}}$
26: **for** $j = 1$ to $k_{\text{slow}} - 1$ **do**
27: apply single-objective (parental) selection with variation to K_j to generate G_j^{fast}, and evaluate G_j^{fast}
28: apply environmental selection to K_j and G_j^{fast} to give K_{j+1}
29: apply environmental selection to $G_{i-k_{\text{slow}}}$ and G_i^{slow} to give G_i
30: $i := i + k_{\text{slow}}$; $S_i := S_{i-k_{\text{slow}}} \cup G'''_{i-k_{\text{slow}}} \cup G_{i-k_{\text{slow}}+1}^{\text{fast}} \cup \ldots \cup G_{i-1}^{\text{fast}}$
31: identify and output the set of non-dominated solutions in S_i

by the MOEA selected (e.g. in Lines 7 and 13 in Algorithm 1, and Lines 6, 9, 16 and 19 in Algorithm 4). For Fast-First (Line 6 in Algorithm 2) and the process of creating interleaving populations within Speculative Interleaving (Lines 6 to 8, and Lines 26 to 28, in Algorithm 5), we would replace these operators with the ones used by the single-objective EA selected.

4. Empirical Study

To understand the relative performance of the above strategies, we conduct an empirical study on a variety of problem instances with delayed objectives. In particular, we focus on the analysis of the following aspects:

1. The optimization performance of Waiting, Fast-First and Interleaving strategies compared to the optimization performance obtained in an undelayed environment, in general terms.
2. The relative optimization performance of the more advanced methods of Fast-First and the two Interleaving strategies compared to the optimization performance of a standard (Waiting) strategy, in general terms.
3. The effect of problem type on the relative optimization performance of the strategies. Specifically, we expect this to be influenced by the correlation between objectives, the length of the delay, and the ruggedness of the fitness landscape being optimized.
4. Time budget effects: the relative optimization performance of the different strategies as a function of total optimization time.

4.1. Theoretical considerations

Given the design of the algorithms, we can set out the following rough expectations regarding their individual and relative performance:

1. In most scenarios, the performance of an MOEA in the absence of delays would be expected to provide us with an estimated "upper bound" on the optimization performance of any of the alternative strategies (embedded within the same MOEA). This is because a delay has the effect of reducing the per-objective budgets available within these strategies. Any further increase in the length of the delay has the effect of reducing these budgets further. Therefore, any performance gap between the three strategies and this "upper bound" may be expected to increase as a function of the length of the delay.
2. Similarly, the Waiting strategy can be seen to provide an estimated "lower bound" on the optimization performance of any *sensible* strategy that we could design to account for the presence of delays. This is because Waiting is equivalent to a scenario with the same delay length on both objectives. As a consequence, the per-objective budget for the fast objective is not fully used, and the gap between the available and the used evaluations increases as a function of the length of the delay. Clearly, an improved strategy should attempt to fully utilize unused portions of the per-objective budgets. Assuming that the size of the evaluation budget has

a tangible effect on optimization performance (increasing monotonically in the number of evaluations), any performance advantage of such a strategy (compared to Waiting) should then be expected to increase with the length of the delay.
It is interesting to note that the overall number of evaluations (summing across all objectives) performed by the Fast-First strategy may indeed be smaller than the overall number of evaluations used by Waiting (e.g. if $1 < k_{\text{slow}} \leq 2$), as Fast-First effectively abandons large parts of the per-objective budget for the slow objective. In terms of maintaining a 'minimum number of overall evaluations' (as defined by the Waiting strategy), a Fast-First strategy is, arguably, not particularly sensible for small delays (roughly $k_{slow} \leq 2$, unless objectives are highly correlated).

3. The Fast-First strategy performs a "focused" (single-objective) optimization on the fast objective, followed by a final round of evaluations on the slow objective. This makes optimal use of the per-objective budget for the fast objective. On the other hand, the per-objective budget for the delayed objective is not used until the very end of the optimization. Hence, such a strategy could be considered to be optimal in the presence of a perfect correlation between objectives (in which case the evaluation of the slower objective would result in the duplication of information at all times). The extent to which Fast-First can cope with objectives that are less highly correlated is doubtful, but may be affected somewhat by the mechanisms of diversity maintenance within the underlying EA.
4. In contrast to Fast-First, both Speculative Interleaving and Brood Interleaving make full use of the per-objective budgets for each objective. In order to do so, the time lag until the return of the delayed objectives needs to be used to partially evaluate an additional $\lambda(k_{slow} - 1)$ solutions. The two schemes differ in the way these additional solutions are obtained. Brood Interleaving simply generates $\lambda(k_{slow} - 1)$ offspring, which are partially evaluated on the fast objective. Once results from the delayed objectives are returned, the results obtained on the fast objective are employed to inform selection of the next population. In contrast to this, Speculative Interleaving uses a greedy strategy that corresponds to a temporary switch to a single-objective EA (optimizing several generations on the fast objective only). The difference to Fast-First is that these episodes of single-objective optimization are then interleaved within generations of the multiobjective EA (once the delayed evaluations return).
Out of the two Interleaving strategies, Brood Interleaving may be expected to be less affected by the presence/absence of correlation between objectives, and the ruggedness of the fitness landscape, as its bias towards the fast objective is less pronounced (a preference for the fast objective is only introduced during the filtering step every k_{slow} time steps). On the other hand, the reduction of selection pressure (no selection pressure is applied during waits for the delayed objective) will likely hinder convergence, especially for larger delays.
In contrast, Speculative Interleaving maintains selection pressure during all generations, which may help in driving the search — promoting quick convergence to local optima on rugged fitness landscapes. On the other hand, selection pressure may be too biased towards the fast objective, particularly so for large delays (which will increase the ratio of fast compared to slow evaluations). This effect will be undesired for problems with little (or anti-)correlation between objectives.

4.2. Experimental setup

The following subsection describes in detail the binary test functions (all functions are to be maximized), the parameter settings, the MOEAs and the evaluation measures used to provide empirical investigation of the performance differences.

Test functions: The first test function, a *mapped bi-objective OneMax problem*, which is inspired by the generalized OneMax problem (Droste et al., 2006), is a family of functions that allows us to control the correlation between objectives. Assuming $n_1(x)$ to be the number of 1s in a candidate solution vector x, $n_1(y)$ the number of 1s in y, where y is a mapped version of solution vector x, then the mapped bi-objective OneMax problem can be defined as

$$\mathbf{f} = (f_1, f_2) = (n_1(x), n_1(y)),$$
$$\text{where } y_i = (x_i + map_i) \text{mod} 2,\ i = 1, \ldots, n.$$

The mapped value of a decision variable is $map_i \in \{0, 1\}$ and is set independently for each $i = 1, \ldots, n$ by flipping a coin biased by the degree of correlation $corr \in [-1, 1]$ desired. For instance, for a problem with no correlation between the objectives ($corr = 0$), the probability of $map_i = 1$ or 0 is 0.5. For a maximal positive correlation ($corr = 1$), we set $map_i = 0, i = 1, \ldots, n$, whilst for a maximal negative correlation ($corr = -1$) we set $map_i = 1, i = 1, \ldots, n$. For an intermediate correlation of c, we set a map bit to zero with a probability of $(1.0 + c)/2$.

The second test function used in this study is the *leading ones trailing zeros* (LOTZ) function (Laumanns et al., 2004), which can be defined as

$$\mathbf{f} = (f_1, f_2) = \left(\sum_{i=1}^{n} \prod_{j=1}^{i} x_j \ , \ \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j) \right).$$

The Pareto front of this problem consists of solutions of the form 1^a0^b with $a + b = n$. The LOTZ problem has often been used to investigate theoretical properties of MOEA algorithms, such as running times (Laumanns et al., 2004). Since the problem is well-understood it should also aid the process of understanding the impact of delayed objectives on performance.

The third test function we use is the family of multi-objective *NK* landscapes, or *MNK* landscapes (Aguirre & Tanaka, 2007). *MNK* landscapes extend Kauffman's *NK* model (Kauffman, 1993) to multiple objectives by associating a different *NK* landscape instance to each objective. An *NK* landscape instance can be used to model epistatic interactions between bits so as to control the ruggedness (number and density of local optima) of the fitness landscape being optimized. More formally, an *MNK* landscape with m objectives (we fix $m = 2$ in this work) can be defined as

$$\mathbf{f} = (f_1, \ldots, f_m),$$
$$\text{where } f_i = \frac{1}{N} \sum_{j=1}^{N} g_j(x_j, z_1^{(j)}, z_2^{(j)}, \ldots, z_K^{(j)}), \ i = 1, \ldots, m.$$

In this equation, N defines the number of bits (in our notation this variable is denoted as n), and the function g_j the fitness contribution of x_j and the K bits (also called neighbors), $z_1^{(j)}, z_2^{(j)}, \ldots, z_K^{(j)}$, interacting with bit x_j. Typically, the K neighbors are selected at random, and the fitness contributions g_j initialized uniformly in the range $[0; 1)$ (i.e. $f_i \in [0; 1)$). These settings are set independently for each objective function f_i. The value of K can vary between the objective functions to tune the ruggedness of each objective function separately (with larger values of K resulting more rugged landscapes), whilst, of course, the number of bits N remains constant. For the sake of gaining a better understanding of the effect of delayed objectives on performance, in this work we focus on the simple case where the value of K is identical for each objective function (but the K neighbors and the fitness contributions g_j are set anew and at random for each objective function). Unlike the mapped OneMax and LOTZ problem, *MNK* landscapes will allow us to investigate the performance of the different strategies as a function of the ruggedness of the landscape optimized.[4]

Table 1: EA parameter settings.

Parameter	*Setting*
Parent population size μ	50
Offspring population size λ	50
Per-variable mutation probability p_m	$1/n$
Crossover probability p_c	0.6
Fitness scaling factor, κ	0.05
#Time steps (generations) B	40

Search algorithms: To test the delay-handling strategies described in Section 3 we augment them onto the indicator-based EA (IBEA) (Zitzler & Künzli, 2004), a generational MOEA maximizing (in this case) the hypervolume indicator (Zitzler, 1999).

The algorithm uses binary tournament selection (with replacement) for parental selection, uniform crossover (Syswerda, 1989), bit flip mutation, and does not check whether a solution has been evaluated previously, i.e. identical solutions may be evaluated multiple times. The parameter settings of the MOEA are given in Table 1. For Fast-First, and the process of generating interleaving populations within Speculative Interleaving, a single-objective EA with the same parental selection and variation operators is employed (note, in this case tournament selection is based on a single objective only); environmental selection is done using a $(\mu + \lambda)$-ES (evolution strategy) reproduction scheme. Note, for IBEA we are using the adaptive version, involving scaling the hypervolume indicator values in combination with a fitness scale factor of $\kappa = 0.05$ (as recommended by Zitzler & Künzli (2004)).[5]

If not otherwise stated we use a budget of $B = 40$, a search space of size $n = 20$, and assume that objective f_2 is the slow objective. For the mapped OneMax problem and *MNK* landscapes, we create a new problem instance at random for each algorithmic run assum-

[4] Recently, *MNK* landscapes have been extended to control the correlation between objectives (Verel et al., 2011). We do not consider this problem here but investigate the impact of correlations between objectives using the mapped OneMax problem, which is more straightforward to analyze.

[5] All strategies have been coded up in Java within the jMetal framework (Durillo & Nebro, 2011). The code to run the strategies within IBEA and other generational MOEAs, such as NSGA-II (Deb et al., 2002), as well as steady-state MOEAs, such as SEMO (Laumanns et al., 2004) and SMS-EMOA (Beume et al., 2007), is available at `http://www.ucl.ac.uk/~ucberal/`.

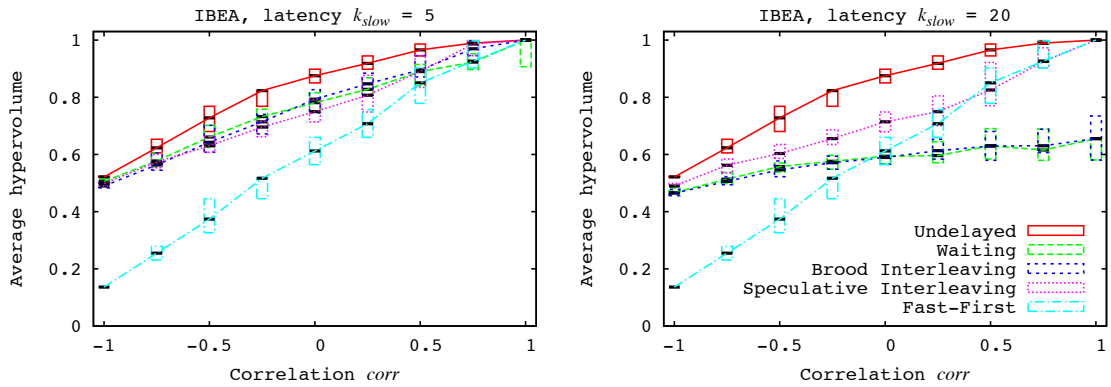

Figure 3: Median and interquartile ranges of the hypervolume achieved on the mapped OneMax problem by different strategies embedded within IBEA after 40 generations using a population size of 50. The correlation between the two objectives was varied from -1 to 1, in steps of 0.25. Results are shown for a latency of 5 (left) and 20 (right) generations on the slow objective (here f_2). For every setting marked by a point in the line graphs, a Friedman test (significance level of 5%) has been carried out. In the left plot, Waiting performs best out of the four strategies for $corr = -0.5$. In the right plot, Speculative Interleaving performs best out of the four strategies in the range $-1 < corr < 0.5$. There is no clear winner for the other settings.

ing a fixed level of correlation (the problem instances are the same for each strategy investigated). Any results shown are average results across 30 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.

Evaluation measures: We use two approaches to assess the performance of a multiobjective optimizer subject to delayed objectives. The first is based on the *hypervolume indicator* (Zitzler, 1999; Zitzler et al., 2003). This indicator assesses the size of the bounded region (in the objective space) dominated by a set of points (vectors of objective function values of non-dominated solutions). The region dominated is bounded on one side by a set of non-dominated solutions, and on the other side by a bounding point. If the bounding point is set appropriately and kept the same when comparing multiple sets of non-dominated solutions, then larger indicator values indicate that a solution set is better than another one in terms of various desirable aspects including diversity, extent and proximity to the Pareto optimal front. As, for the problems considered here, the value ranges of the objectives are known, the bounding point is set to the worst possible point in the objective space shifted by 1 in each objective. To compare the hypervolume obtained by different algorithms, we run each algorithm multiple times on a problem and report the median and interquartile ranges of the hypervolume across the algorithmic runs. To investigate significant differences between algorithms with respect to the hypervolume, we use a repeated-measures statistical test, the Friedman test (Friedman, 1937). This test does not assume an underlying distribution of the data (i.e. is non-parametric).

The second approach is based on *attainment surfaces* (Fonseca & Fleming, 1996). Attainment surfaces express graphically the performance of an MOEA in terms of the surface in the objective space that can be attained in a fraction of algorithmic runs. For each strategy, we show the surfaces that can be attained in 50% of the runs, also known as median attainment surface.

4.3. Experimental Results

In the following, we explore the empirical results on the mapped OneMax, LOTZ, and *MNK* landscapes problem, and discuss the extent to which they meet the expectations from our theoretical considerations above.

4.3.1. Correlation between objectives

The empirical results confirm some of the key expectations regarding the sensitivity of different strategies to the correlation between objectives. Results are given in Figure 3, which analyzes optimization performance on the mapped OneMax problem. The graphs show the average hypervolume obtained by the strategies as a function of the correlation between objectives. For the LOTZ problem and *MNK* landscapes, the correlation between objectives cannot be controlled so no

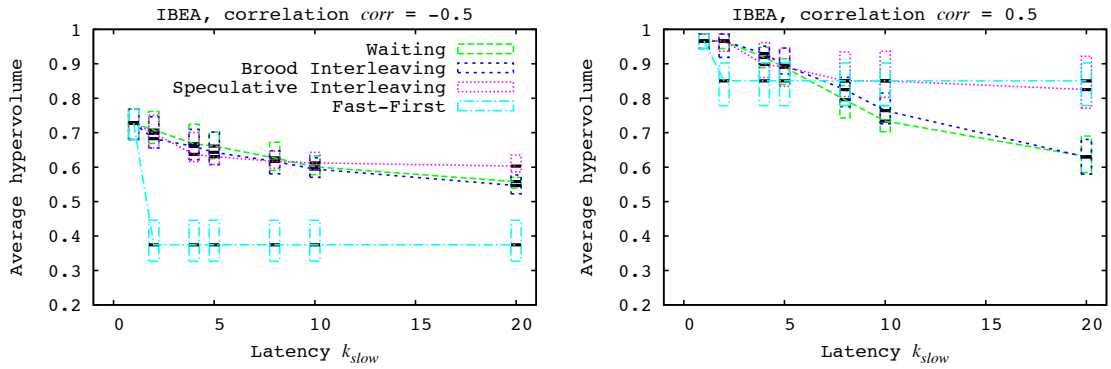

Figure 4: Median and interquartile ranges of the hypervolume achieved on the mapped OneMax problem by different strategies embedded within IBEA after 40 generations using a population size of 50. Results are shown for a correlation of -0.5 (left) and 0.5 (right) between the two objectives, for different latencies. For each setting shown on the abscissa, a Friedman test (significance level of 5%) has been carried out. In the left plot, Speculative Interleaving performs best out of the four strategies for $k_{slow} = 20$. There is no clear winner for the other settings.

results are shown. The following key observations can be made from these figures:

- For problem instances with a negative correlation between objectives, Waiting can be considered an appropriate strategy. In fact, in the presence of negative correlations and only short delays, Waiting emerges as the preferred strategy (see the results obtained for $k_{slow} = 5$ and correlations in the range $[-1, 0]$). This can be explained by the fact that all three alternative strategies bias the search towards the fast objective — albeit to varying degrees. It appears that this bias is sufficiently strong to be detrimental to optimization performance (when there is anti-correlation between objectives) and is not offset by the increased number of evaluations afforded by these strategies.

- For problem instances with a positive correlation between objectives, Fast-First, Brood Interleaving and Speculative Interleaving tend to perform as well or better than the Waiting strategy. As expected, the strategies start to gain an advantage as the correlation between objectives increases. Overall, Speculative Interleaving emerges as the most robust performer. The performance of Brood Interleaving is more varied and appears influenced by the length of the delay (explored in more detail in the following section). The performance of Fast-First in all settings is disappointing and becomes competitive for very high levels of correlation only.

4.3.2. Length of the delay

We further investigate how the different strategies are affected by increases in the delay period. Figure 4 shows the average hypervolume obtained by the strategies as a function of the delay in the slow objective. Results are shown for the mapped OneMax problem for the case of correlated and anti-correlated objectives. Equivalent results for the LOTZ problem and *MNK* landscapes are included in the Appendix (see Figure A.1). The following key observations can be made from these figures:

- For small delays of the slow objective, Waiting is the preferred strategy. This is the case independent of a positive or negative correlation between objectives.

- For larger delays, the interleaving strategies start to become competitive, particularly so in the case of positive correlations. In the case of positive correlations, Brood Interleaving emerges as the preferred strategy for medium-length delays, with Speculative Interleaving becoming the most competitive for long delays. This confirms our prior expectation that a lack of selection pressure may cause problems for Brood Interleaving in combination with long delay periods.

- The performance of Fast-First is generally poor, but almost unaffected by the delay length, which could potentially make it a reasonable strategy in the case of very long delays.

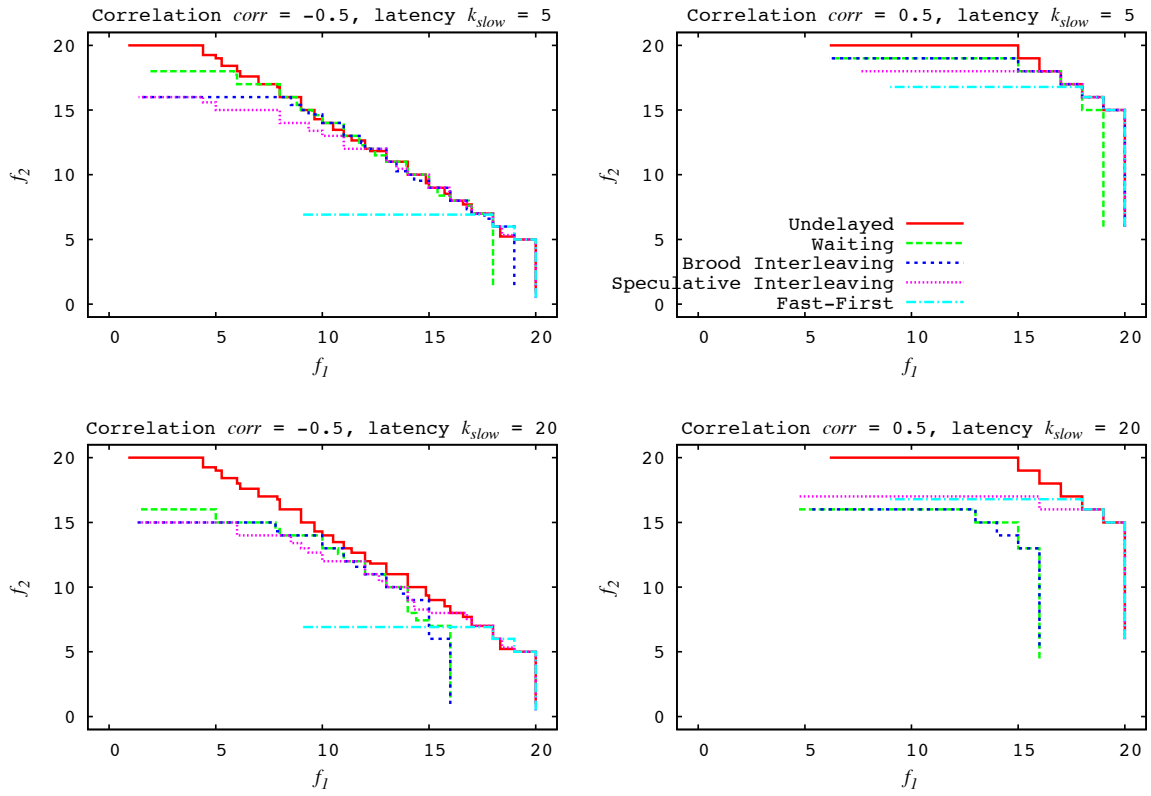

Figure 5: Median attainment surface on the mapped OneMax problem obtained by different strategies embedded within IBEA after 40 generations using a population size of 50. Results are shown for a correlation of -0.5 (left) and 0.5 (right) between the two objectives, and a latency of 5 (top) and 20 (bottom) generations on the slow objective (here f_2).

We further use median attainment surfaces to provide a more detailed investigation of the convergence behavior of the different strategies. Figure 5 shows results obtained by the different strategies for delay lengths of $k_{slow} = 5$ (top plots) and $k_{slow} = 20$ (bottom plots) time steps on a mapped OneMax problem with a correlation of 0.5 (right plots) and -0.5 (left plots). Equivalent results on the LOTZ problem and *MNK* landscapes are included in the Appendix (see Figures A.2 and A.3).

The figures confirm the intuition that, compared to a search in an undelayed environment, all strategies would be affected more severely for longer delay periods. Looking at the strategies in turn, it can be seen from Figure 5 that generally Waiting is able to maintain the most diverse set of non-dominated solutions among the strategies as it is not subject to any search bias arising from the preferential evaluation of the fast objective. As observed above, an exception is apparent for problems combining a positive correlation between objectives with long delays (bottom right plot), where strategies with a high selection pressure, such as Fast-First and Speculative Interleaving, do well. There is evidence of this expected bias for all other three strategies (see plots on the left-hand side with a correlation of $corr = -0.5$), with a trade-off between the optimization of the non-delayed objective (i.e. identifying fit solutions on the bottom right part of the Pareto front) and a uniform approximation of the Pareto front. As expected, Fast-First has the most pronounced bias and it performs best with respect to the fast objective (detecting reliably the optimal solution with respect to f_2), although almost no other parts of the Pareto front are found. Brood Interleaving shows the least bias towards the fast objective, but the results do indicate a lack of convergence towards the Pareto front. This effect is more pronounced for large delays, and is a direct consequence of the reduced

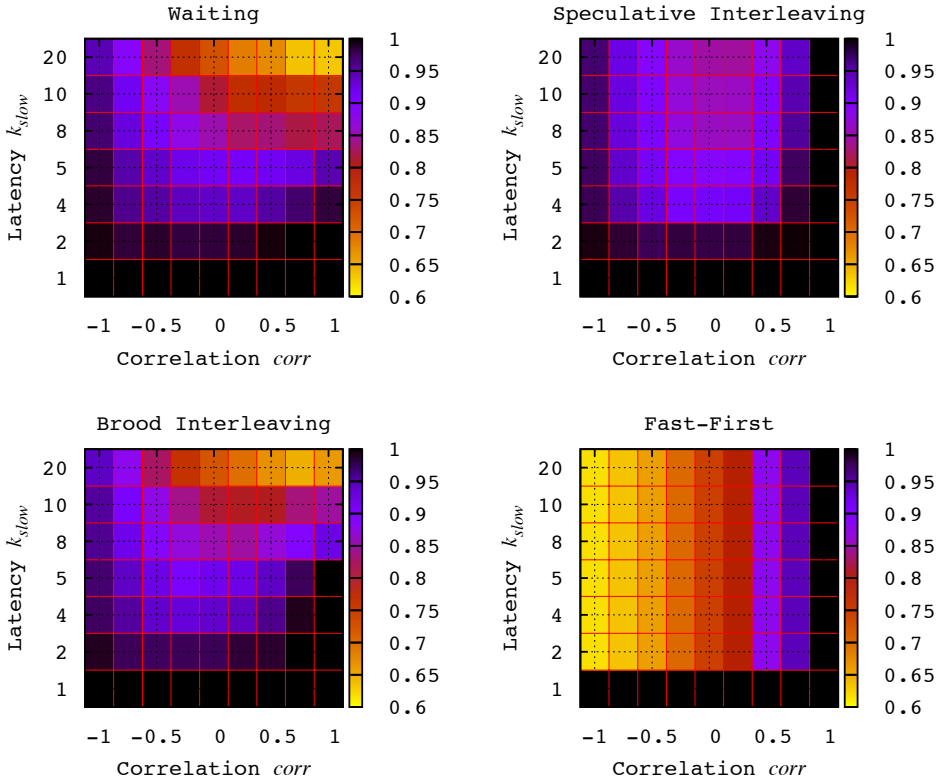

Figure 6: Proportion of attainable hypervolume (as derived in an undelayed environment) achieved by a strategy when embedded within IBEA on the mapped OneMax problem after 40 generations using a population size of 50. The color coding shows the performance ratio associated with each strategy as a function of the latency (y-axis) and the correlation between the objectives (x-axis).

selection pressure. The bias of Speculative Interleaving is more pronounced than that of Brood Interleaving, but it does appear to strike the best balance between bias and convergence over all.

The strategies behave symmetrically when the delay is put on objective f_1 instead of f_2.

4.3.3. Combined view

In order to provide a more comprehensive picture of the interaction of correlation and delay length for different search strategies, Figure 6 shows the ratio of the average hypervolume achieved between each of the strategies and an undelayed search as a function of the correlation between objectives *corr* and of the delay length k_{slow}. For the LOTZ problem and *MNK* landscapes, the correlation between objectives cannot be controlled so no results are shown.

From Figure 6 it can clearly be seen that a highly negative correlation between objectives and/or a short delay does not affect the performance significantly, except for Fast-First, when compared with the performance achieved in an undelayed environment. Comparing the strategies against each other it is apparent that Waiting and Brood Interleaving perform similarly for almost all correlation and delay values. A slight performance advantage of Brood Interleaving (compared to Waiting) is visible for positive correlations and medium to long delays. Overall, for this setting, Speculative Interleaving and Fast-First perform best. However, comparing Speculative Interleaving (top right plot) with Fast-First (bottom right plot), it can be seen that the performance of Fast-First degrades significantly as the correlation between objectives reduces. Speculative Interleaving is only inferior to Brood Interleaving for problems with little or no correlation between objectives and a delay of medium length. Overall, Speculative Interleaving emerges as the most promising strategy.

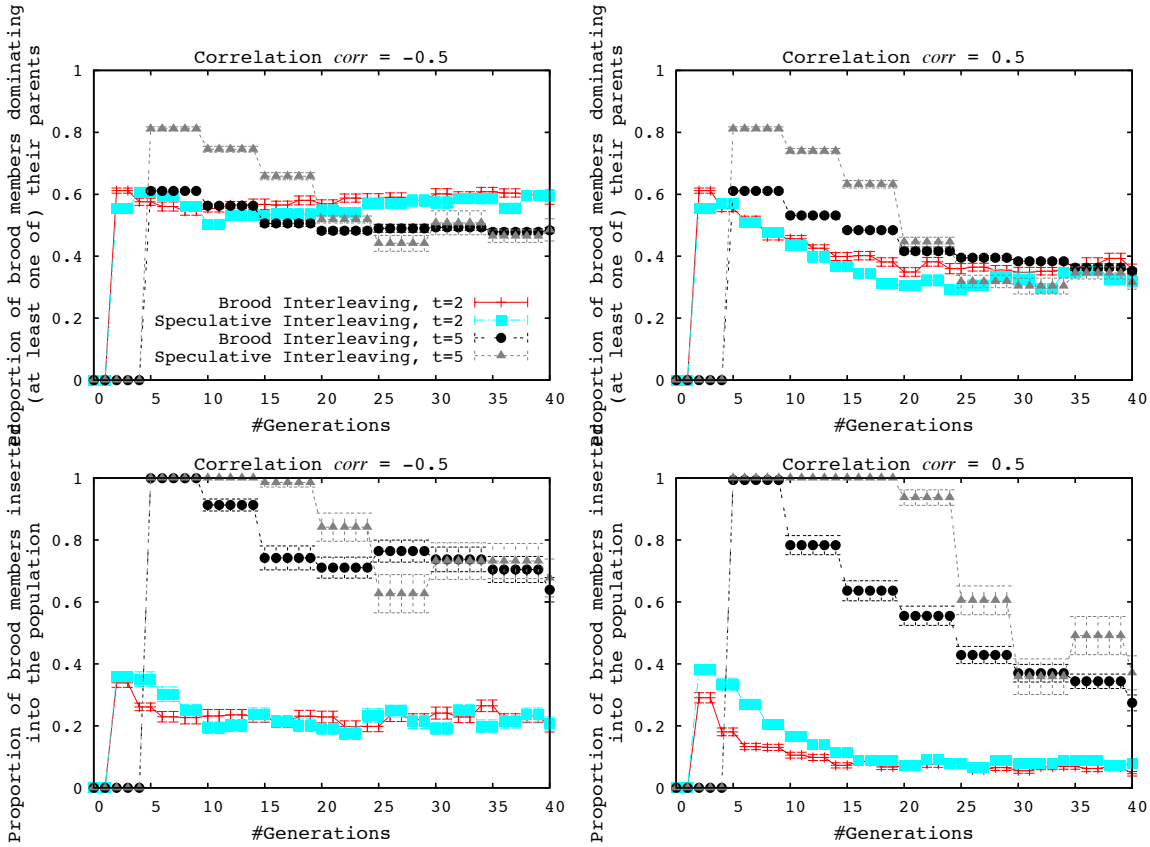

Figure 7: Proportion of brood members dominating (at least one of) their parents in terms of the fast objective (top) and proportion of brood members inserted into the population (bottom) as a function of the generation counter obtained by Brood and Speculative Interleaving on the mapped OneMax problem. Left plots show the two metrics for a correlation of -0.5, and right plots for 0.5 between the two objectives.

4.3.4. Analysis of the interleaving strategies

To understand the population dynamics of Brood and Speculative Interleaving, we investigate in more detail the impact that the brood has on offspring generation. Figure 7 gives insights into the contribution of partially evaluated solutions for both Brood and Speculative Interleaving (on the mapped OneMax problem). Unsurprisingly, increasing the delay leads to a higher probability that brood members dominate at least one of their ancestors (top plots) and thus are inserted into the population for evaluation on the slow objective (bottom plots). Due to the convergence of the population, both probabilities reduce as the optimization progresses, though the reduction is smaller for shorter delays. Due to the guided optimization of the fast objective, Speculative Interleaving is able to create fitter brood members than Brood Interleaving that are also more likely to be inserted into the population. This is particularly true at the beginning of the search when the population has not converged yet.

4.3.5. Impact of problem size

In this section we consider the impact of problem size on the performance of the different strategies. The results obtained for the LOTZ problem are shown in Figure 8. Equivalent results for the mapped OneMax problem and *MNK* landscapes are included in the Appendix (see Figures A.4 and A.5). The results show an increasing performance gap between the strategies and an undelayed search for larger problem sizes. In particular, the performance of Brood Interleaving quickly deteriorates for increasing search space sizes, and those strategies with higher level of selection pressure (Fast-First and Speculative Interleaving) show a higher level of robustness.

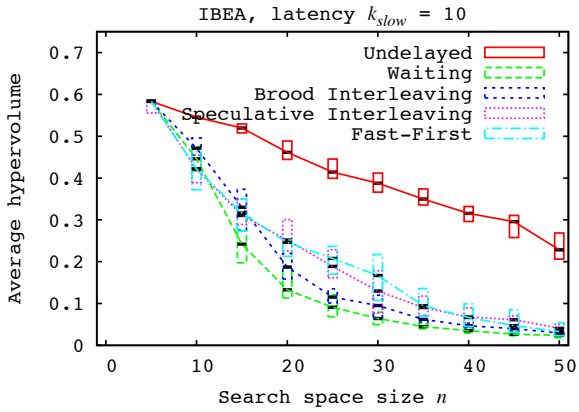

Figure 8: Median and interquartile ranges of the hypervolume achieved on the LOTZ problem by different strategies embedded within IBEA after 40 generations using population size of 50. Results are shown for a latency of 10 generations on the slow objective (here f_2) as a function of problem size, which is varied from 5 bits to 50, in steps of 5. For every setting marked by a point in the line graphs, a Friedman test (significance level of 5%) has been carried out. Brood Interleaving performs best out of the four strategies for $n = 10$, and Fast-First for $n = 30$. There is no clear winner for the other settings.

4.3.6. Impact of fitness landscape ruggedness

Finally, we investigate the impact of fitness landscape ruggedness, as can be be controlled within *MNK* landscapes, on the different strategies. Figure 9 shows the ratio of the average hypervolume achieved between each of the strategies and an undelayed search as a function of the delay and number of neighbors, K, each bit is interacting with. From the figure it can be seen that changing the ruggedness of the landscape, controlled here by K, impacts performance. For Fast-First (bottom right plot) it can be seen that increasing the ruggedness of the landscape tends to reduce the performance gap to an undelayed search. For the other strategies, a different pattern is observed: the performance tends to degrade with increasing K until a value of around $K = 5$, and then again improves for larger values of K. As observed previously, the strategies, Waiting and Brood Interleaving, behave similarly. Overall, Speculative Interleaving performs most robustly for varying values of K.

5. Related Work

Our work on handling delayed objectives has been informed and inspired by a number of other studies in the literature. In the following we briefly consider some of these parallels, and provide some further context and justification for the methods investigated herein.

We may begin by noticing that certain types of constraint handling in the EA literature operate in a similar fashion to the methods of ours that employ interleaving. In the constraint dominance method (Deb, 2000), for example, the idea of checking feasibility before investing in fitness function evaluation of a solution is used, partly, to improve efficiency of search. (This has proved a valuable idea in constrained EAs, including multiobjective EAs.) Although our method of Brood Interleaving derives more directly from other ideas (see below), there is a similarity to constraint dominance, since a cheaper evaluation is being used to assess the worthiness of a more expensive one. Constraint dominance is perhaps more straightforward, though, as in Brood Interleaving (and Speculative Interleaving) it is less certain that evaluating on the fast objective will provide a guide for solutions worthy of evaluation on a slower objective, and is more dependent on the structure of the problem, particularly the correlation between objectives.

Delta-evaluation in local search (see e.g. Ross et al. (1994) and Bianchi et al. (2005)) and in hybrid EAs provides another parallel to our work. Delta-evaluation refers to the use of a fast computation of the objective value of a solution by basing the computation on a "delta" (a small change) from another previously evaluated solution; this is often possible when neighborhood search is being used. With the use of delta-evaluation there remains the question about how effort should be shared between the faster local moves, and more expensive but more global ways of generating new solutions.

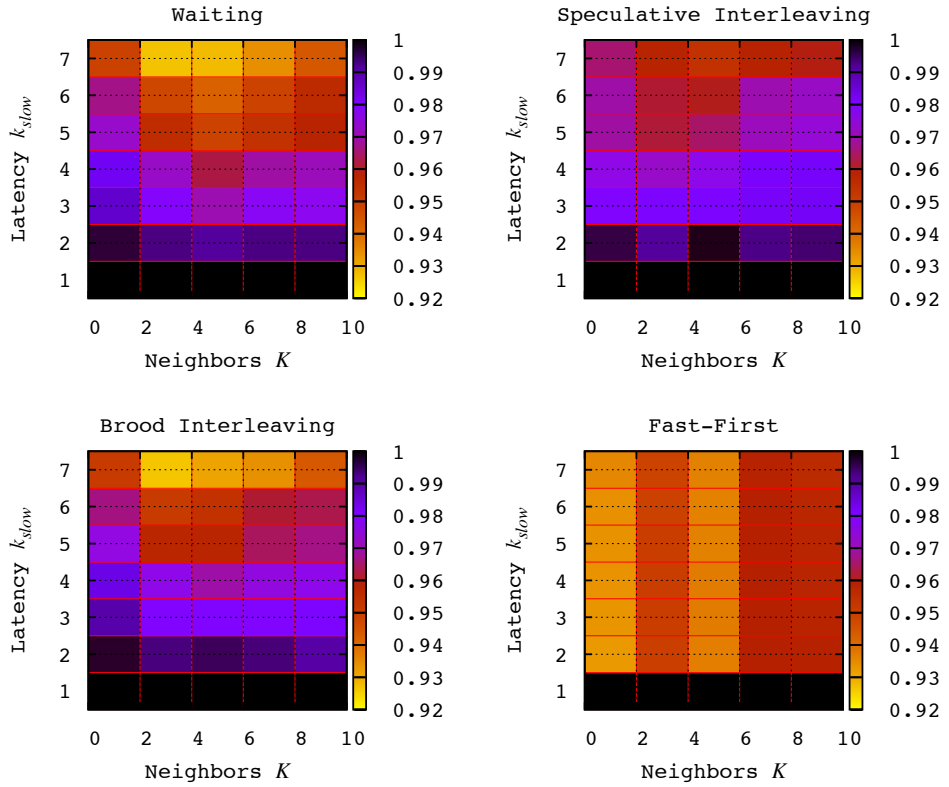

Figure 9: Proportion of attainable hypervolume (as derived in an undelayed environment) achieved by a strategy when embedded within IBEA on *MNK* landscapes after 40 generations using a population size of 50. The color coding shows the performance ratio associated with each strategy as a function of the latency (y-axis) and number of neighbors, K, each bit is interacting with (x-axis).

In contrast, our interleaving methods are more restricted since the choice of how often the faster objective can be used relative to the slow one is determined more by the problem itself: the time lag, and the relative sizes of the populations (or batches) for the slow and fast objective (assumed here to be equal for simplicity). We also note here that delta evaluation itself can be further related to the idea of using a hierarchy of successively more accurate but more costly functions for evaluating a solution, sometimes employed in engineering optimization, but we have left this promising vein of research out of our considerations here as it falls under the category of metamodeling approaches, which is not in our scope (but see the Discussion section).

Staying with our interleaving methods, we note that Brood Interleaving is inspired by the idea of brood selection, as first proposed by Tackett & Carmi (1994), and developed subsequently by Walters (1998), as stated in Section 3.1.3. It is interesting to note that a similar approach to brood selection/recombination was recently rediscovered by Doerr et al. (2013) when attempting to design an algorithm for OneMax with running time equal to the lower bound. Overproduction followed by culling, if culling can be guided by the objective function without its full cost, is proving to be an efficient method to obtain progress. Similar motivations apply in our use of the same mechanism.

Speculative Interleaving, as the name suggests, is inspired by speculative parallelization (Marchesi et al., 1994), a method that can be used to speed up individual-based (as opposed to population-based) optimizers by evaluating all or some future search paths to some depth. Although speculative parallelization has not always proved to be the most effective way to parallelize algorithms like simulated annealing (Chandy et al., 1997), our comparison to Brood Interleaving showed

that in some important cases flattening future generations into the present one is an effective way to guide search.

As pointed out in (Allmendinger & Knowles, 2013a), methods for dimensionality reduction of the objective space (Brockhoff & Zitzler, 2009) may prove to be a fruitful direction also for delayed objectives. The usual rationale for dimensionality reduction is of course different; it is that many-objective problems are inherently difficult, or difficult for some existing MOEAs, and so removing of 'unnecessary' objectives, if any can be identified, is rational. But equally a similar argument applies, only more so, if one of the unnecessary objectives is more costly to evaluate than its proxy (a highly correlated objective). Nevertheless, there is a caveat which applies in both cases: although objectives may be correlated, they are usually supposed to be independent functions. Hence, the correlation between an expensive to evaluate objective and its proxy is only a statistical observable, which may not hold all over the search space and in particular may not hold crucially at or near optimality, i.e. the tradeoffs may be apparent only near the Pareto front. Thus removal of objectives on statistical grounds could be unwise; for these reasons we have chosen to investigate other methods that retain evaluation on the slow objective, but still exploit correlation between objectives where it exists.

Amongst the methods we investigated, Fast-First does not interleave objectives, but rather concentrates on one objective for some time before changing to the others. This is rather reminiscent of the two-phase local search (TPLS) method introduced by Paquete & Stützle (2003) for bi-objective combinatorial optimization problems. Paquete cites the connectedness and 'global convexity' of the Pareto front in many multiobjective combinatorial optimization problems as the motivation behind this approach. The TPLS first optimizes one objective before optimizing a series of weighted sum programs with the weights gradually changing from the first objective (alone) to favoring the second objective. The solution to the previous program is used as the starting solution for the successive one.

Earlier work by two of us on *ephemeral resource constraints* (ERCs) (Allmendinger & Knowles, 2011; Allmendinger, 2012; Allmendinger & Knowles, 2013b) was the main driver for the proposed method that we call here Waiting. ERCs are not standard constraints, but are restrictions on the set of solutions that are actually evaluable at a given time during optimization, arising due to resourcing issues. In considering how EAs should be applied in such problems, a straightforward solution, which can usually be applied, is to wait for the resourcing issue to pass, and not evaluate any solutions in the meantime. Although this wastes time steps (and often a time budget is imposed), the method has two key virtues: (i) its simplicity of implementation; and, (ii) it avoids introducing a search bias which might have happened had solutions in only a restricted part of the solution space been evaluated. We found that this method was a good baseline method in our studies on ERCs, and was surprisingly effective in many cases. Similarly, here, we found that waiting for the slow objective, and not using the faster one, is worthy of consideration, and provides an excellent baseline.

6. Discussion

Our study above is necessarily limited and leaves open a number of questions concerning how to deal with problems subject to delays of objectives. This section briefly considers some of these questions.

How well do approximation-free strategies fare against approximation-based strategies? Having looked at both, approximation-free strategies in this work and approximation-based strategies in our previous work (Allmendinger & Knowles, 2013a), it would be interesting and helpful to know which type of approach is best suited for a given problem. We have done some preliminary experiments on the problems considered here using the approximation-based strategy that performed best and most robustly in our previous study. In brief, this strategy maintains an unlimited population (or archive) and submits the most recently generated solutions for evaluation on the slow objective(s). Missing objective values, i.e. values of the slow objective of solutions that have been evaluated on the fast objective only, are approximated using a fitness inheritance-based method. This method fills the missing objective value of a solution with the objective value of the genetically closest solution that has been evaluated on both the fast and slow objective. After completing the batch of evaluations on the slow objective, the approximated solution values are replaced with the true (slow objective function) values. (Then, rather than relying on evolution alone to update the population and search direction upon observing the new objective function values, potentially, we could accelerate this process using the "guardian dominator" approach of Fieldsend & Everson (2014).)

Figure 10 shows several plots comparing this strategy with the approximation-free strategies introduced in this work on the LOTZ problem. In general, the issue of controlling search bias induced by the fast objective

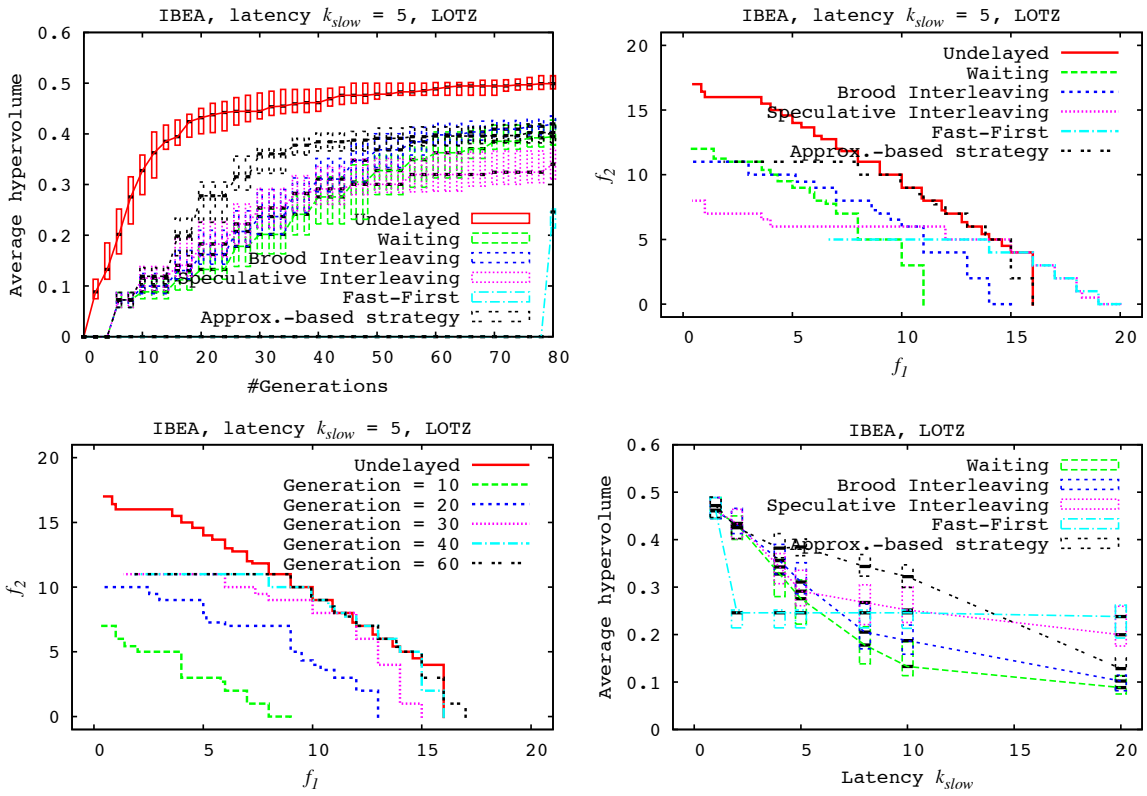

Figure 10: Selection of plots comparing the performance of approximation-free strategies against an approximation-based strategy for various problem scenarios: convergence plot on the LOTZ problem using a latency of 5 generations on the slow objective (here f_2) (top left), relative performance of the approximation-based strategy compared to the undelayed problem on the mapped OneMax problem (top right), attainment surfaces obtained on the LOTZ problem using a latency of 5 generations on the slow objective (here f_2) (bottom left), and median and interquartile ranges of the hypervolume achieved on the LOTZ problem as a function of the latency (bottom right). For every setting marked by a point in the bottom right line graph, a Friedman test (significance level of 5%) has been carried out. In the top left plot, the approximation-based strategy peforms best out of the five strategies for $15 <$ #Generations < 60. In the bottom right plot, the approximation-based strategy performs best out of the five strategies in the range $2 < k_{slow} \leq 10$, and Fast-First for $k_{slow} = 20$. There is no clear winner for the other settings.

without reducing convergence speed significantly exists with approximation-based strategy too. The top left plot shows the convergence behavior of the approximation-based strategy in comparison to the approximation-free strategies introduced in this study. It can be seen from the plot that the approximation-based approach converges even faster than Speculative Interleaving at the beginning of the search but then stagnates in performance to be eventually overtaken by Brood Interleaving and Waiting. Looking at the attainment surfaces obtained after 40 generations (see top right plot), it is apparent that the approximation-based strategy is less affected by search bias towards the fast objective (i.e. is less likely to converge to a population where many solutions are of high quality wrt the fast objective only) than Speculative Interleaving and Fast-First. In fact, as can be observed from the bottom left plot, the strategy converges (quickly) to the middle of the Pareto-front, similar to Waiting, but then stagnates there (due to a loss of diversity in the population as many solutions are copies of each other). Finally, it is important to note that the performance of the approximation-based strategy depends heavily on the accuracy of objective value approximations made. It is easy to see that this accu-

racy reduces as the number of solutions evaluated on the slow objective reduces. The bottom right plot confirms this assumptions, indicating a significant degradation in the performance of the strategy with increasing latency k_{slow}. On the mapped OneMax problem and multiobjective NK landscapes, the approximation-based strategy is affected similar to Speculative Interleaving for different objective function correlations and K values, respectively. The main difference between both strategies was observed on the mapped OneMax problem for positive correlations and long latencies (see Figure A.6 in the Appendix); here Speculative Interleaving performs significantly better.

Generalization of strategies to a pair of objectives where k_{slow} is not an exact multiple of k_{fast}: In Definition 2.7 we defined the latencies associated with the objectives to be exact multiples of each other. Of course, in reality, this may not be the case. However, this definition simplifies things greatly to synchronize the objectives in this way, and more generally this can be done as follows. Let's say the fast objective takes 1 unit and the slow one 1.7. Then we could just approximate the slow one as taking 2 timesteps for synchronization purposes (and we would have to put in a buffer delay on the second objective to make the synchronization work). However, if the fast objective took one unit of time and the slow objective took 1.5, then it would be ideal to define the first as being 2 timesteps, and second as 3, so that the amount of buffering is kept small. We have not considered this second way of synchronizing, but believe that the algorithms could be altered to do it without much additional work.

Generalization of strategies to $m > 2$ objectives: Our experimental study investigated multiobjective problems with $m = 2$ objectives of which one was fast and one slow. A pressing question is whether the strategies can be generalized to problems with $m > 2$ objectives and multiple objectives with different latencies. The strategies, Waiting and Fast-First, are readily applicable to other problem setups by simply going at the rate of the slowest and fastest objective, respectively. To be able to apply the two Interleaving strategies we could split the objectives into two groups based on their latencies, i.e. a group containing rather slow objectives and a group with fast objectives, and use some buffering (waiting) within each group to synchronize the evaluations. In the case where the group with the slow objectives contains multiple elements (objectives), we would use a MOEA (instead of a single-objective EA) to drive the optimization during the interleaving periods. An alternative and probably more complex strategy is to first rank objectives according to their latencies, and then apply a nested approach where the faster objectives are used as look ahead for slower objectives, and, in turn, these objectives as look ahead for even slower objectives, and so on. Clearly, the application of approximation-based approaches could be of great benefit in such challenging scenarios, replacing the need of evaluating all objective functions.

Instantiations of delay-handling strategies on steady-state MOEAs: Unlike generational-based MOEAs, which evaluate a set (population) of $\lambda > 1$ solutions at any time, steady state MOEAs create (and evaluate) one solution at a time (and thus are not usable off-the-shelf in the case where several solutions can be evaluated in parallel). This does not effect the Waiting and First-First strategy, which can be used in the same way as within a generational-based MOEA (though we might want to use the Waiting strategy to create an unbiased initial population in the case of First-First and the two Interleaving strategies). Slight modifications are, however, necessary for Brood and Speculative Interleaving to account for the fact that only k_{slow} fast evaluations are performed during each "interleaving period", compared to $k_{\text{slow}} \times \lambda$ fast evaluations in a generational MOEA. Consequently, to simulate the performance of Brood Interleaving obtained in a generational MOEA, the interleaving populations for a steady state-based MOEA can be created by applying selection and variation to the current EA population G_i (which is of size λ) instead of the current generation on the slow objective, G_i^{slow} (which is also of size λ in the case of a generational MOEA). Similarly, to simulate the performance of Speculative Interleaving obtained in a generational MOEA, which involves the application of a single-objective EA for $k_{\text{slow}} - 1$ generations, each of the interleaving populations of a steady-state MOEA can represent a (single) mutant created by applying variation to the single solution (parent) forming G_i^{slow}; the single best mutant (in terms of the fast objective) can then be selected to give G_i^{slow} assuming it is better than its parent, otherwise a single offspring can be created by random (parental) selection and variation applied to the interleaving populations (mutants) to give G_i^{slow}.

The performance of the delay-handling strategies when embedded to a steady-state MOEA, as described above, seems to be similar to the performance obtained for a generational MOEA. The main difference we observed is that Brood Interleaving and Waiting perform better within a steady-state MOEA than a generational MOEA for problem with high correlations be-

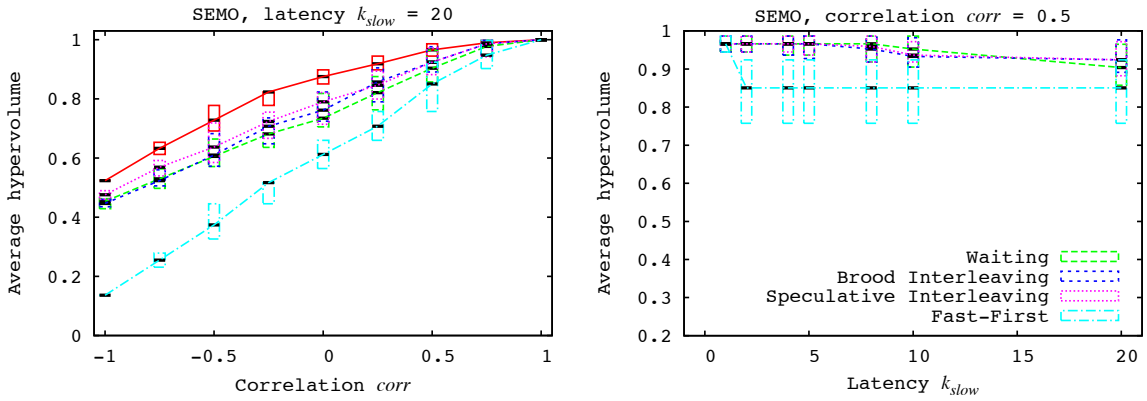

Figure 11: Median and interquartile ranges of the hypervolume achieved on the mapped OneMax problem by different strategies embedded within a steady-state EMOA, here the Simple Evolutionary Multiobjective Optimizer (SEMO) (Laumanns et al., 2004), after 40 generations using a population size of 50. In the left plot, the latency was set to 5 generations on the slow objective (here f_2), and the correlation was varied from -1 to 1, in steps of 0.25. In the right plot, the correlation was set to 0.5 between the two objectives and the latency was varied. For every setting marked by a point in the line graphs, a Friedman test (significance level of 5%) has been carried out. In the left plot, Speculative Interleaving performs best out of the four strategies in the range $-1 \leq corr < 0$. There is no clear winner for the other settings.

tween objectives, especially when combined with long delay lengths. This observation can also be made from Figure 11 where results are shown for an example of a steady-state MOEA, in this case the Simple Evolutionary Multiobjective Optimizer (SEMO) (Laumanns et al., 2004). This pattern may be due to the relatively higher selection pressure of a steady-state MOEA (compared to a generational MOEA, such as IBEA). However, a more thorough investigation is needed to confirm the performance differences between these two reproduction schemes.

Relationship to problems with asynchronous evaluations: Asynchronous evaluation in optimization in the context of grid computing was considered in [12,16]. 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.[12] 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).

How to select an appropriate strategy for dealing with delayed objectives in the absence of knowledge about the problem at hand? If the experimentalist (i.e. an expert on the problem domain at hand) is unaware about correlations between objectives and/or structure of the fitness landscape, then some knowledge about these aspects might be obtained from evaluating a set of random solutions on the problem. The knowledge gained might be used to select an appropriate strategy, and the solution set evaluated could form the initial population of the MOEA within which the strategy is embedded. Alternatively, if this approach is not feasible (e.g. because it is too expensive) or does not yield useful insights into the problem, one might select an appropriate strategy based on the latencies of the objectives. In this work we have shown that a strategy with high selection pressure, such as Speculative Interleaving, is well-suited for small budgets and long latencies, whilst a Waiting strategy is more appropriate for large budgets. Finally, confidence in the strategy selected can be increased by designing problem functions that simulate the problem at hand as closely as possible, and then to test several strategies offline on these functions and use the best one for the real-world problem.

7. Conclusion

Continuing our previous work (Allmendinger & Knowles, 2013a), we have considered a multiobjective optimization scenario in which the objective functions of a problem require different evaluation times. This kind of problem can be encountered in closed-loop optimization scenarios, where physical experiments may be of differing temporal durations, as well as problems in which objective functions are evaluated on the computer but have differing latency (e.g. due to lengthy computer simulations). We considered scenarios with two objectives, a fast and a slow objective, and, unlike to our previous work, proposed and analyzed several approximation-free strategies to deal such scenarios: Waiting, Fast-First, Brood Interleaving, and Speculative Interleaving. We showed how these strategies can be augmented on generational and steady state-based MOEAs, and assessed them on three well-known binary test functions: (i) the well-understood LOTZ problem, (ii) a family of mapped OneMax problems to investigate the impact of correlation between objectives on strategy performance, and (iii) multiobjective NK landscapes (MNK landscapes) to investigate the impact of varying fitness landscape ruggedness on strategy performance.

The experimental study revealed that the performance of all strategies is affected by the presence of delayed objectives. In general, the degree to which performance is affected depends on how well a strategy is able to control search bias towards solutions that are fit on the fast objective without degrading population diversity and convergence speed severely. Based on the experimental study, we can conclude tentatively that a strategy with relatively high selection pressure towards optimizing the fast objective, such as Speculative Interleaving, performs well if little optimization time is available and/or delays are long, objectives are positively correlated, and fitness landscapes are rugged.

A strategy that focuses on optimizing the fast objective only, such as Fast-First, has shown to perform well for problems with highly positively correlated objectives. Furthermore, Fast-First was also unaffected by the length of delays, and performed better within a generational MOEA than a steady state-based MOEA. In fact, when embedded within a generational-based MOEA and applied to problems with long delays, the performance of Fast-First was competitive to the one of Speculative Interleaving.

Compared to Speculative Interleaving and Fast-First, we observed that the strategies Waiting and Brood Interleaving are less aggressive with respect to the optimization of the fast objective, and that, with increasing budgets, both strategies become increasingly competitive. They perform similarly in several settings, but Brood Interleaving outperforms Waiting for long delays and large problem sizes. Furthermore, Brood Interleaving turns out to perform significantly better when used in combination with a steady state-based MOEA (here SEMO) rather than a generational-based MOEA (here IBEA).

Our study has shown that, for most scenarios, smart approximation-free strategies are more competitive than simple approaches, such as Waiting, to deal with multiobjective problems where objectives have differing evaluation times. In particular, EA performance depends on the mechanism used to create offspring (interleaving populations) during the periods the slow objective is evaluated. Hence, future research could look at merging the strategies proposed here (e.g. for creating offspring) with the approximation-based strategies we considered in our previous work (e.g. for estimating their objective values). To improve the approximation-based strategy itself, or, more precisely, the way a population is updated after observing the true values of the slow objective function, we could experiment with the "guardian dominator" approach of Fieldsend & Everson (2014). We believe that there is also scope for improvement on the strategy Fast-First. For example, one could learn (using a reinforcement learning-based approach as done e.g. in (Allmendinger & Knowles, 2011)) when to switch during the search between single-objective optimization of the fast objective and multiobjective optimization of both the fast and slow objective. To improve the applicability of the strategies proposed here, it is also important to extend and investigate them for multiobjective problems with $m > 2$ objectives where possibly all objectives may have different latencies. Finally, we can look into merging our strategies with surrogate-assisted evolutionary optimization (Ong et al., 2005; Jin, 2011) for coping with delayed environments, as well as get inspiration from asynchronous evolution techniques (Lewis et al., 2009; Harada & Takadama, 2014), which account for latencies across individual solutions rather than objectives.

References

Aguirre, H. E., & Tanaka, K. (2007). Working principles, behavior, and performance of MOEAs on MNK-landscapes. *European Journal of Operational Research*, *181*, 1670–1690.

Allmendinger, R. (2012). *Tuning Evolutionary Search for Closed-Loop Optimization*. Ph.D. thesis School of Computer Science, The University of Manchester.

Allmendinger, R., & Knowles, J. (2011). Policy learning in resource-constrained optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2011)* (pp. 1971–1978).

Allmendinger, R., & Knowles, J. (2013a). 'Hang On a Minute': Investigations on the effects of delayed objective functions in multiobjective optimization. In *Evolutionary Multi-Criterion Optimization* (pp. 6–20).

Allmendinger, R., & Knowles, J. (2013b). On handling ephemeral resource constraints in evolutionary search. *Evolutionary Computation*, *21*, 497–531.

Altenberg, L. (1994). The evolution of evolvability in genetic programming. *Advances in Genetic Programming*, *3*, 47–74.

Beume, N., Naujoks, B., & Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, *181*, 1653–1669.

Bianchi, L., Knowles, J., & Bowler, N. (2005). Local search for the probabilistic traveling salesman problem: Correction to the 2-p-opt and 1-shift algorithms. *European Journal of Operational Research*, *162*, 206–219.

Box, G. (1957). Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics*, *6*, 81–101.

Brockhoff, D., & Zitzler, E. (2009). Objective reduction in evolutionary multiobjective optimization: theory and applications. *Evolutionary Computation*, *17*, 135–166.

Caschera, F., Gazzola, G., Bedau, M., Moreno, C., Buchanan, A., Cawse, J., Packard, N., & Hanczyc, M. (2010). Automated discovery of novel drug formulations using predictive iterated high throughput experimentation. *PloS ONE*, *5*, e8546.

Chandy, J. A., Kim, S., Ramkumar, B., Parkes, S., & Banerjee, P. (1997). An evaluation of parallel simulated annealing strategies with application to standard cell placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *16*, 398–410.

Chmielewski, H. T. (2013). *Satisfying Multiple Priorities with a Diversity Preserving Evolutionary Algorithm*. Master's thesis North Carolina State University.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*, 311–338.

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

Doerr, B., Doerr, C., & Ebel, F. (2013). Lessons from the black-box: fast crossover-based genetic algorithms. In *Proceeding of the fifteenth annual conference on Genetic and Evolutionary Computation (GECCO-2013)* (pp. 781–788).

Droste, S., Jansen, T., & Wegener, I. (2006). Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of computing systems*, *39*, 525–544.

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*, 760–771.

Fieldsend, J., & Everson, R. (2014). Efficiently identifying Pareto solutions when objective values change. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2014)* (pp. 605–612).

Fonseca, C. M., & Fleming, P. J. (1996). On the performance assessment and comparison of stochastic multiobjective optimizers. In *Parallel Problem Solving from Nature (PPSN IV)* (pp. 584–593).

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*, 675–701.

Harada, T., & Takadama, K. (2014). Asynchronously evolving solutions with excessively different evaluation time by reference-based evaluation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2014)* (pp. 911–918).

Jansen, T., & Zarges, C. (2013). Performance analysis of randomised search heuristics operating with a fixed budget. *Theoretical Computer Science*, *545*, 39–58.

Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, *1*, 61–70.

Kauffman, S. A. (1993). *The Origins of Order*. Oxford University Press.

Laumanns, M., Thiele, L., & Zitzler, E. (2004). Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. *IEEE Transactions on Evolutionary Computation*, *8*, 170–182.

Lewis, A., Mostaghim, S., & Scriven, I. (2009). Asynchronous multi-objective optimisation in unreliable distributed environments. *Biologically-Inspired Optimisation Methods*, (pp. 51–78).

Marchesi, M. L., Molinari, G., & Repetto, M. (1994). A parallel simulated annealing algorithm for the design of magnetic structures. *IEEE Transactions on Magnetics*, *30*, 3439–3442.

Ong, Y., Nair, P., Keane, A., & Wong, K. (2005). Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In *Knowledge Incorporation in Evolutionary Computation* (pp. 307–331).

Paquete, L., & Stützle, T. (2003). A two-phase local search for the biobjective traveling salesman problem. In *Evolutionary Multi-Criterion Optimization* (pp. 479–493).

Rechenberg, I. (2000). Case studies in evolutionary experimentation and computation. *Computer methods in applied mechanics and engineering*, *186*, 125–140.

Ross, P., Corne, D., & Fang, H.-L. (1994). Improving evolutionary timetabling with delta evaluation and directed mutation. In *Parallel Problem Solving from Nature (PPSN III)* (pp. 556–565).

Small, B., McColl, B., Allmendinger, R., Pahle, J., López-Castejón, G., Rothwell, N., Knowles, J., Mendes, P., Brough, D., & Kell, D. (2011). Efficient discovery of anti-inflammatory small molecule combinations using evolutionary computing. *Nature Chemical Biology*, *7*, 902–908.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms* (pp. 2–9).

Tackett, W. A., & Carmi, A. (1994). The unique implications of brood selection for genetic programming. In *Proceedings of First IEEE World Congress on Computational Intelligence* (pp. 160–165).

Verel, S., Liefooghe, A., Jourdan, L., & Dhaenens, C. (2011). Pareto local optima of multiobjective NK-landscapes with correlated objectives. In *Evolutionary Computation in Combinatorial Optimization* (pp. 226–237).

Walters, T. (1998). Repair and brood selection in the traveling salesman problem. In *Parallel Problem Solving from Nature (PPSN V)* (pp. 813–822).

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*. Ph.D. thesis Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

Zitzler, E., Knowles, J., & Thiele, L. (2008). Quality assessment of pareto set approximations. In *Multiobjective Optimization* (pp. 373–404). Springer.

Zitzler, E., & Künzli, S. (2004). Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature (PPSN VIII)* (pp. 832–842).

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, *7*, 117–132.

Appendix A. Supplementary Experimental Results

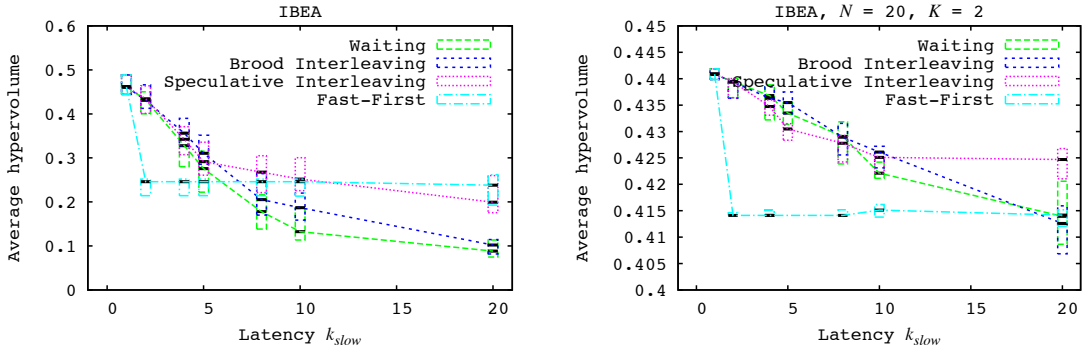

Figure A.1: Median and interquartile ranges of the hypervolume achieved on the LOTZ problem (left) and on *MNK* landscapes (for N=20 and K=2) (right) by different strategies embedded within IBEA after 40 generations using a population size of 50. Results are shown for different latencies. For every setting marked by a point in the line graphs, a Friedman test (significance level of 5%) has been carried out. In the left plot, Speculative Interleaving performs best out of the four strategies for $5 < k_{slow} \leq 10$, and Fast-First for $k_{slow} = 20$. In the right plot, Speculative Interleaving performs best out of the four strategies for $k_{slow} = 20$. There is no clear winner for the other settings.

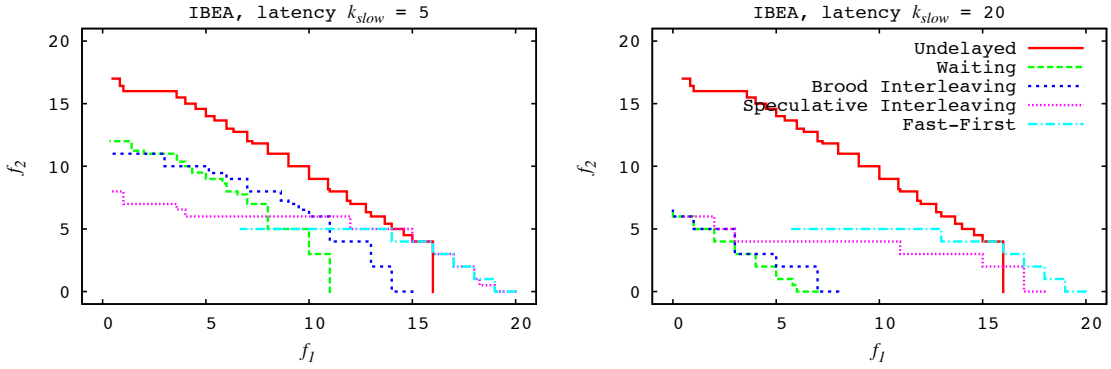

Figure A.2: Median attainment surface on the LOTZ problem obtained by different strategies embedded within IBEA after 40 generations using a population size of 50. Results are shown for a latency of 5 (left) and 20 (right) generations on the slow objective (here f_2).

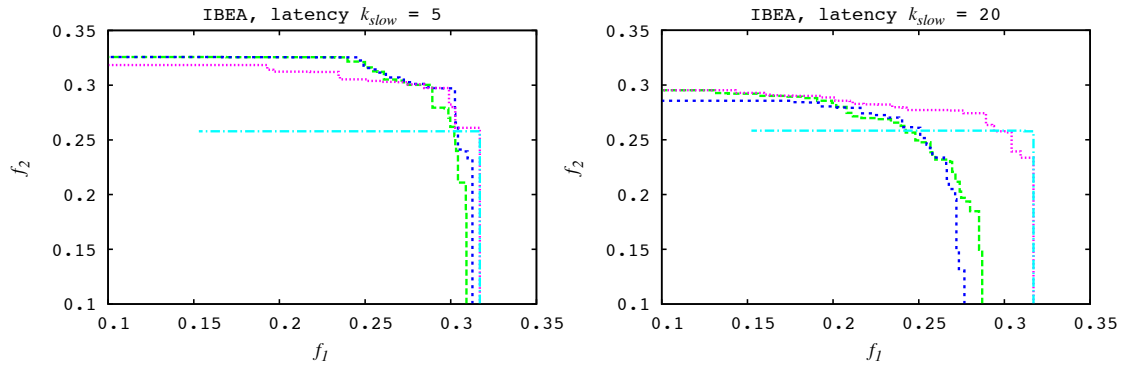

Figure A.3: Median attainment surface on MNK landscapes (for N=20 and K=2) obtained by different strategies embedded within IBEA after 40 generations using a population size of 50. Results are shown for a latency of 5 (left) and 20 (right) generations on the slow objective (here f_2).

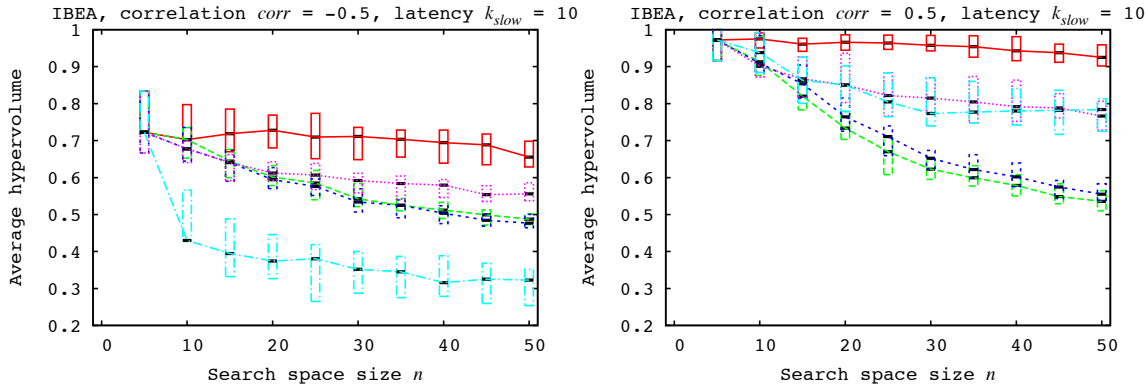

Figure A.4: Median and interquartile ranges of the hypervolume achieved by different strategies embedded within IBEA on the mapped OneMax problem after 40 generations using a population size of 50. Results are shown for a latency of 10 generations on the slow objective (here f_2) as a function of problem size, which is varied from 5 bits to 50, in steps of 5. For every setting marked by a point in the line graphs, a Friedman test (significance level of 5%) has been carried out. In theleft plot, Speculative Interleaving performs best out of the four strategies for $n > 20$. There is no clear winner for the other settings.

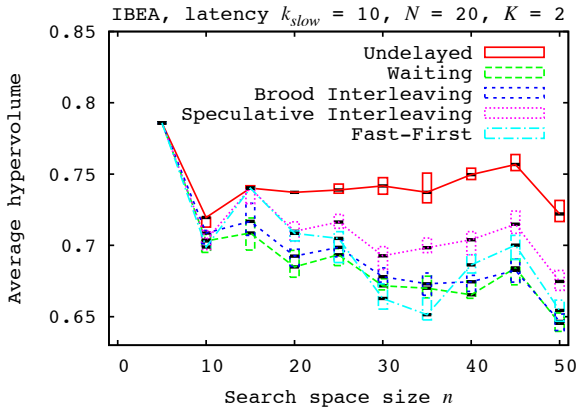

Figure A.5: Median and interquartile ranges of the hypervolume achieved on *MNK* landscapes (for N=20 and K=2) by different strategies embedded within IBEA after 40 generations using a population size of 50. Results are shown for a latency of 10 generations on the slow objective (here f_2) as a function of problem size, which is varied from 5 bits to 50, in steps of 5. For every setting marked by a point in the line graphs, a Friedman test (significance level of 5%) has been carried out. Speculative Interleaving performs best out of the four strategies for $n > 20$. There is no clear winner for the other settings.

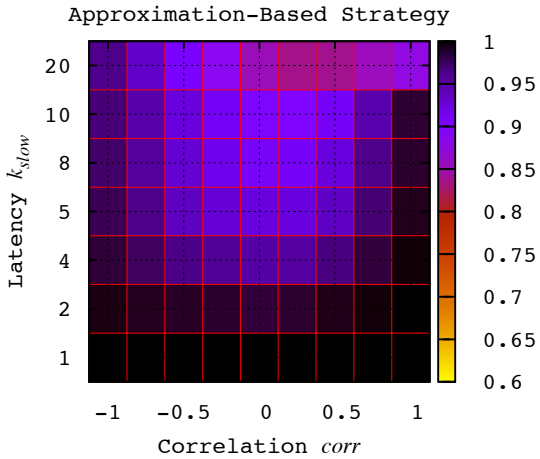

Figure A.6: Proportion of attainable hypervolume (as derived in an undelayed environment) achieved by an approximate-based strategy on the mapped OneMax problem after 40 generations using a population size of 50. The color coding shows the performance ratio associated with each strategy as a function of the latency (y-axis) and the correlation between the objectives (x-axis).