Towards Adaptation in Multiobjective Evolutionary Algorithms for Integer Problems

Günter Rudolph
Department of Computer Science
TU Dortmund University
Dortmund, Germany
guenter.rudolph@tu-dortmund.de

Markus Wagner
Department of Data Science and AI
Monash University
Clayton, Australia
markus.wagner@monash.edu

***Abstract*—Parameter control refers to the techniques that dynamically adapt the parameter values of the evolutionary algorithm during the optimization process, such as population size, crossover rate, or operator selection. Adaptation can improve the performance and robustness of the algorithm, however, parameter control mechanisms themselves need to be designed and configured carefully. With this article, we contribute a systematic investigation of an adaptive, multi-objective algorithm that is designed for the optimisation of problems in unbounded integer decision spaces. We find that (1) adaptation outperforms the best static configurations by 39–82%, and (2) performance of the multi-objective algorithm is often independent of the adaptation scheme's initial configuration.**

***Index Terms*—multiobjective evolutionary algorithm, integer search space, step size control, self-adaptation**

I. Introduction

Multi-objective optimization (MOO) is a branch of optimization addressing problems with multiple conflicting objectives that require simultaneous optimization. MOO finds extensive applications across diverse fields, such as science, engineering, economics, and logistics, where optimal decisions frequently involve trade-offs between two or more criteria. Examples of MOO problems include maximizing profit while minimizing the environmental impact of an industrial process, maximizing the potency while minimizing the side effects of a drug, and minimizing error rates while maximizing the accuracy of a machine learning model.

Evolutionary computation (EC) encompasses a range of bio-inspired algorithms replicating natural processes like evolution, immune systems, and swarm intelligence for addressing intricate optimization problems. Successfully applied to tackle multi-objective optimization (MOO) problems, EC leverages population-based search to approximate the Pareto frontier—a collection of nondominated solutions representing optimal trade-offs among objectives. The field of EC methods for MOO, also termed evolutionary multi-objective optimization (EMO) methods, has witnessed extensive study and development over the past decades, yielding a diverse array of algorithms. For a comprehensive overview, interested readers are directed to two surveys [14, 15].

Many EC methods are parameterized, necessitating the setting of parameters; for instance, population sizes and probabilities for variation operators. These parameters are subject to adjustment for performance enhancement and robustness. Predictably, following the taxonomy by Eiben et al. [4] on parameter settings, they can be fixed through offline optimization (also called "parameter tuning") or they can be dynamically set based on online optimization (also called "parameter control"). In the latter, we can distinguish three approaches: (1) deterministic approaches assume a consistent pattern in optimal settings; (2) adaptive approaches utilize feedback from the optimization process; and (3) self-adaptive approaches embed parameter choices within individuals. It is noteworthy that while this taxonomy provides a classification for certain mechanisms, the usage of terms like "adaptive" and "self-adaptive" in existing literature can be quite liberal, especially with the influence of related concepts from other domains such as "self-adaptive software systems" or "self-driving cars". Our work can be viewed as "adaptive" following Eiben et al., or as self-adaptive in that the algorithms adjust themselves based on observations.

Interestingly, despite the necessity of offline and online optimization of algorithms, work on adaptive parameter control in MOO seems a much under-explored topic. For example, the referenced taxonomy on parameter settings overlooks MOO, and the two MOO surveys mentioned earlier only touch upon a handful of works involving some form of adaptation, where it is often limited to adjustments in the objective space, such as dynamic reference point allocation, or focuses on adapting population sizes and varying operator selection. But even in the more recent literature, especially adaptive approaches are few and far apart, even though self-adaptive approaches enjoy a long history and popularity, for example, with that of Abbas [5] being one of the first and with that of Igel et al. [9] being one that is based on self-adaptive, single-objective evolution strategies.

In contrast to this, the area of a more "algorithm-level", adaptive control appears very much overlooked, to control aspects such as population sizes, crossover rates, or the choice of crossover operators in general, possible because "the situation is much more complicated than in single-objective optimization" as Wessing et al. [17] noted. Noteworthy examples are the investigations [11, 12, 16] which dynamically adjust the mutation parameter and the crossover rate based on progress in the objective space in multi-objective differential evolution,

and the adaptive operator selection [13] that is based on improvements in the objective scores. However, these works tend to share two characteristics:

- The mechanisms are outlined, and a singular parameterization is experimentally assessed using standard benchmarks, usually with an emphasis on surpassing the state-of-the-art, rather than systematically exploring the design choices underpinning the mechanisms.
- The problems involve continuous design spaces, posing a challenge in devising practical adaptive mechanisms. For instance, when success is gauged by the number of nondominated solutions generated, minor mutations may mislead mechanisms into adhering to those minor changes instead of achieving substantial progress toward the true Pareto frontier with larger mutations.

When it comes to systematic investigations that are purely theoretical or systematic from the ground up, works are typically limited to single objectives, and to convergence for continuous settings [10] or to bitstrings for computational complexity analyses. For example, Doerr et al. presented a simple $(1+1)$ EA with success-based multiplicative mutation rate updates [20] and investigated the influence of the adaptation scheme's starting values [19]. The only theoretical work that we are aware of on self-adaptation and on single-objective problem with integer decision spaces is another one by Doerr et al. [18].

With this present article, we make two contributions:

1) We define an adaptive, multi-objective framework for unbounded integer decision spaces intentionally constructed from well-understood components, aiming to facilitate future theoretical investigations. We focus on this particular problem class since it is widely unexplored at present in theory as well as algorithm design.
2) We systematically explore the parameterization of the framework through a full factorial experimental design. Our observations indicate the necessity of adaptive approaches to achieve the best results, emphasizing the importance of setting up even adaptive approaches correctly.

The remainder of the paper is organized as follows. In Section II, we define a generalization of the biobjective optimization problem proposed in [21], which has an unbounded integer search space. Then, in Section III, we describe our algorithmic starting point, which we base on SEMO [6] and which we tailor for integer settings to employ a Bilateral Geometrical Distribution [3]. Section IV then describes how we employ the concept of Rechenberg's success rule [1] to control a single step size being valid for all individuals; this results in our parameterized evolutionary algorithm named ASEMO (adaptive SEMO). Next, we lay out the experimental plan in Section V and then report on the results of our computational study in Section VI, before outlining directions for future work in Section VII.

Our data and code are available online [22].

Algorithm 1: SEMO: Algorithmic skeleton for evolutionary multi-objective minimization of a given d-objective function $f\colon \mathbb{Z}^n \to \mathbb{R}^d$ and a given starting point $x^{(0)} \in \mathbb{Z}^n$.

1 $P^{(0)} = \{x^{(0)}\}$;
2 $t = 0$;
3 **while** *termination criterion not met* **do**
4 choose x from $P^{(t)}$ uniformly at random;
5 $y = \text{mutation}(x)$;
6 $Q = P^{(t)} \setminus \{z \in P^{(t)} \colon f(y) \preceq f(z)\}$;
7 **if** $\nexists z \in Q \colon f(z) \prec f(y)$ **then** $P^{(t+1)} = Q \cup \{y\}$;
8 **else** $P^{(t+1)} = Q$;
9 $t = t + 1$;

II. Multiobjective Problem Class

Since an established benchmark for multi-objective unbounded integer problems does not exist apparently, we consider a slight generalization of the biobjective optimization problem (Equation 2) proposed in [21]. Let $f : \mathbb{Z}^n \to \mathbb{N}_0^2$, $u, v \in \mathbb{Z}^n$ with $u \neq v$. Then a class of biobjective optimization problems is given by

$$f(x) = \begin{pmatrix} \|x-u\|_1 \\ \|x-v\|_1 \end{pmatrix} \to \min! \tag{1}$$

where $\|\cdot\|_1$ denotes the ℓ_1-norm. If $v = -u$, $a \in \mathbb{N}$, and $u = (a, 0, \ldots, 0)^\intercal \neq 0 \in \mathbb{Z}^n$ we obtain the special case $f : \mathbb{Z}^n \to \mathbb{N}_0^2$ with

$$f(x) = \begin{pmatrix} |x_1 - a| + |x_2| + \ldots |x_n| \\ |x_1 + a| + |x_2| + \ldots |x_n| \end{pmatrix} \to \min! \tag{2}$$

for which Pareto set and Pareto frontier have been derived analytically in [21]. In the general case, a slightly modified version of the proof delivers the Pareto front

$$F^* = \{h \in \mathbb{Z}^2 : h = (k, \|u-v\|_1 - k)^\intercal, k = 0, 1, \ldots, \|u-v\|_1\} \tag{3}$$

with $|F^*| = \|u - v\|_1 + 1$ and the Pareto set

$$X^* = [u, v] \cap \mathbb{Z}^n \tag{4}$$

with $|X^*| = \prod_{i=1}^n (1 + |u_i - v_i|)$.

III. Algorithm

A. Control Flow

The algorithmic skeleton is the integer version of SEMO [6] originally designed for binary problems. By design, the offspring is accepted if it either dominates an individual in the population or if it is incomparable to all individuals in the population. Note that the offspring y replaces x from the population if $f(y) = f(x)$ (implied by Line 6). This is an intentional deviation from the original SEMO that would have rejected y in this situation. In fact, this practice is in use for many years as it supports the ability to traverse and finally leave fitness plateaus.

Another deviation from SEMO regards the concept of mutation (Line 5). In SEMO a single dimension is drawn uniformly at random and the variable associated with that dimension is mutated, whereas in *global* SEMO (GSEMO) [7] the variable of each dimension is mutated independently with probability $p = 1/n$ where n is the dimension of the decision space. We follow this approach by choosing $p \in \{1/n, 1\}$.

It remains to specify the mutation distribution and the step size control.

B. Mutation Distribution

In principle, there are infinitely many possibilities to choose a mutation probability with support $\mathbb{Z}$. Because mutations in integer space are modelled by adding a random value or vector, plausible candidates are distributions whose probability mass functions (p.m.f.s) are symmetric with respect to 0, unimodal and shapeable by a parameter. Among the distributions in this subset it was proposed (for integer settings) to pick out the distribution with maximum entropy [3]. These conditions led to a so-called *Bilateral Geometrical Distribution* [3] that appeared later also under the name *Discrete Laplace Distribution* [8]. Its p.m.f. is

$$\mathsf{P}\{\, Z = k \,\} = \frac{q}{2-q}(1-q)^{|k|} \tag{5}$$

with $q \in (0,1) \subset \mathbb{R}$ for $k \in \mathbb{Z}$ and moments

$$\mathsf{E}[\, Z \,] = 0, \mathsf{V}[\, Z \,] = \frac{2\,(1-q)}{q^2} \text{ and } \mathsf{E}[\,|Z|\,] = \frac{2\,(1-q)}{q\,(2-q)}. \tag{6}$$

We consider two approaches to apply this one-dimensional mutation distribution in n-dimensional decision space.

1) Sub-dimensional mutation ($p = 1/n$): Each dimension is mutated with probability $p = 1/n$ by adding an independently drawn integer random number with distribution (Equation 5). In this case, the probability that none of the dimensions is mutated is given by the sum over the probabilities that k dimensions have been selected for mutations but for all of these mutations we draw $Z = 0$ with probability $b = \mathsf{P}\{\, Z = 0 \,\} = q/(2-q)$. Thus, we obtain

$$\sum_{k=0}^{n} \binom{n}{k} p^k\,(1-p)^{n-k}\,b^k = (1 + p\,(b-1))^n$$

which yields

$$\left(1 + \frac{b-1}{n}\right)^n \to e^{b-1} = e^{\frac{q}{2-q}-1} \in (e^{-1}, 1)$$

for $p = 1/n$ and taking the limit $n \to \infty$. This result reveals that some action must be taken in an EA without recombination to avoid the useless fitness evaluation if such an event has happened.

2) Full-dimensional mutation ($p = 1$): Each dimension is mutated by adding a random number independently drawn from distribution (Equation 5). In this case, the expected length of the resulting random vector Z in ℓ_1-norm is

$$s = \mathsf{E}[\,\|Z\|_1\,] = n \cdot \mathsf{E}[\,|Z_1|\,] = n\,\frac{2\,(1-q)}{q\,(2-q)}$$

which can be rearranged to

$$q = 1 - \frac{s/n}{1 + \sqrt{1 + (s/n)^2}}\,. \tag{7}$$

The probability that mutation does not change any dimension is

$$\mathsf{P}\{\, Z_1 = 0 \,\}^n = \left(\frac{q}{2-q}\right)^n \to e^{-s}$$

after insertion of (Equation 7) and taking the limit $n \to \infty$.

IV. Step Size Control

Evidently, the shape of the mutation distribution (Equation 5) can be manipulated by parameter q. The larger the value of q, the narrower the p.m.f. of Z, and vice versa. The concept of a step size is probably best conveyed by the mean absolute deviation $\mathsf{E}[\,|Z|\,]$ in (Equation 6), as it roughly indicates how many units you move to the left or right on average.

Because our proxy $s = \mathsf{E}[\,|Z|\,]$ for the mean step size in one dimension can be solved for q (simply set $n = 1$ in (Equation 7))

$$q = \frac{1 + s - \sqrt{1+s^2}}{s} = 1 - \frac{s}{1 + \sqrt{1+s^2}} \tag{8}$$

we can multiplicatively adapt s as known from evolution strategies in $\mathbb{R}^n$ before we determine q from the adapted value.

In [3] it is suggested to use Schwefel's mutative self-adaptation [2] also in the integer setting, which might be reasonable if we equip each individual with its own step size. But here, we will use the concept of Rechenberg's success rule [1] to control a single step size that is then valid for all individuals.

To this end, we log the number of offspring within a certain time window, that are nondominated with respect to the population and which of them are actually dominating any member in the population. A nondominated offspring is referred to as an *acceptance* and a dominating offspring as an *improvement*.

Let $w \in \mathbb{N}$ be the size of the time window, $n_a \leq w$ be the number of acceptances and $n_i \leq w$ be the number of improvements. If $s^{(t)} \in \mathbb{R}_+$ is the current mean step size at step $t \geq 0$ and $p_s \in (0,1)$ the threshold of the relative success frequency, then we might update the mean step size after each time window as follows:

$$s^{(t+w)} = \begin{cases} s^{(t)} \cdot c^+ & \text{if } n_i/w > p_s \\ s^{(t)} \cdot c^- & \text{otherwise} \end{cases} \tag{9}$$

for some constants $c^+ > 1$ and $c^- \in (0,1)$. Alternatively, we may replace n_i by $n_i + n_a$ in (Equation 9) yielding

$$s^{(t+w)} = \begin{cases} s^{(t)} \cdot c^+ & \text{if } (n_i + n_a)/w > p_s \\ s^{(t)} \cdot c^- & \text{otherwise} \end{cases} \tag{10}$$

for possibly different choices of w, p_s, c^+ and c^-. Evidently, an appropriate hyperparameter tuning is required, although a proof of concept may be possible for any reasonable choice of hyperparameters.

To prevent the algorithm from getting stuck prematurely, the mean step size s should not fall below the threshold 1,

Algorithm 2: ASEMO: Evolutionary algorithm for a biobjective function $f\colon \mathbb{Z}^n \to \mathbb{R}^2$ with given starting point $x^{(0)} \in \mathbb{Z}^n$, mean step size $s^{(0)} \in \mathbb{R}_+$ and hyperparameters w, c^+, c^-, p_s.

$P^{(0)} = \{x^{(0)}\}$;
$n_a = n_i = 0$;
$t = 0$;
while *termination criterion not met* **do**
  choose x from $P^{(t)}$ uniformly at random;
  $y = \mathrm{mutation}(s^{(t)}, x)$;
  // selection
  $E(y) = \{z \in P^{(t)} \colon f(z) = f(y)\}$;
  $B(y) = \{z \in P^{(t)} \colon f(z) \prec f(y)\}$;
  $W(y) = \{z \in P^{(t)} \colon f(y) \prec f(z)\}$;
  $I(y) = \{z \in P^{(t)} \colon f(y) \parallel f(z)\}$;
  if $|E(y)| > 0$ **then**
    $P^{(t+1)} = P^{(t)} \setminus E(y) \cup \{y\}$;
    if $x \neq y$ **then** $n_a = n_a + 1$;
  else if $|W(y)| > 0$ **then**
    $P^{(t+1)} = P^{(t)} \setminus W(y) \cup \{y\}$;
    $n_i = n_i + 1$;
  else if $|B(y)| > 0$ **then**
    $P^{(t+1)} = P^{(t)}$;
  else
    $P^{(t+1)} = P^{(t)} \cup \{y\}$;
    $n_a = n_a + 1$;
  // step size adaptation
  if $(t+1) \mod w = 0$ **then**
    $s^{(t+1)} = update(s^{(t)}, n_a, n_i)$;
    $n_a = n_i = 0$;
  $t = t + 1$;

as this is the smallest nonzero distance between two points in integer space. Thus, whenever $s < 1$ after the adaptation via (Equation 9) or (Equation 10) we readjust the step size to $s = 1$ in the update procedure.

The modifications of the original SEMO are integrated in the pseudo code shown in algorithm 2.

V. Experimental Plan

Our main goal is to show that a success-based adaptive step size control in MOEA can be realized, works well and finally leads to better running times. As a baseline we compare the runtime of ASEMO (Algorithm 2) with SEMO (Algorithm 1) using fixed step sizes, after we have got an overview of the sensitivity of the hyperparameters.

As the adaptive algorithm ASEMO is new and basically nothing is known *a priori* about the interactions of the hyperparameters we have decided to run a full factorial design (FFD) with the parameter values given in Table I. The problem parameters and starting points can be found in Table II.

Although only few values per parameter are varied, we end up with 1 728 experiments which are run 20 times each. For each run we log the number of function evaluations (FEs), the closest distance to the Pareto front (dPF) within the current population, the current mean step size s, the current population size μ and the current number of identified elements of the Pareto frontier.

TABLE I
Parameter ranges of FFD in dimension $n = 2$.

parameter	range of values
problem id (pid)	0, 1, 2, 3
starting point $x^{(0)}$ (xid)	0, 1, 2, 3
initial step size $s^{(0)}$	1, 60 000
mutation prob. p	$1/n$, 1
window size	$20\,n$
success prob. p_s	0.1, 0.2, 0.3
decrease factor c^-	$0.5, 0.6, 0.7$
increase factor c^+	$1.5, 1.75, 2.0$

TABLE II
Problem parameters u and v for $f(\cdot)$ and starting points x in dimension $n = 2$.

id	0	1	2	3
u	$(-50, 0)$	$(-40, -10)$	$(-25, -25)$	$(0, -50)$
v	$(+50, 0)$	$(+10, +40)$	$(+25, +25)$	$(0, +50)$
x	$(0, 20000)$	$(10000, 10000)$	$(15000, 5000)$	$(20000, 0)$

Whereas most of these values can be taken directly from the algorithm, two values need a problem-specific calculation.

A. Determining distance to Pareto Front

As part of the data we will monitor during our experiments we must be able to determine the distance of a solution in objective space to the Pareto front. For all problems of our problem class with the same value for $\delta = \|u - v\|_1$ the Pareto front F^* is identical, see (Equation 3). By design, all objective vectors must be in $\mathbb{N}_0^2$. Therefore, the calculation can be split into only three cases. Let $h \in f(X) \subset \mathbb{N}_0^2$. Then

1) If $h_1 \leq \delta$ then choose $h^* = (h_1, \delta - h_1)^\intercal \in F^*$ so that $d(h, F^*) = h_2 - (\delta - h_1)$.
2) If $h_2 \leq \delta$ then choose $h^* = (\delta - h_2, h_2)^\intercal \in F^*$ so that $d(h, F^*) = h_1 - (\delta - h_2)$.
3) Otherwise choose $h^* = (\delta, 0)^\intercal \in F^*$ so that $d(h, F^*) = h_1 - \delta + h_2$.

Putting all three cases together we obtain

$$d(h, F^*) = h_1 + h_2 - \delta = \|h\|_1 - \|u - v\|_1.$$

Remark: There are other choices of h^* in the three cases above, but they all have the same distance.

B. Number of distinct optimal objective vectors discovered

We shall also keep track of the number of optimal individuals regarding the objective space, i.e., the number of elements on the Pareto front F^* that has been discovered. For this purpose we iterate over all members of the current population and check if their distance to F^* is zero. In this case, we insert the first component of the objective vector in a set-based data structure that eliminates duplicates by design. The size of this

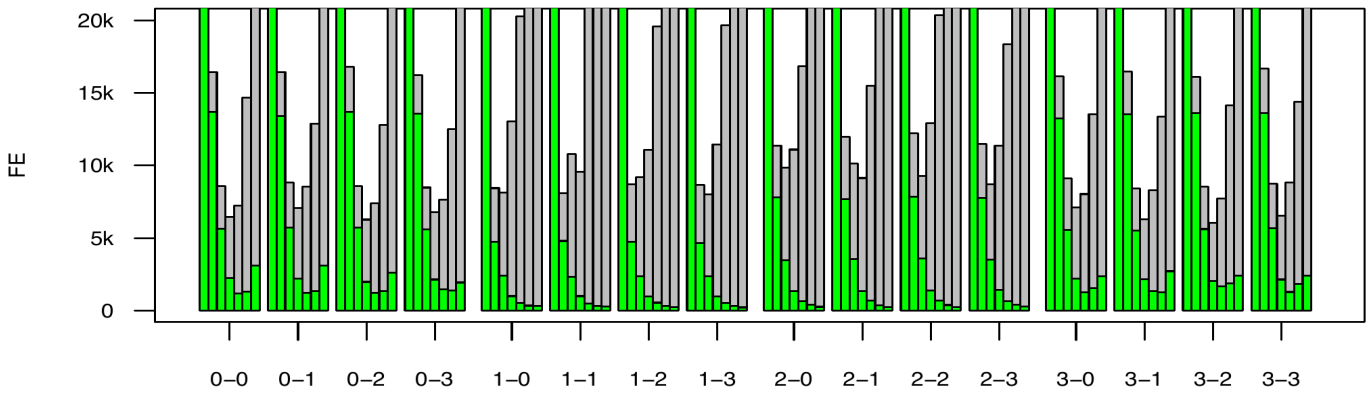

Fig. 1. Performance of the baseline approach SEMO with fixed step size and sub-dimensional mutation. Shown from left to right (in groups of 7 bars) are the 16 pid–xid combinations. Each group of 7 shows the performance of the baseline SEMO with the static step size of $s^{(0)} \in \{1, \ldots, 500\}$. Green shows the number of function evaluations needed to find the first element of the Pareto frontier, and grey shows the subsequent number of function evaluations needed to discover the entire Pareto frontier (medians of 20 runs).

set is equal to the number of distinct optimal objective vectors. If the number of elements in the set is equal to $|F^*| = \delta + 1$ at the end of the iteration the entire Pareto front has been identified and the algorithm can be terminated.

VI. Results

A. Baseline

We ran SEMO (Algorithm 1) with fixed step size and set the problem parameter as in Table II. The first four rows of Table I describe the test plan, except that we set the fixed step size to $s^{(0)} \in \{1, 10, 20, 50, 100, 200, 500\}$.

Figure 1 illustrates the results as stacked bars. The lower (green) bar represents the number of function evaluations required to find the first Pareto optimal solution, while the grey bar on top indicates the additional function evaluations needed to uncover the entire Pareto frontier. Further details are available in the caption. We cap the y-axes to facilitate readability only; our qualitative observations still hold.

In the following, our focus is on a few key observations. First, concerning the step sizes, we observe that the largest range (100–500) leads to the quickest discovery of initial solutions on the Pareto front (indicated by short green bars). However, the smallest range (1–20) proves to be the fastest in uncovering the entire Pareto frontier subsequently. Overall, the optimal fixed step size appears to be 50. These observations, in themselves, strongly advocate for adaptive parameter control.

Second, employing a static configuration with a step size of 1 unit—arguably a reasonable choice given the problem's definition on integers—can result in extended runtimes, exceeding 300,000 evaluations merely to find the first Pareto front solution. This is almost three orders of magnitude beyond the efficiency of the fastest static configurations, which manage to explore the entire Pareto frontier much faster.

Third, we note that for problems pid=1 and pid=2 (with the largest Pareto sets), it takes the most time to explore the entire frontier, especially when the step size is large.

While the figure illustrates results with sub-dimensional mutation (refer to Section III-B), the outcomes for full-dimensional mutation generally exhibit the same pattern for problems pid=1 and pid=2. For the other two problems, however, runtimes are substantially longer for the full-dimensional mutation as uncovering the entire front takes much longer then.

B. Adaptive SEMO based on domination

Next, we investigate the impact of the configuration of our ASEMO with the domination-based update rule (Equation 9). The results are shown in Figure 2 (with initial step size $s^{(0)} = 1$) and Figure 5 (with $s^{(0)} = 60\,000$). In contrast to before, we focus on the results for the full-dimensional mutation, as those with sub-dimensional mutation are slower.

In terms of the overall runtime required, we note that the adaptation mechanism performs well, almost independently of the mechanism's configuration and regardless of the problem and starting point. (1) Shifting from a static to an adaptive configuration saves 55–81% of the evaluations when comparing the fastest static and adaptive configurations (per problem and per starting point). (2) In terms of robustness to the configuration: almost all adaptive configurations are faster than the fastest, static one (Section VI-A's fastest median on any problems was 6 052): of the 1728 configurations 94% (1626) are faster, with the fastest adaptive one requiring only 1 729 (median) function evaluations.

Second, the configurations exhibiting the slowest performance are predominantly those with the highest required success probability ($p_s = 0.3$). We conjecture that this trend is influenced by the fact that achieving a success probability of 0.3, crucial for increasing the step size and thereby accelerating progress, occurs infrequently, given the baseline success probability is approximately 0.25. Supporting this conjecture is the absence of a similar pattern in Figure 5, where the initial step size is very large across all experiments, which gives the algorithm enough time to get to the front despite the step size collapsing. Examplarily, Figures 3 and 4 support this further. First, when started with a small initial step size (which in Section VI-A was performing poorly when used statically), it increases quickly until the first solution on the frontier is found and then reduces again until to aid the discovery of the entire Pareto frontier. Second, when started with a large step

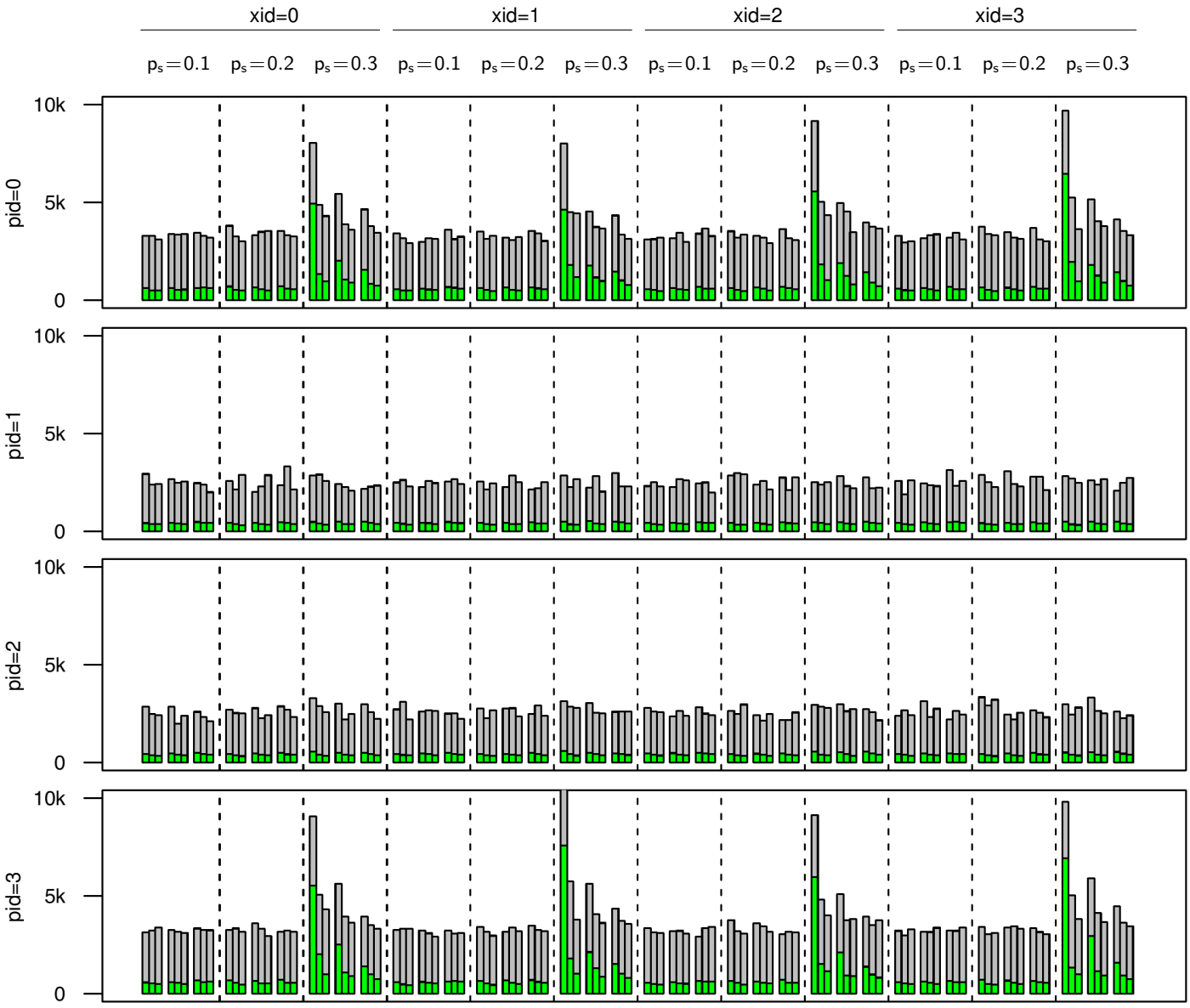

Fig. 2. Performance of ASEMO (with full-dimensional mutation and initial step size $s^{(0)}=1$) on the 16 combinations of problems and starting points (pid–xid). For each pid–xid combination, we show from left to right the three success probabilities ($p_s \in \{0.1, 0.2, 0.3\}$). For each success probability we show the nine combinations of decrease factors and increase factors (from left to right): $\left(c^-=0.5, c^+ \in \{1.5, 1.75, 2.0\}\right)$, $\left(c^-=0.6, \ldots\right)$, and $\left(c^-=0.7, \ldots\right)$. As in Figure 1, green shows the number of function evaluations needed to find the first element of the Pareto frontier, and grey shows the subsequent number of function evaluations needed to discover the entire Pareto frontier (medians of 20 runs).

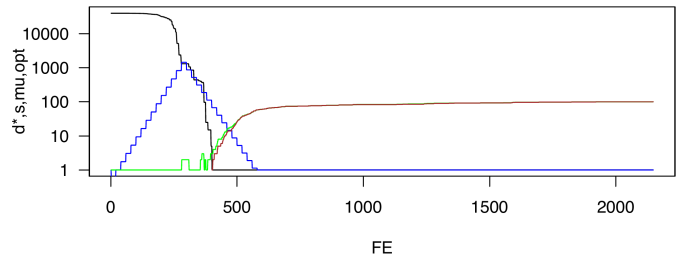

Fig. 3. Sample run, starting with the small step size $s^{(0)}=1$, pid=1, xid=2, and full-dimensional mutation. Colours: black is the distance d^* to the Pareto frontier, blue is step size, green is population size μ, red is number of optimal objective vectors found.

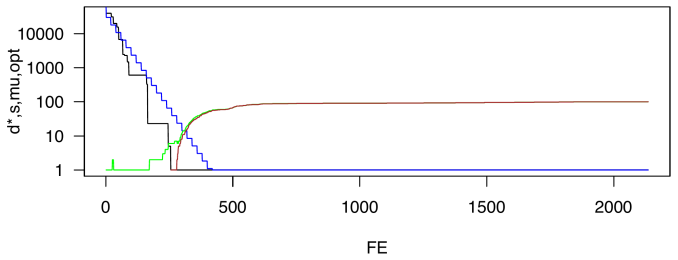

Fig. 4. Sample run, starting with the large step size $s^{(0)}=60\,000$, pid=1, xid=2, and full-dimensional mutation. Colours: black is the distance d^* to the Pareto frontier, blue is step size, green is population size μ, red is number of optimal objective vectors found.

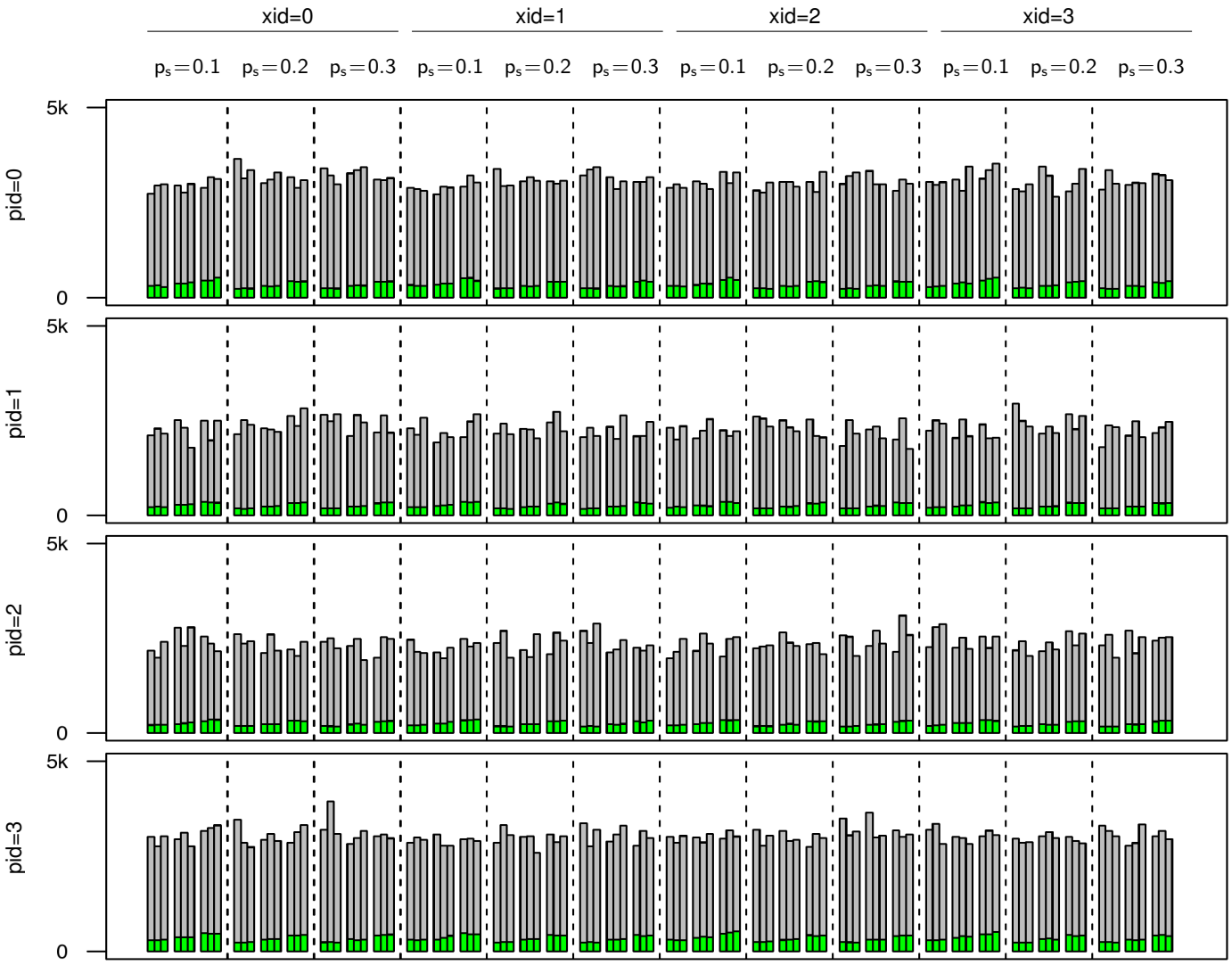

Fig. 5. Performance of ASEMO (with full-dimensional mutation and initial step size $s^{(0)}=60\,000$) on the 16 combinations of problems and starting points (pid–xid). For each pid–xid combination, we show from left to right the three success probabilities ($p_s \in \{0.1, 0.2, 0.3\}$). For each success probability we show the nine combinations of decrease factors and increase factors (from left to right): $\left(c^-=0.5, c^+ \in \{1.5, 1.75, 2.0\}\right)$, $\left(c^-=0.6, \ldots\right)$, and $\left(c^-=0.7, \ldots\right)$. As in Figure 1, green shows the number of function evaluations needed to find the first element of the Pareto frontier, and grey shows the subsequent number of function evaluations needed to discover the entire Pareto frontier (medians of 20 runs).

size (far beyond which in Section VI-A was performing poorly when used statically), it keeps decreasing and never increases.

C. Adaptive SEMO based on nondomination

Next, we investigate the impact of the configuration of our ASEMO with the update rule that considers both non-dominated and dominated solutions (Equation 10). We limit ourselves here to just the key observations.

First, ASEMO still performs well. 39–62% of evaluations are saved by the best ASEMO configuration over the best SEMO configuration (compared per starting point and problem), and 52% (902) remain below 6 052 function evaluations.

Furthermore, in direct comparison with ASEMO, these two update rules perform very comparably (based on median performance), but for interesting reasons: (1) the dominating one *stricly* outperforms the nondominating rule in on all problems (100%) with large Pareto sets (pid=1 and pid=2), because the time needed to cover the entire Pareto front is much larger when the nondominating rule is used (resulting in "too many" successes and thus too large mutation step sizes); (2) on the other two problems (which have small Pareto sets), the nondominating rule is faster 80% (688) of the time, in particular when the needed success probability is high ($p_s \in \{0.2, 0.3\}$).

VII. Summary and Future Work

We defined ASEMO as an flexible framework for the systematic investigation of adaptation in multi-objective optimisation on problems with integer decision spaces. We showed that adaptive parameter control is needed to efficiently solve the investigated problems. ASEMO saves 39–82% of function evaluations, while being robust to changes to its configuration.

Still, we have identified situations that warrant further research. First, a large success rate setting caused the adaptive

algorithm to deteriorate into a static one with unimpressive performance; hence, there is a potential need for adaptability in the success rate as well. Second, we will revisit the acceptance criterion: it proved to be beneficial to consider nondominating while approaching the Pareto frontier, but nondominating solutions can also trick the mechanism into large mutation steps that are detrimental when attempting to discover the entire Pareto frontier. Third, we plan to extend the study to $n > 2$, other classes of test problems and presently used multi-objective EAs.

References

[1] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart (DE): Frommann-Holzboog, 1973.

[2] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Basel (CH): Birkhäuser, 1977. DOI: 10.1007/978-3-0348-5927-1.

[3] Günter Rudolph. "An Evolutionary Algorithm for Integer Programming". In: *Parallel Problem Solving From Nature (PPSN III)*. Ed. by Y. Davidor, H.-P. Schwefel, and R. Männer. Springer, 1994, pp. 139–148. DOI: 10.1007/3-540-58484-6_258.

[4] A.E. Eiben, R. Hinterding, and Z. Michalewicz. "Parameter control in evolutionary algorithms". In: *IEEE Transactions on Evolutionary Computation* 3.2 (1999), pp. 124–141. DOI: 10.1109/4235.771166.

[5] H.A. Abbass. "The self-adaptive Pareto differential evolution algorithm". In: *Congress on Evolutionary Computation (CEC)*. Vol. 1. 2002, 831–836 vol.1. DOI: 10.1109/CEC.2002.1007033.

[6] Marco Laumanns et al. "Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem". In: *Parallel Problem Solving from Nature (PPSN VII)*. Springer, 2002, pp. 44–53. DOI: 10.1007/3-540-45712-7_5.

[7] Oliver Giel. "Expected runtimes of a simple multi-objective evolutionary algorithm". In: *IEEE Congress on Evolutionary Computation, (CEC)*. IEEE, 2003, pp. 1918–1925. DOI: 10.1109/CEC.2003.1299908.

[8] Seidu Inusaha and Tomasz J. Kozubowski. "A discrete analogue of the Laplace distribution". In: *Journal of Statistical Planning and Inference* 136.3 (2006), pp. 1090–1102. DOI: 10.1016/j.jspi.2004.08.014.

[9] Christian Igel, Nikolaus Hansen, and Stefan Roth. "Covariance Matrix Adaptation for Multi-objective Optimization". In: *Evolutionary Computation* 15.1 (2007), pp. 1–28. DOI: 10.1162/evco.2007.15.1.1.

[10] Silja Meyer-Nieberg and Hans-Georg Beyer. "Self-Adaptation in Evolutionary Algorithms". In: *Parameter Setting in Evolutionary Algorithms*. Springer, 2007, pp. 47–75. DOI: 10.1007/978-3-540-69432-8_3.

[11] Karin Zielinski and Rainer Laur. "Differential evolution with adaptive parameter setting for multi-objective optimization". In: *IEEE Congress on Evolutionary Computation (CEC)*. 2007, pp. 3585–3592. DOI: 10.1109/CEC.2007.4424937.

[12] Weiyi Qian and Ajun Li. "Adaptive differential evolution algorithm for multiobjective optimization problems". In: *Applied Mathematics and Computation* 201.1 (2008), pp. 431–440. DOI: https://doi.org/10.1016/j.amc.2007.12.052.

[13] Ke Li et al. "Adaptive Operator Selection With Bandits for a Multiobjective Evolutionary Algorithm Based on Decomposition". In: *IEEE Transactions on Evolutionary Computation* 18.1 (2014), pp. 114–130. DOI: 10.1109/TEVC.2013.2239648.

[14] Shelvin Chand and Markus Wagner. "Evolutionary many-objective optimization: A quick-start guide". In: *Surveys in Operations Research and Management Science* 20.2 (2015), pp. 35–42. DOI: https://doi.org/10.1016/j.sorms.2015.08.001.

[15] Bingdong Li et al. "Many-Objective Evolutionary Algorithms: A Survey". In: *ACM Computing Surveys* 48.1 (2015). DOI: 10.1145/2792984.

[16] Fran Sérgio Lobato and Valder Steffen Jr. *Multi-objective optimization problems: concepts and self-adaptive parameters with mathematical and engineering applications*. Springer, 2017. DOI: 10.1007/978-3-319-58565-9.

[17] Simon Wessing et al. "Toward Step-Size Adaptation in Evolutionary Multiobjective Optimization". In: *Evolutionary Multi-Criterion Optimization (EMO)*. Ed. by Heike Trautmann et al. Springer, 2017, pp. 670–684.

[18] Benjamin Doerr, Carola Doerr, and Timo Kötzing. "Static and self-adjusting mutation strengths for multi-valued decision variables". In: *Algorithmica* 80.5 (2018), pp. 1732–1768. DOI: 10.1007/s00453-017-0341-1.

[19] Carola Doerr and Markus Wagner. "Sensitivity of Parameter Control Mechanisms with Respect to Their Initialization". In: *Parallel Problem Solving from Nature (PPSN XV)*. Ed. by Anne Auger et al. Springer, 2018, pp. 360–372. DOI: 978-3-319-99259-4_29.

[20] Carola Doerr and Markus Wagner. "Simple on-the-fly parameter selection mechanisms for two classical discrete black-box optimization benchmark problems". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2018, pp. 943–950. DOI: 10.1145/3205455.3205560.

[21] Günter Rudolph. "Runtime Analysis of (1+1)-EA on a Biobjective Test Function in Unbounded Integer Search Space". In: *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2023, pp. 1380–1385. DOI: 10.1109/SSCI52147.2023.10371816.

[22] Günter Rudolph and Markus Wagner. *Towards Adaptation in Multiobjective Evolutionary Algorithms for Integer Problems (Dataset)*. Zenodo, May 2024. DOI: 10.5281/zenodo.11096148. URL: https://doi.org/10.5281/zenodo.11096148.