Differentiable breeding: Automatic differentiation enables efficient gradient-based optimization of breeding strategies

Short title: Differentiable breeding: gradient-based optimization of breeding

Kosuke Hamazaki[1*], Hiroyoshi Iwata[2], and Koji Tsuda[1,3*]

[1] Molecular Informatics Team, RIKEN Center for Advanced Intelligence Project (AIP), RIKEN, 178-4-4 Wakashiba, Kashiwa, Chiba 277-0871, Japan.

[2] Laboratory of Biometry and Bioinformatics, Department of Agricultural and Environmental Biology, Graduate School of Agricultural and Life Sciences, The University of Tokyo, 1-1-1 Yayoi, Bunkyo-Ku, Tokyo 113-8657, Japan.

[3] Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, The University of Tokyo, 5-1-5 Kashiwa-no-ha, Kashiwa, Chiba 277-8561, Japan.

* Corresponding author

E-mail: kosuke.hamazaki@riken.ac.jp (KH), tsuda@k.u-tokyo.ac.jp (KT)

Abstract

Conventional breeding methods often require extensive time to develop new cultivars, hindering rapid adaptation to global challenges. While genomic selection has accelerated breeding, there remains substantial room for improvement. Recent studies have explored complex decision-making in breeding schemes using various optimization techniques such as black-box optimization. However, these methods are challenged by constraints in simultaneously optimizing multiple parameters necessary for achieving more efficient and flexible optimization. To address these limitations, this study implemented automatic differentiation of breeding schemes using PyTorch. By treating the entire breeding scheme as a differentiable computational graph, we enabled efficient gradient calculations for final genetic gains relative to progeny allocation parameters for each mating pair. We first validated our approaches by comparing automatic differentiation with analytical results in a simple gamete generation test case. Next, we used automatic differentiation to perform gradient-based optimization of progeny allocation strategies, aiming to maximize genetic gains in breeding schemes. The gradient-based strategy was then compared to black-box-based optimized and non-optimized strategies. Our framework successfully reduced the number of function evaluations needed for optimization compared to the black-box-based approach and

outperformed the non-optimized strategy in terms of genetic gains. This demonstrates that our framework effectively harnessed gradient information via automatic differentiation. Integrating automatic differentiation into breeding strategy optimization is expected to enhance flexibility and lay the groundwork for future breeding optimization methods.

Author summary

Plant breeding plays a crucial role in addressing global challenges like population growth and climate change by developing more efficient and adaptable crop varieties. However, conventional breeding methods often require several years to produce new cultivars, making it difficult to keep pace with these challenges. Recent advancements in breeding techniques, such as genomic selection, have significantly enhanced selection accuracy and speed. Despite these improvements, substantial room for optimization remains in real breeding programs. This study explores the application of automatic differentiation to optimize breeding strategies, specifically focusing on the number of progenies allocated to each mating pair. Automatic differentiation enables efficient calculation of derivatives for complex functions, potentially accelerating the optimization process in breeding schemes. We implemented differentiable breeding schemes based on automatic differentiation using PyTorch, treating the entire breeding scheme as a differentiable computational graph. By integrating automatic differentiation into breeding strategy optimization, this approach is expected to enable more efficient exploration of optimal solutions while achieving more flexible optimization with a greater number of parameters. This novel method has the potential to significantly enhance the efficiency and flexibility of breeding optimization, ultimately contributing to the development of

more adaptable and productive crop varieties to address global food challenges.

Introduction

Addressing global challenges such as population growth and climate change through sustainable food production requires the development of more efficient and environmentally adaptable crop varieties is crucial [1]. Breeding plays a vital role in creating new varieties that enhance yield, improve disease resistance, and increase nutritional value [2]. However, conventional breeding methods face a major obstacle: depending on the crop species, they often take a decade or more to produce new cultivars, making it difficult to keep up with urgent global demands. In recent years, breeding has been revolutionalized by technologies that utilize genomic information, such as Genomic Selection (GS) [3]. GS, which identifies superior genotypes based on predicted genotypic values using genome-wide marker data, has greatly improved selection accuracy and speed compared to traditional phenotypic or pedigree-based selection methods [4,5]. These advancements have been confirmed in numerous simulation and empirical studies [6–13]. Nonetheless, in practical breeding programs, conventional GS often remains limited to basic selection using prediction models, leaving considerable potential for further refinement and optimization.

In response to these challenges, recent studies have moved beyond solely enhancing model accuracy [14–16] to address complex decision-making issues in

breeding schemes. For example, various selection criteria have been developed to boost future genetic gains while preserving genetic diversity within breeding populations [17–22]. Moreover, optimization techniques have been increasingly applied to refine breeding strategy decisions. One such method, optimal contribution selection (OCS), maximizes genetic gains while constraining inbreeding by optimizing each individual's contribution to the next generation [23,24]. Although OCS itself is a traditional selection method, it has been adopted for selecting crossing pairs, with its efficiency demonstrated in several GS studies [25,26]. Another approach, look-ahead selection (LAS), optimizes mating descisions through long-term breeding simulations that anticipate future outcome of each pair under simple assumptions [27].

Beyond OCS and LAS, research has explored other optimization techniques, including branch-and-bound method [28], integer programming [29], Bayesian optimization [30,31], and reinforcement learning [32], to determine appropriate breeding strategies and planning. Hamazaki and Iwata [33] advanced this area by optimizing progeny allocation strategies through a combination of future-oriented breeding simulations and black-box optimization. They conceptualized the breeding scheme as a black-box function, with allocation parameters as inputs and the final genetic gains as outputs, and applied a black-box optimization technique called stochastic simultaneous

optimistic optimization (StoSOO) to optimize it [34]. Their approach provided greater flexibility by accommodating any breeding scheme configuration without restrictive assumptions [33]. Despite these advancements, current methods for optimizing breeding strategies continue to face substantial challenges.

One significant challenge in breeding optimization is the limited number of available methods. Breeding schemes inherently involve complex stochastic processes spanning multiple generations, requieing the use of black-box optimization methods—such as genetic algorithms [36], Bayesian optimization [36], or StoSOO [34]—to perform bereeding optimization while accurately evaluating each strategy [38]. While these are powerful optimization tools, constraints on the number of parameters that can be optimized simultaneously limit the ability to optimize all aspects of a breeding scheme at once [39,40]. These challenges have intensified the demand for more efficient and flexible optimization methods that can leverage information beyond mere inputs and outputs.

These challenges extend beyond breeding, as automatic differentiation—a powerful computational technique—has recently garnered significant attention across various fields addressing complex optimization problems [39–42]. Automatic differentiation automatically computes derivatives of complex functions defined in computer programs, enabling rapid gradient calculations that significantly accelerate

optimization processes even in intricate cases. This technique has seen widespread adaptation across disiplines. For instance, in machine learning, the backpropagation process in deep learning employs automatic differentiation to enable efficient neural network training [41,43,44]. Beyond machine learning, automatic differentiation has found applications in diverse scientific domains, including physics [45,46], chemistry [47–50], and biology [51,52], where it helps solve complex computational problems and optimize various processes. This broad adoption has been facilitated by programming frameworks like TensorFlow [53], PyTorch [54], and JAX [55], which simplify implementation. This trend is fostering an environment where traditional simulators are increasingly replaced by differentiable forms.

Applications of automatic differentiation are also emerging in agriculture. For instance, in crop growth modeling, automatic differentiation has been used for sensitivity analysis to improve biomass [56]. In the study, they developed a differentiable crop model with ten agronomic traits as inputs and leaf area index and biomass as outputs, helping identify the most influential input parameters affecting these outputs. While this study highlights the potential of automatic differentiation for complex agricultural optimization problems, no research to date has applied or evaluated this framework specifically within breeding context. Thus, this study aims to investigate the applicability of automatic

differentiation in plant breeding and demonstrate its effectiveness. We focus on optimizing the progeny allocation strategies in breeding schemes based on simple recurrent genomic selection, expanding on the optimization framework proposed by Hamazaki and Iwata [33] to incorporate gradient information derived from the automatic differentiation of breeding schemes. Our approach treats the entire breeding scheme as a differentiable computational graph, allowing for efficient gradient calculations of final genetic gains with respect to each allocation-related parameter. This integration of automatic differentiation into breeding strategy optimization is expected to faciliate efficient exploration of optimal solutions, even with a larger number of parameters.

Materials and Methods

In this study, we aimed to implement automatic differentiation of breeding schemes. Automatic differentiation is a computational technique for efficiently calculating function derivatives [39–42]. Even for complex functions/systems comprising many processes, it can automatically compute their derivatives by utilizing the chain rule. Figure 1 illustrates an example of the automatic differentiation of the genetic variance of gametes against their parent's marker genotype. As shown in Figure 1, by decomposing the function into differentiable steps and linking them via the chain rule, partial derivatives can be automatically calculated in the backward process after the forward simulation of computing the function output.

Figure 1. Example of automatic differentiation of the genetic variance of gametes with respect to their parent's marker genotype. The figure illustrates one example of the automatic differentiation process with the reverse mode in the context of breeding. Here, we considered the process of generating gametes from one parent and computed the derivative of the genetic variance of progenies with respect to the parent's marker genotype via automatic differentiation.

To achieve this automatic differentiation, all breeding simulations were implemented using PyTorch version 2.3.1 [54,57]. PyTorch enables automatic differentiation of breeding schemes in reverse mode, facilitating gradient-based optimization of breeding strategies. We first validated the suitability of automatic differentiation via PyTorch in the context of breeding by comparing it with analytical results from a simple test case evaluating the genetic variance of gametes. Following this, we optimized the progeny allocation strategy to maximize the final genetic gain, as outlined by Hamazaki and Iwata [33]. Unlike the previous approach, we used gradient-based optimization instead of black-box optimization for more efficient computation. Throughout the study, R version 4.4.1 [59] was used to simulate the marker genotype of the initial breeding population, while Python version 3.12.2 [60] was used to conduct breeding simulations and gradient-based optimization.

Simulation of the initial breeding population and QTLs

Simulation of genome structure in the initial breeding population

The marker genotype of the initial population was simulated similarly to

185 Hamazaki and Iwata [33]. As in their study, we assumed true quantitative trait loci (QTL)
186 positions and effects were known for simplicity. In the simulation, only genome-wide
187 QTLs for founder haplotypes were generated using the coalescent simulator GENOME
188 [62]. This study assumed a virtual diploid crop with ten chromosomes ($2n = 20$) and
189 $m^{(\mathrm{QTL})}$ QTLs on each chromosome. Here, we prepared two different scenarios
190 concerning the number of QTLs: $m^{(\mathrm{QTL})} = 3$ (Scenario 1) or $m^{(\mathrm{QTL})} = 50$ (Scenario 2).
191 First, 4,000 founder haplotypes for each chromosome were independently
192 generated using GENOME with the following parameters: "-pop 1 4000 -N 4000 -c 1 -
193 pieces 20000 -rec 0.0001 -s 40 -maf 0.01 -tree 0 -mut 0.00000001" (Scenario 1) or "-pop
194 1 4000 -N 4000 -c 1 -pieces 20000 -rec 0.0001 -s 416 -maf 0.01 -tree 0 -mut 0.00000001"
195 (Scenario 2). Among the $8\left(m^{(\mathrm{QTL})} + 2\right)$ loci, $m^{(\mathrm{QTL})}$ loci whose minor allele frequency
196 (MAF) was equal to or larger than 0.01 were randomly selected as QTLs. Next, 2,000
197 founders were generated by sampling two haplotypes from all founder haplotypes with
198 replacement. These founders were randomly mated for one generation to simulate a
199 historical population of the 2,000 genotypes. These procedures for simulating the
200 historical population are similar to those in the R package "BreedingSchemeLanguage"
201 by Yabe et al. [63].
202 Then, the base breeding population was simulated from this historical population

203 in a similar manner as Müller et al. [21,62]. Concretely, the historical population
204 underwent random mating for 300 generations, followed by a population bottleneck that
205 randomly selected 200 individuals and conducted random mating for 15 generations. This
206 bottleneck was used to create extensive linkage disequilibrium (LD), often observed in
207 elite plant breeding populations [63]. The population was then subject to random mating
208 for three more generations to remove close family relationships, resulting in pre-breeding
209 materials.

210 From these simulated pre-breeding materials, $N^{(\mathrm{pop})(0)} = 250$ genotypes were
211 selected for the initial population of a breeding scheme so that these 250 genotypes
212 represented the simulated genetic resources in terms of genetic diversity. This selection
213 was conducted in the completely same manner as Hamazaki and Iwata [33], i.e., using
214 the k-medoids method on the marker genotypes of the pre-breeding material using the
215 "pam" function of the R package "cluster" version 2.1.2 [64,65]. After clustering the pre-
216 breeding materials into 250 groups, we selected the medoids as the representative
217 genotypes from each group for the initial breeding population. This base breeding
218 population was regarded as generation $t = 0$ in a breeding scheme. Here, the initial
219 population consisted of a relatively small number of genotypes, assuming small-scale
220 breeding schemes where optimizing breeding strategies significantly impacts.

Simulation of QTLs and genotypic values

In this study, we assumed quantitative traits with a simple genetic architecture as target traits, as described below. First, as described in the previous subsection, $m^{(\mathrm{QTL})} = 3$ or $m^{(\mathrm{QTL})} = 50$ QTLs were assumed to have some effects on phenotypes (i.e., a total of $M^{(\mathrm{QTL})} = 30$ or $M^{(\mathrm{QTL})} = 500$ QTLs). QTL effects were then sampled from the normal distribution, as shown in Equation 1.

$$p\left(\boldsymbol{\alpha}^{(\mathrm{QTL})}\right) \sim \mathrm{MVN}\left(\mathbf{0}, \frac{1}{M^{(\mathrm{QTL})}}\mathbf{I}_{M^{(\mathrm{QTL})}}\right), \tag{1}$$

where $\boldsymbol{\alpha}^{(\mathrm{QTL})}$ is an $M^{(\mathrm{QTL})} \times 1$ vector of QTL effects and $\mathbf{I}_{M^{(\mathrm{QTL})}}$ is an $M^{(\mathrm{QTL})} \times M^{(\mathrm{QTL})}$ identity matrix. Here, all QTL effects were assumed to be additive for simplicity. The true additive genotypic values $\mathbf{u}^{(\mathrm{TGV})(0)}$ for the parent panel were then simulated using the following Equation 2.

$$\mathbf{u}^{(\mathrm{TGV})(0)} = \sum_{m}^{M^{(\mathrm{QTL})}} \sum_{l=1}^{2} \mathbf{w}_{\cdot,m,l}^{(0)} \alpha_m \,, \tag{2}$$

where $\mathbf{w}_{\cdot,m,l}^{(0)} = \left[w_{1,m,l}^{(0)}, \dots, w_{n,m,l}^{(0)}, \dots, w_{N^{(\mathrm{pop})(t)},m,l}^{(0)}\right]^{\mathrm{T}}$ is a $N^{(\mathrm{pop})(t)} \times 1$ vector of allele scores at m-th QTL ($1 \leq m \leq M^{(\mathrm{QTL})}$), and $w_{n,m,l}^{(t)} \in \{0,1\}$ is a haplotype score at m-th QTL on maternal/paternal chromosome ($1 \leq l \leq 2$, $l = 1$ and $l = 2$ correspond to maternal and paternal chromosome, respectively, in the case of diploid) of an individual n ($1 \leq n \leq N^{(\mathrm{pop})(t)}$) in generation t. Here, since we used the true marker effects, a

heritability was assumed to be 1 throughout the study.

Differentiate the genetic variance of gametes with respect to the marker genotype of a parent

To verify the validity of automatic differentiation via PyTorch in the context of breeding, we first differentiated the genetic variance of gametes relative to a parent's marker genotype using PyTorch (Figure 1). We then compared this result to the analytical solution. We chose to focus on the genetic variance of gametes because it represents the parent's contribution to progeny when mated with another individual. In this subsection, we will further detail how the marker genotype of gametes is generated and how to compute the analytical solution of the first derivative of genetic variance.

Two methods for generating gametes by PyTorch

Here, we define the recombination rate between two markers m and m' as $r_{m,m'}$. We note that $r_{m,m'} = 0.5$ if two markers m and m' are located on different

chromosomes. These two markers satisfy $1 \leq m, m' \leq M$, where M is the number of total markers, and $M = M^{(\mathrm{QTL})}$ in this study. In the following, we consider a straightforward case of generating $N^{(t+1)}$ gametes from an individual n_1 in generation t and calculate the genetic variance of these $N^{(t+1)}$ gametes in generation $t+1$. In this study, we employed two methods for generating gametes from an individual.

1. A method of considering the occurrence of recombination between adjacent markers one by one (Method 1)

The first method checks whether each recombination occurs between adjacent markers one by one. Thus, since the recombination occurs with a probability of $r_{m-1,m}$ between markers $m-1$ and m, we can formulate a $M \times 1$ vector of allele scores of gamete n in $t+1$ generation, $\mathbf{w}_{n,\cdot}^{(t+1)} = \left[w_{n,1}^{(t+1)}, \ldots, w_{n,m}^{(t+1)}, \ldots, w_{n,M}^{(t+1)}\right]^{\mathrm{T}}$, as shown in Equation 3.

$$\mathbf{w}_{n,\cdot}^{(t+1)} = \left[w_{n_1,1,l_1}^{(t)}, \cdots, w_{n_1,m,l_m}^{(t)}, \cdots, w_{n_1,M,l_M}^{(t)}\right]^{\mathrm{T}}. \quad (3)$$

Here, l_m above is sequently sampled from the following recurrence relation,

Equation 4.

$$l_m = \begin{cases} l_{m-1} & \text{with probability } 1 - r_{m-1,m} \\ 3 - l_{m-1} & \text{with probability } r_{m-1,m} \end{cases}, \tag{4}$$

where we virtually set $l_0 = 1$ and $r_{0,1} = 0.5$.

2. A method of predetermining recombination sites (Method 2)

The second method priorly samples recombination sites based on recombination rates and counts the total number of recombination events until marker m. Then, the method utilizes the fact that each time recombination occurs twice, l_m in Equation 1 returns to the original chromosome l_1. Thus, l_m in Equation 1 can be formulated as in the following Equation 5.

$$l_m = \sum_{m=1}^{M} c_{m-1,m} - 2d_m + 1 \quad s.t.\ l_m \in \{1,2\}, \tag{5}$$

where d_m is an integer that satisfies the condition $l_m \in \{1,2\}$, and $c_{m-1,m}$ is sampled from the following Equation 6.

$$c_{m-1,m} = \begin{cases} 0 & \text{with probability } 1 - r_{m-1,m} \\ 1 & \text{with probability } r_{m-1,m} \end{cases}. \tag{6}$$

In this study, we implemented the above two methods for generating gametes using PyTorch, and computed genetic values of gametes, $\mathbf{u}_{n_1}^{(t+1)}$, by only assuming additive

marker effects $\boldsymbol{\alpha}$ ($\boldsymbol{\alpha} = \boldsymbol{\alpha}^{(\mathrm{QTL})}$ in this study) as in Equation 7.

$$\mathbf{u}_{n_1}^{(t+1)} = \mathbf{W}_{n_1}^{(t+1)}\boldsymbol{\alpha}, \tag{7}$$

where $\mathbf{W}_{n_1}^{(t+1)}$ is a $N^{(t+1)} \times M^{(\mathrm{QTL})}$ matrix of allele scores for gametes generated from one individual n_1. Here, since $\mathbf{u}_{n_1}^{(t+1)}$ represents genetic values expressing gamete contributions rather than genotypic values, making it distinct from $\mathbf{u}^{(\mathrm{TGV})(0)}$ in Equation 2. Finally, we computed the sample genetic variance of $\mathbf{u}_{n_1}^{(t+1)}$, $\sigma_{u_{n_1}}^{(t+1)^2}$, and estimated its derivative with respect to the marker genotype score of the parent, i.e., $\frac{\partial \sigma_{u_{n_1}}^{(t+1)^2}}{\partial w_{n_1,m,1}^{(t)}}$, by using automatic differentiation function of PyTorch [54,57].

Theoretical derivation of the derivative of genetic variance of gametes

On the other hand, the above derivative $\frac{\partial \sigma_{u_{n_1}}^{(t+1)^2}}{\partial w_{n_1,m,1}^{(t)}}$ can be derived theoretically. The details are given in Supplementary Note in Appendix S1, but by considering the genetic variance can be expressed using the covariance matrix between markers, the derivative can be formulated as the following Equation 8.

$$\frac{\partial \sigma_{u_{n_1}}^{(t+1)^2}}{\partial w_{n_1,m,1}^{(t)}} = \frac{1}{2}\sum_{i=1}^{M} \left(1 - 2r_{i,m}\right)\left(w_{n_1,i,1}^{(t)} - w_{n_1,i,2}^{(t)}\right)\alpha_i\alpha_m. \tag{8}$$

Thus, by comparing the derivative estimated by gamete simulations and PyTorch to that computed from Equation 8, we verified the suitability of automatic differentiation via

PyTorch in terms of producing gametes for the next generation. Here, we used one individual in the initial population ($t = 0$) as n_1 and generated $N^{(1)} \in \{100,1000,10000,50000\}$ gametes to compare the results.

Gradient-based optimization of progeny allocation strategy in breeding schemes

Next, we implemented full breeding schemes via PyTorch to optimize the progeny allocation strategy using gradient-based optimization. In the following subsections, we briefly summarize the system design of the progeny allocation optimization framework as outlined by Hamazaki and Iwata [33]. Then, we introduce the breeding schemes assumed in this study, followed by the evaluation of the outcome of breeding schemes. Finally, we explain how to parameterize and optimize the progeny allocation strategy.

System design

As outlined by Hamazaki and Iwata [33], we assumed that breeders proceed with

a breeding scheme according to the progeny allocation strategy proposed by an "AI breeder" (Figure 2). The breeders can obtain a set of parameters representing the optimal allocation strategy, $\{\hat{\mathbf{h}}^{(\tau)}\}_{\tau=0,\ldots,T-1}$, by providing the AI breeder with information on an initial breeding population, such as marker genotype, genetic marker effects, and recombination rates between genetic markers. Here, we briefly describe the framework in which the AI breeder optimizes the strategy for the resource allocation of progenies.

Figure 2. The optimization framework of allocation strategy in the breeding scheme. In our framework, the breeders can obtain a set of parameters representing the optimal allocation strategy, $\{\hat{\mathbf{h}}^{(\tau)}\}_{\tau=0,\ldots,T-1}$, by providing the AI breeder with information on an initial breeding population, as in Hamazaki and Iwata [33]. Here, a breeding simulator is regarded as a function whose input is a set of allocation parameters, and whose output is the final genetic gain. After computing the function derivative by using the automatic differentiation technique with PyTorch, the allocation parameters are optimized by the gradient-based approach such as SGD.

After receiving the intial population information, the AI breeder explores the

optimal allocation strategy by repeatedly performing breeding simulations with different allocation parameters $\{\mathbf{h}^{(\tau)}\}_{\tau=0,\ldots,T-1}$ (Figure 2). At each generation τ in the simulated schemes, $\tau \in \{0, \ldots, T-1\}$ is a current generation number in simulations conducted by the AI breeder and T is a final generation, whereas $t \in \{0, \ldots, T-1\}$ appeared in the subsequent subsections is a current generation number in breeding schemes proceeded by actual breeders), the breeder first selects the parent candidates from the current generation based on the weighted breeding values (WBV) [68,69]. Here, we define $\tau \in \{0, \ldots, T-1\}$ as the current generation number in simulations conducted by the AI breeder, where T is the final generation, whereas $t \in \{0, \ldots, T-1\}$, which appears in subsequent subsections, represents the current generation number in breeding schemes carried out by actual breeders. Then, the breeder determines mating pairs based on the diallel crossing between the selected parent candidates, followed by the automatic determination of the number of progenies allocated to each pair $\mathbf{b}^{(\tau)}$ given the parameter $\mathbf{h}^{(\tau)}$, by utilizing the selection criteria information $\mathbf{\Omega}^{(\tau)}$ computed from the marker genotypes and effects, such as WBV and the expected genetic variance of progenies (GVP) (Figure S1). Here, we assumed that $\mathbf{b}^{(\tau)}$ is determined according to the softmax conversion of the weighted sum of the selection criteria, as shown in Equation 9.

$$\mathbf{b}^{(\tau)} = \left\lfloor N^{(\mathrm{pop})(\tau+1)} \times \frac{\exp\left(\mathbf{\Omega}^{(\tau)}\,\mathbf{h}^{(\tau)}\right)}{\sum_{i=1}^{N^{(\mathrm{pair})(\tau)}} \exp\left(\mathbf{\Omega}^{(\tau)}\,\mathbf{h}^{(\tau)}\right)_i} \right\rfloor, \tag{9}$$

where $N^{(\mathrm{pop})(\tau+1)}$ is the population size in the next generation $\tau + 1$, $N^{(\mathrm{pair})(\tau+1)}$ is the number of mating pairs in the current generation τ, and $\lfloor x \rfloor$ is the floor function of x (i.e., the maximum integer that is equal to or lower than x). The softmax function is used to allocate progenies in proportion to probabilities reflecting the weighted sum of $\mathbf{\Omega}^{(\tau)}$. In this framework, $\mathbf{h}^{(\tau)}$ can be regarded as the weighting parameters that determines the importance of each selection criterion. By determining $\mathbf{b}^{(\tau)}$ based on $\mathbf{h}^{(\tau)}$, the AI breeder can generate progenies for the next generation based on the principle of meiosis. Then, by repeating these selection and mating steps until the final generation, the AI breeder can finally evaluate the performance of the allocation strategy with the weighting parameter $\mathbf{h}^{(\tau)}$. Thus, the main objective of the AI breeder is to estimate the optimal $\left\{\mathbf{h}^{(\tau)}\right\}_{\tau=0,..,T-1}$ that maximizes the final genetic gain via breeding simulations.

This objective of the AI breeder is completely as same as Hamazaki and Iwata [33], but how to optimize the weighting parameters is different in this study. To optimize the weighting parameters, a gradient-based optimization algorithm called stochastic gradient descent (SGD) [65] was used in this study (Figure 2). SGD is an iterative approach for locally optimizing an objective function using gradient information of the function and is widely applied to machine learning techniques including deep learning. To apply SGD to our optimization problem, the AI breeder was required to access gradient

information of the function given a set of weighting parameters. Thus, in this study, full breeding schemes were implemented in PyTorch, which enabled us to get gradient information by utilizing automatic differentiation approaches [54]. Here, the AI breeder performed 100 breeding simulations with one set of the weighting parameters, and the empirical mean of the 100 final genetic gains was regarded as the objective function to be maximized by SGD. The function evaluations are repeated 200 times in SGD while computing gradient information by automatic differentiation, and a set of the optimized parameters $\{\hat{\mathbf{h}}^{(\tau)}\}_{\tau=0,\ldots,T-1}$ was passed from the AI breeder to the real breeders.

In this study, we checked whether the automatic differentiation could properly contribute to the gradient-based optimization via simulation studies as a proof of concept. We prepared two versions of the gradient-based optimized allocation (GORA) strategy depending on the initial weight for GVP and compared them with the non-optimized equal allocation (EQ) and black-box-based optimized (ORA) strategies [33], by conducting 10,000 breeding simulations using the optimized $\{\hat{\mathbf{h}}^{(\tau)}\}_{\tau=0,\ldots,T-1}$ (Figure 3). For simplicity and due to computational limitations, we also assumed true QTL positions and effects were known when computing the WBV and GVP. The problem of using the true marker effects was discussed in Hamazaki and Iwata [33], but in this study, we simplified the problem settings to prove our concept.

Figure 3. Breeding schemes assumed in this study. Similar to Hamazaki and Iwata [33], we assumed small-scale plant breeding programs with recurrent selection. In a scheme, the selection, mating, and allocation steps were repeated until the final generation. Then, the gradient-based optimized allocation strategy was compared with the black-box-based optimized allocation and non-optimized equal allocation strategies. Here, we prepared for the two scenarios with different numbers of QTLs and three selection strategies with different selection intensities.

Breeding schemes

The breeding scheme in this study was almost the same as that in Hamazaki and Iwata [33] (Figure 3). We also assumed small-scale plant breeding programs with recurrent selection to reduce the computational burdens for the simulation studies, but we can flexibly deal with various types of breeding schemes by changing the following simulation settings.

1. Select parent candidates (Figure 3 (1))

Based on the true marker (QTL) effects, we first computed WBVs for individuals in the current generation t ($t \in \{0, \dots, T-1\}$) to select parent candidates as in the following Eq 10 [76].

$$\mathbf{u}^{(\mathrm{WBV})(t)} = \sum_{m=1}^{M^{(\mathrm{QTL})}} \sum_{l=1}^{2} \mathbf{w}_{\cdot,m,l}^{(t)} \alpha_m {p_m^{(t)}}^{-\frac{1}{2}}, \tag{10}$$

where $\mathbf{u}^{(\mathrm{WBV})(t)}$ is an $N^{(\mathrm{pop})(t)} \times 1$ vector of WBV for generation t, $p_m^{(t)}$ is the allele frequency at the m-th marker for generation t, and the other terms are defined in Equation 2. In this study, we assumed that the true marker effect α_m was known. WBV has been expected to maintain genetic diversity by emphasizing rare allele effects and thus avoid the saturation of genetic gains in mid- or long-term breeding programs compared to BV [77].

Then, based on the order of increasing WBV, the $N^{(\mathrm{sel})(t)}$ genotypes were selected as parent candidates for mating. In this study, we implemented the following three selection methods to examine how selection intensity affected the optimization of allocation strategies. Here, these methods remained constant throughout the breeding scheme:

a. SI1: Selected $N^{(\mathrm{sel})(t)} = N^{(\mathrm{sel})} = 15$ genotypes from the current population.

b. SI2: First performed hierarchical clustering based on marker genotypes, creating $N^{(\mathrm{cl})} = 50$ clusters. Then, it selected the only top genotype from each cluster,

resulting in $N^{(\mathrm{sel})(t)} = N^{(\mathrm{sel})} = 25$ genotypes.

c. SI3: Selected $N^{(\mathrm{sel})(t)} = N^{(\mathrm{sel})} = 50$ genotypes from the current population.

2. Determine mating pairs for the next generation (Figure 3 (2))

The selected $N^{(\mathrm{sel})(t)}$ parent candidates were then subjected to diallel crossing, including selfing, to determine the mating pairs for the next generation. This process yielded $N^{(\mathrm{pair})(t)} = \frac{N^{(\mathrm{sel})(t)}(N^{(\mathrm{sel})(t)}+1)}{2}$ total crossing pairs.

3. Allocate progenies to each mating pair (Figure 3 (3))

Three strategies were employed to allocate progenies to each mating pair determined in Step 2, after which a crossing table was created. Here, to account for crossing costs, we implemented a constraint requiring that progeny allocations for crosses be made in minimum units of $N^{(\mathrm{unit})} = 5$ individuals. All strategies produced $N^{(\mathrm{pop})(t+1)}$ progenies in total for the next generation. In this study, we assumed $N^{(\mathrm{pop})(t+1)}$ remained constant over generations, i.e., $N^{(\mathrm{pop})(t+1)} = N^{(\mathrm{pop})} = 250$.

a. The black-box-based optimized resource allocation method (ORA)

The first strategy was the optimized allocation method proposed in Hamazaki and Iwata [33]. This method applied the softmax function to the weighted sum of the selection criteria computed in Step 1. Then, the weights for the criteria were optimized by the black-box optimization method through repeated

breeding simulations. For more details, please refer to the **Progeny allocation strategies for automatic differentiation** subsection.

b. The gradient-based optimized resource allocation method (GORA)

The second strategy was the gradient-based optimized allocation method proposed in this study. We enhanced the first strategy by incorporating the Gumbel-softmax function [19,20], extending its applicability to the automatic differentiation framework. For more details, please refer to the **Progeny allocation strategies for automatic differentiation** subsection.

c. Equal resource allocation method (EQ)

We also implemented an equal allocation method that allocated $N^{(\mathrm{unit})}$ progenies each to $\frac{N^{(\mathrm{pop})(t+1)}}{N^{(\mathrm{unit})}}$ pairs, which were randomly selected from the $N^{(\mathrm{pair})(t)}$ candidate pairs.

4. Generate progenies for the next generation (Figure 3 (4))

Following the crossing table created in Step 3, two gametes were generated for each mating pair based on the second method described in the **Two methods for generating gametes by PyTorch** subsection, accounting for marker recombination. Recombination rates between markers were calculated using the Kosambi map function [21,22] based on the linkage map generated by GENOME. These two

gametes were then combined to form one new progeny, resulting in a new population which consisted (with generation $t+1$) of $N^{(\mathrm{pop})(t+1)}$ progenies.

5. Repeat the parent selection and mating process

Steps 1-4 were repeated until the population reached the final generation, $t = T$. This process was carried out for $T - 1$ generations, where T represents the total number of generations in the breeding program.

Finally, we evaluated the results for each strategy using the true additive genotypic values $\mathbf{u}^{(\mathrm{TGV})(T)}$ of the final population with generation T. In this study, similar to Hamazaki and Iwata [33], we set the final generation as $T = 4$, assuming a scenario where rapid genetic improvement was needed in small-scale breeding schemes. Here, to evaluate the population's maximum for the breeding strategy, we computed $\tilde{u}^{(\mathrm{top})(t)}$, the empirical mean of $\mathbf{u}^{(\mathrm{TGV})(t)}$ for the top $N^{(\mathrm{top})(t)}$ genotypes. These genotypes were selected based on their $\mathbf{u}^{(\mathrm{TGV})(t)}$ values in descending order. We then scaled $\tilde{u}^{(\mathrm{top})(t)}$ based on the initial population's mean and standard deviation, $u^{(\mathrm{mean})(0)}$ and $\sigma_u^{(0)}$, resulting $u^{(\mathrm{top})(t)} = \frac{\tilde{u}^{(\mathrm{top})(t)} - u^{(\mathrm{mean})(0)}}{\sigma_u^{(0)}}$. We chose $N^{(\mathrm{top})(t)} = 5$ to represent the top individuals for new variety development while mitigating stochastic variation between simulations. For each parent panel simulation dataset, we ran 10,000 different breeding schemes per strategy and calculated the empirical mean of $u^{(\mathrm{top})(t)}$ from these results.

We repeated this simulation and evaluation process for 10 replicates of the phenotype simulation, each with different QTL positions and effects.

Progeny allocation strategies for automatic differentiation

In this study, we allocated progenies to each mating pair based on the idea proposed by Hamazaki and Iwata [33]. Prior to the mating step, they determined the number of progenies allocated to each mating pair at generation τ, $\mathbf{b}^{(\tau)}$, all at once, by applying the softmax function to the weighted sum of various "goodness" metrics for each mating pair, as shown in Equation 11 (Figure S1).

$$\mathbf{b}^{(\tau)} = \left\lfloor \frac{N^{(\mathrm{pop})(\tau+1)}}{N^{(\mathrm{unit})}} \times \mathbf{p}^{(\tau)} \right\rfloor \times N^{(\mathrm{unit})}, \tag{11}$$

$$\mathbf{p}^{(\tau)} = \frac{\exp\left(\mathbf{\Omega}^{(\tau)}\,\mathbf{h}^{(\tau)}\right)}{\sum_{i=1}^{N^{(\mathrm{pair})(\tau)}} \exp\left(\mathbf{\Omega}^{(\tau)}\,\mathbf{h}^{(\tau)}\right)_i}, \tag{12}$$

where $\mathbf{b}^{(\tau)}$ is an $N^{(\mathrm{pair})(\tau)} \times 1$ vector representing the number of progenies allocated to each mating pair at generation τ, $\mathbf{\Omega}^{(\tau)}$ is an $N^{(\mathrm{pair})(\tau)} \times Q$ matrix of Q different "goodness" metrics of each mating pair (e.g., WBV for an average progeny or GVP for each pair), $\mathbf{p}^{(\tau)}$ is an $N^{(\mathrm{pair})(\tau)} \times 1$ vector of softmax-transformed probabilities, and the other terms are defined in Equation 9. Equation 11 is almost the same as Equation 9 but incorporates the constraint on the number of minimum allocation units, $N^{(\mathrm{unit})}$. The

softmax function converts a vector of weighted goodness for each mating pair, $\mathbf{\Omega}^{(\tau)}\,\mathbf{h}^{(\tau)}$, into a vector of probabilities, $\mathbf{p}^{(\tau)}$, which determines the proportion of progenies to be allocated to each mating pair. Here, $\mathbf{h}^{(\tau)}$ is a $Q \times 1$ vector that determines the weight given to each selection criterion in the allocation step, thus becoming a parameter to be optimized in this study. Since the breeding scheme can be viewed as a black-box function with the set of weighting parameters $\left\{\mathbf{h}^{(\tau)}\right\}_{\tau=0,\ldots,T-1}$ as input and the final genetic gain $g^{(T)} = u^{(\mathrm{top})(T)} - u^{(\mathrm{top})(0)}$ as output, Hamazaki and Iwata [33] employed a black-box optimization technique called StoSOO to estimate the weighting parameters that maximized $g^{(T)}$.

On the other hand, in this study, we tried to optimize $\left\{\mathbf{h}^{(\tau)}\right\}_{\tau=0,\ldots,T-1}$ by utilizing automatic differentiation and gradient-based optimization. However, directly applying Equation 11 as a progeny allocation strategy hinders the automatic differentiation of breeding simulations because the floor function in Equation 11 is discontinuous and non-differentiable. To address this issue, in breeding simulations conducted by the AI breeder, instead of pre-determining the number of progenies for each mating pair, we sequentially assigned each progeny to a mating pair based on softmax-transformed probabilities, $\mathbf{p}^{(\tau)}$. This approach approximates the original allocation strategy in Equation 11 while allowing for automatic differentiation. Here, we also utilized the Gumbel-Softmax function so that

the softmax transformation is differentiable [19,20] (Figures S1 and S2). Specifically, for each progeny, we sample an $N^{(\mathrm{pair})(\tau)} \times 1$ vector, $\mathbf{q}^{(\tau)}$, from the following Gumbel-Softmax function in Equation 13.

$$\mathbf{q}^{(\tau)} = \lim_{c \to 0} \frac{\exp\left(\frac{\mathbf{\Omega}^{(\tau)}\,\mathbf{h}^{(\tau)} + \mathbf{r}^{(\tau)}}{c}\right)}{\sum_{i=1}^{N^{(\mathrm{pair})(\tau)}} \exp\left(\frac{\mathbf{\Omega}^{(\tau)}\,\mathbf{h}^{(\tau)} + \mathbf{r}^{(\tau)}}{c}\right)_i}. \tag{13}$$

Here, $\mathbf{r}^{(\tau)}$ is an $N^{(\mathrm{pair})(\tau)} \times 1$ vector sampled from the Gumbel distribution, c corresponds to the temperature parameter, and the other terms are defined as in Equation 12. As c approaches 0 as in Equation 13, $\mathbf{q}^{(\tau)}$ becomes a one-hot encoded vector—a vector where a single element is 1 and all others are 0, reflecting the softmax-transformed probabilities $\mathbf{p}^{(\tau)}$. Consequently, we select the mating pair corresponding to the element in $\mathbf{q}^{(\tau)}$ that equals 1 for each progeny. This procedure allows us to conduct differentiable breeding simulations with respect to $\mathbf{h}^{(\tau)}$.

Gradient-based optimization of allocation strategies and comparison with the other strategies

In this study, we leveraged gradient information obtained through automatic differentiation of breeding simulations implemented in PyTorch for gradient-based optimization methods (Figure S2). Among various gradient techniques, we employed

Stochastic Gradient Descent (SGD) [18] with an initial learning rate of 0.15 to optimize the allocation strategy. To ensure proper control of the learning rate in the latter stages of optimization, we applied a learning rate scheduler. Using PyTorch's torch.optim.lr_scheduler.StepLR function [6], we decreased the learning rate by a factor of 0.8 every 10 epochs. For each parameter set evaluation, we conducted 100 breeding simulations to obtain gradient information. Here, we defined the domain of definition using the inverse logit transformation as shown in Equation 14.

$$\mathbf{h}^{(\tau)} = (h_{\max} - h_{\min}) \times \frac{\exp\left(\mathbf{z}^{(\tau)}\right)}{1 + \exp(\mathbf{z}^{(\tau)})} + h_{\min}, \tag{14}$$

where $h_{\max} = 3.5$ and $h_{\min} = -0.5$ are the minimum and maximum values of $\mathbf{h}^{(\tau)}$, respectively, and $\mathbf{z}^{(\tau)} \in (-\infty, \infty)$ is logit-transformed weighting parameter, serving as the actual input to the function and the target of optimization. Then, we set 1.5 as the initial parameter for each element of $\left\{\mathbf{h}^{(\tau)}\right\}_{\tau=0,\ldots,T-1}$ (GORA1). To assess the impact of initial values on our results, we included a strategy where the initial weight for GVP was set to 0 (GORA2). We then repeated these function evaluations 200 times, updating parameters via SGD. The final parameter set after these 200 epochs was considered the optimal set of parameters.

In this study, we compared the gradient-based optimized resource allocation strategy (GORA) with two other approaches: the black-box optimization using the

StoSOO algorithm [34] by Hamazaki and Iwata [33] (ORA) and the non-optimized equal allocation strategy (EQ). Here, after the gradient-based optimization of the weighting parameter $\{\mathbf{h}^{(\tau)}\}_{\tau=0,\ldots,T-1}$ assuming the approximated breeding scheme based on the **Progenies allocation strategies for automatic differentiation** subsection, the GORA approach was also evaluated under the constraint with regard to the minimum number of allocated progenies to each pair to compare GORA on an equal footing with the other strategies.

Genetic variance of progenies to consider diversity for allocation strategies

As the candidates for "goodness" to compute $\mathbf{b}^{(\tau)}$ in the allocation step, we used a selection criterion called WBV and the expected GVP for each mating pair (Figure S1). The WBVs of the parent candidates were computed using Equation 10, and the mean of the two parents for each mating pair was used as one of the column vectors of $\mathbf{\Omega}^{(\tau)}$. This vector of WBV was the same as that for the average progeny of each mating pair in this study because we only assumed additive QTL effects.

On the other hand, when computing the GVP, we focused on the genetic diversity

of gametes resulting from the self-fertilization of progeny from each mating pair so that we could consider the genetic distance between the two parents. Here, this GVP can be theoretically derived as in Equation 15, based on the idea proposed by Allier et al. [24–26]:

$$\sigma_n^{(\tau)^2} = \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{C}_n^{(\tau)} \boldsymbol{\alpha}, \tag{15}$$

where $\sigma_n^{(\tau)^2}$ is an expected GVP for mating pair n with generation τ, $\boldsymbol{\alpha}$ is a vector of the true marker effects, $\mathbf{C}_n^{(\tau)}$ is a $M \times M$ covariance matrix between markers caused by the segregation of progeny derived from mating pair n, respectively. The detailed derivation of the covariance matrix $\mathbf{C}_n^{(\tau)}$ is provided in Allier et al. [25]. In this study, after computing $\sigma_n^{(\tau)^2}$ for all mating pairs, a vector of the expected GVPs was used as one of the column vectors of $\boldsymbol{\Omega}^{(\tau)}$.

Results

Differentiate the genetic variance of gametes with respect to the marker genotype of a parent

First, to validate the automatic differentiation in the breeding context, we used the gamete generation process as an example. We calculated the derivatives of the genetic variance of gametes with respect to a parent's marker genotype using two distinct PyTorch-based methods of gamete generation. We then compared these results with the theoretically derived solutions (Figure 4 and Figures S3 and S4).

Figure 4. Comparison between automatic differentiation and analytical solutions for the first-order derivatives of the genetic variance of 1,000 gametes with respect to the parent's marker genotype. Scatter plots of the first-order derivatives of the genetic variance of 1,000 gametes with respect to the parent's marker genotype. The horizontal and vertical axes represent the results derived from the automatic differentiation and analytical solutions. The red line indicates where the automatic differentiation results perfectly match the analytical solutions. (A) Method 1. (B) Method 2.

The results showed that both gamete generation methods highly correlated ($r >$

0.85) with the analytical solutions in all cases. These correlations strengthened as the number of gametes increased, indicating that automatic differentiation performed well in the breeding context (Figure 4 and Figure S3). Furthermore, comparing the two gamete generation methods revealed that they converged almost perfectly as the number of gametes grew (Figure S4). Given that Method 2 is significantly more computationally efficient than Method 1, we employed it for gamete generation in subsequent analyses.

Evaluation of convergence conditions of the gradient-based optimization algorithm

Having confirmed that automatic differentiation functioned properly for gamete generation, we proceeded to optimize the progeny allocation strategy. We first verified whether the gradient-based optimization algorithm with automatic differentiation could successfully optimize the allocation strategy by monitoring changes in function values across two scenarios under three selection strategies (Figure S5).

The results showed that function values generally improved as the number of SGD algorithm epochs increased, with the exception of Scenario 1 under SI1 (Figure S5).

Notably, allocation strategies under SI2 and SI3 were optimized effectively, with the algorithm converging to certain levels within 100 epochs (Figure S5C-F). These findings indicate that the gradient information derived from automatic differentiation facilitated a more efficient search for optimal parameters.

Genetic gains over four generations

We compared four allocation strategies: two gradient-based optimized (GORA1 and GORA2), black-box-based optimized (ORA), and equal (EQ) allocation strategies. We plotted the change in genetic gains, $g^{(t)}$, for each generation across two scenarios under three selection strategies (Figure 5). Here, the two GORAs were obtained after 200 epochs, and the ORA was obtained after 20,000 function evaluations. These strategies were compared based on 10,000 breeding simulation results, using one replication for the quantitative phenotype simulation.

Figure 5. Change in the expected genetic gains over four generations under different selection intensities. The horizontal and vertical axes represent the number of

generations and the genetic gains, respectively. We compared the following four allocation strategies under the different selection intensities (SI1 – SI3): GORA1: gradient-based optimized allocation with larger initial weights on GVP (light purple dash-dotted), GORA2: gradient-based optimized allocation with smaller initial weights on GVP (purple dashed), ORA: black-box-based optimized allocation (blue solid), EQ: equal allocation (black dotted). (A) Scenario 1. (B) Scenario 2.

In the final generation, $t = 4$, the ORA strategy demonstrated the highest genetic gain for both scenarios (Figure 5). Under strong selection intensity (SI1), there was minimal difference between the proposed and equal allocation strategies. However, under weaker intensities (SI2 and SI3), the proposed GORA strategies outperformed the non-optimized EQ strategy in both scenarios. The effectiveness of optimized allocation (ORA and GORA) was particularly pronounced when the trait was controlled by many QTLs, as in Scenario 2. Comparing the two proposed strategies, GORA2—with smaller initial weights on GVP—generally outperformed GORA1, which had larger initial weights on GVP. This suggests that initial values significantly influenced the gradient-based optimization.

For Scenario 2, the ORA strategy initially showed lower genetic gains than the GORA strategies in the first generation ($t = 1$). However, from the second generation ($t = 2$) onward, ORA overtook the GORAs. This indicates that the black-box-based strategy successfully fine-tuned the balance between the weights on WBV and GVP over time.

Genetic gains across different simulation repetitions

To better understand the nuance of each strategy, we analyzed the cumulative distribution functions (CDFs) of genetic gain in the final generation ($t = 4$) for each allocation strategy across two scenarios under three selection strategies, using 10,000 simulation repetitions. Figure S6 displays these results, with the horizontal axis showing the final genetic gain of an individual and the vertical axis indicating the percentile of simulation repetitions. The 1st and 99th percentiles represent the worst and best performances among the 10,000 repetitions, respectively. The effectiveness of each strategy is demonstrated by its CDF curve shifting rightward and downward on the graph—the further this shift, the better the strategy's overall performance.

Once again, the ORA strategy outperformed all others across all cases (Figure S6). Among the proposed strategies, GORA2 consistently surpassed GORA1 and also outperformed EQ under low selection intensities (SI2 and SI3). In Scenario 2, the CDF curves of the optimized strategies (GORA and ORA) exhibited steeper slopes compared to Scenario 1, suggesting that a larger number of QTLs led to greater variations among simulation repetitions. Moreover, particularly in Scenario 1, the GORA2 strategy displayed steeper slopes than ORA, indicating variable performance: at its best, it matched the performance of ORA, but at its worst, it produced significantly inferior results.

Genetic gains in the final generation using ten replications of the phenotype simulation

We further evaluated the genetic gains in the final generation ($t = 4$) using ten replications of the phenotype simulation (Figure 6). This was done to verify that the gradient-based optimized strategies outperformed the equal allocation strategy for traits with varying QTL positions and effects. When simulating phenotypes for different traits,

701 we altered QTL positions and their effects while keeping the number of QTLs and the
702 distribution of QTL effects constant. We assumed heritabilities of 1 for all ten replications,
703 as we used the true marker effects for optimization. Here, we chose the GORA2 strategy
704 to represent our proposed optimization approach, as it consistently outperformed GORA1
705 across various scenarios and selection strategies. Also, to reduce computational time
706 when evaluating ten phenotype simulation replications, we determined the GORA
707 strategy after 100 function evaluations and the ORA strategy after 5,000. We chose these
708 numbers to ensure the function value reached a satisfactory level without full convergence,
709 based on our earlier assessment of convergence conditions for the SGD algorithm. Full
710 convergence was not necessary here, as our goal was simply to demonstrate that
711 optimization algorithms could improve genetic gain compared to non-optimized
712 strategies, even if only to some extent.
713
714 **Figure 6. Genetic gains in the final generation using ten replications for the**
715 **phenotype simulation under different selection intensities.** Boxplots of the genetic
716 gains in the final generation using ten replications for the phenotype simulation. The
717 horizontal and vertical axes represent the different allocation strategies and the final
718 genetic gains, respectively. We compared the following three allocation strategies under

the different selection intensities (SI1 – SI3): GORA2: gradient-based optimized allocation with smaller initial weights on GVP (light purple), ORA: black-box-based optimized allocation (light blue), EQ: equal allocation (grey). The weighting parameters were optimized based on 100 and 5,000 function evaluations for GORA2 and ORA, respectively. (A) Scenario 1. (B) Scenario 2.

The ORA strategy consistently demonstrated the highest genetic gains across all cases, even when QTL positions and effects of the target trait were altered (Figure 6). Also, the GORA strategy outperformed the EQ strategy under low selection intensities (SI2 and SI3) while showing little difference under high intensities (SI1). Once again in these results, the performance gap between optimized (ORA and GORA) and non-optimized (EQ) strategies was more pronounced under SI2 and SI3 in Scenario 2.

We evaluated the improvement rate in final genetic gain for each optimized strategy compared to the equal allocation strategy using ten phenotype simulation replications. Results showed a similar trend between GORA and ORA strategies (Figure S7). Notably, the GORA strategy failed to optimize and underperformed the EQ strategy in only one replication for Scenario 1. In all other replications, the genetic gains of GORA

were either comparable to or surpassed those of EQ, with remarkable improvements of 15.5–29.8% over EQ under SI2 and SI3 within just four generations.

Genetic diversity over four generations

Next, we compared the GORA, ORA, and EQ strategies in genetic diversity across generations for two scenarios under three selection strategies (Figure 7). The genetic diversity of a breeding population in each generation was evaluated based on the genetic variance of the true genotypic values in the population.

Figure 7. Change in the genetic variances over four generations under different selection intensities. The horizontal and vertical axes represent the number of generations and the genetic variances in the breeding population, respectively. We compared the four allocation strategies under the different selection intensities (SI1 – SI3). The abbreviations of the allocation strategies are the same as those of Figure 5. (A) Scenario 1. (B) Scenario 2.

First, in Scenario 1, the GORA2 and ORA showed the rapid reduction of the genetic diversity, while the GORA1 relatively maintained the genetic variance throughout the scheme under all selection intensities (Figure 7A). In Scenario 2, the loss of diversity was suppressed compared to that in Scenario 1, and the GORA1 showed the largest decrease in genetic diversity (Figure 7B). Also, the genetic diversity of GORA1 in the final generation was equal to or larger than that in the initial generation. These results suggest that the initial weights on GVP in the allocation step largely impacted the genetic diversity for the proposed GORA strategies.

Optimized weighting parameters for each strategy

After 200 function evaluations for the GORAs and 20,000 for the ORA, we obtained optimized weighting parameters $\{\mathbf{h}^{(\tau)}\}_{\tau=0,1,2,3}$ for two scenarios under three selection strategies (Table 1 and Tables S1 – S5). Here, a higher value of $\mathbf{h}^{(\tau)}$ indicates greater emphasis on the corresponding criterion.

Table 1. Optimized weighting parameters $\mathbf{h}^{(\tau)}$ across generations $\boldsymbol{\tau = 0, \ldots, T-1}$ for Scenario 1 under SI1.

Scenario 1, SI1	GORA1		GORA2		ORA	
Generation (τ)	**WBV**	**GVP**	**WBV**	**GVP**	**WBV**	**GVP**
0	2.63	0.38	2.93	0.20	1.50	0.17
1	2.09	1.92	2.13	0.29	1.50	0.17
2	2.26	1.93	2.33	0.04	2.83	0.17
3	2.22	1.99	2.39	−0.09	2.83	0.17

Examining the weighting parameters for Scenario 1 under SI1 (Table 1), we find that GORA1 and GORA2 assigned higher weights to WBV than ORA in earlier generations. This indicates that the GORA strategies prioritized the initial generation. Regarding the weights for GVP, GORA1 showed higher values than GORA2 and ORA, likely due to its larger initial values. For Scenario 2, we observed similar patterns under SI2 and SI3, with generally higher weights compared to SI1 (Tables SI1 and SI2). Additionally, under SI2 and SI3, the GORA strategies tended to emphasize GVP more in the initial generation than under SI1.

In Scenario 2, the comparison between GORA and ORA strategies revealed trends similar to Scenario 1 (Tables S3 – S5). However, the optimized weights for Scenario 2 were generally lower than those for Scenario 1 under identical selection strategies. Interestingly, the trend of prioritizing the initial generation—observed in GORA

strategies for Scenario 1—was absent in Scenario 2. Conversely, the ORA strategies consistently placed greater importance on the final generation across all selection strategies. Moreover, the ORA strategy displayed varying, non-uniform weights for GVP in each generation, distinguishing it from the GORA strategies.

Optical flow for the gradients calculated by automatic differentiation

Finally, to further validate the GORA strategies, we examined the global relationship between function values and gradients calculated by automatic differentiation. We created visualizations similar to optical flow diagrams—a technique used in computer vision to represent object motion— [73] for two parameters. We performed grid-based calculations of function values and gradients for two distinct breeding schemes based on two allocation parameters: (1) a scheme with one generation using both WBV and GVP, and (2) a scheme with two generations using only WBV for allocation under two scenarios for three selection strategies (Figure 8 for SI2 and Figures S8 for SI1 and S9 for SI3). We defined the domains of logit-transformed weighting parameters $\mathbf{z}^{(\tau)}$ as $z^{(\tau)} \in [-3, 2]$ ($h^{(\tau)} \in [-0.31, 3.02]$) for WBV and $z^{(\tau)} \in [-0.47, 1.5]$ for GVP. We then divided each domain into 50 areas and evaluated the

function values and gradients for $51^2 = 2{,}601$ parameter sets in total. In these visualizations, the background color indicates the magnitude of function values for each set of logit-transformed weighting parameters. The arrows, meanwhile, represent both the magnitude and direction of gradients calculated through automatic differentiation.

Figure 8. Optical flow for the gradients calculated by automatic differentiation under SI2. The function values and gradients were evaluated by automatic differentiation via grid search under SI2. The horizontal and vertical axes represent the logit-transformed weighting parameters $\mathbf{z}^{(\tau)}$ in the allocation step. The background color indicates the magnitude of function values for each set of logit-transformed weighting parameters, whereas the arrows represent both the magnitude and direction of gradients calculated through automatic differentiation. The blue and green points correspond to the initial and optimized values by the GORA2 approach. (A), (B) The gradients were evaluated for the breeding scheme with only one generation considering both WBV and GVP as candidates of $\mathbf{\Omega}^{(\tau)}$. The horizontal and vertical axes represent the logit-transformed weighting parameters for WBV and GVP in generation 0, respectively. (C), (D) The gradients were evaluated for the breeding scheme with two generations considering only WBV as candidates of $\mathbf{\Omega}^{(\tau)}$. The horizontal and vertical axes represent the logit-transformed

weighting parameters for WBV in generations 0 and 1, respectively. (A), (C) Scenario 1. (B), (D) Scenario 2.

Examining the results for the scheme with one generation using both WBV and GVP in Scenario 2, we observed that function values increased across all selection strategies as the weight for WBV grew larger and the weight for GVP diminished (Figure 8B and Figures S8B and S9B). Also, the results for the scheme with two generations using only WBV in Scenario 2 showed that function values increased across all selection strategies as the weights for WBV in both generations got larger (Figure 8D and Figures S8D and S9D). In both cases, the arrows representing gradient information pointed in the direction of increasing function values, confirming that the gradient information was accurately calculated.

For the scheme with two generations using only WBV in Scenario 1, under SI3, we observed a trend similar to Scenario 2 (Figure S9C). However, under SI1 and SI2, the highest function values occurred when the weight for WBV was low to moderate in generation 0 and high in generation 1 (Figures 8C and S8C). These findings suggest a strong relationship between the optimal weighting parameters in the allocation step and

both the number of QTLs controlling the trait and the selection intensity. Near the function's maximum value, gradient arrows either pointed towards the peak or became negligible in length, ensuring that optimization wouldn't stray from the optimal point.

The scheme with one generation using both WBV and GVP in Scenario 1 revealed that optimal parameters varied significantly across selection strategies (Figures 8A, S8A, and S9A). Notably, under SI2, a region of high function values seemed to extend along the diagonal line where x equals y (Figure 8A). In these instances, arrows primarily pointed rightward, sometimes failing to capture subtle changes in function values accurately.

The figures depict initial and optimized weighting parameters as blue and green points, respectively. In most cases, the SGD optimization search range was narrower than the grid search area in this subsection, suggesting it may not have reached the true optimal solution. Although this section assumes a shorter number of generations, these insights likely apply to breeding schemes with four generations as well, potentially explaining why GORA strategies showed slightly lower genetic gains than ORA.

Discussion

In this study, we calculated gradient information using PyTorch-based implementation of breeding simulations and automatic differentiation. We then leveraged this gradient information to optimize progeny allocation in breeding schemes through gradient-based methods.

First, to validate the accuracy of gradient information calculated through our automatic differentiation framework, we first examined a simple case: computing partial derivatives of the genetic variance of gametes (Figure 1). Our results demonstrated that the gradients calculated through automatic differentiation closely matched the theoretically derived results for both gamete generation methods commonly employed in general breeding simulations (Figure 4 and Figures S4 and S5). Additionally, we explored the relationship between function values and gradients across various parameter sets using grid search in breeding schemes spanning one or two generations (Figure 8 and Figures S8 and S9). The visualizations similar to optical flow diagrams [73] showed that the arrows depicting gradients generally pointed towards regions of increasing function values. While capturing accurate gradients proved challenging for scenarios involving traits controlled by a small number of QTLs, our automatic differentiation framework performed effectively for scenarios with traits governed by numerous QTLs. The number of QTLs influenced the continuity of genotypic values, which likely affected the precision

of automatic differentiation. In any case, based on these results, we can conclude that gradient estimation through automatic differentiation achieved a satisfactory level of accuracy.

Having confirmed the validity of automatic differentiation, we then attempted to optimize the progeny allocation strategy in breeding schemes using gradient-based optimization methods. In this study, we employed a straightforward SGD algorithm with a scheduler for our gradient-based optimization method. Monitoring the function values across epochs revealed an increasing trend for most of the scenarios and selection intensities, with convergence typically occurring after about 100 iterations (Figure S5). These results demonstrate that our gradient-based method, utilizing automatic differentiation, significantly appeared to reduce the number of breeding simulations required for optimization compared to the StoSOO-based optimization proposed by Hamazaki and Iwata [33]. However, it is important to note a key difference: our automatic differentiation-based optimization used gradient descent, a local optimization method, while the previous study employed StoSOO for global optimization. These approaches are fundamentally distinct, making direct comparisons of computational complexity inappropriate. In fact, despite the time-consuming optimization process, the optimized function values by ORA generally surpassed those of GORA (Figures 5 and 6 and Figures

S6 and S7). Thus, the main focus of this study should be on how automatic differentiation enabled more efficient and appropriate parameter exploration.

Using the optimized weighting parameters by SGD, we evaluated the performance of the proposed method based on the genetic gains (Figures 5 and 6 and Figures S6 and S7). As a result, when the selection intensity in the selection stage was high, the optimized allocation strategies were unable to demonstrate clearly superior performance compared to non-optimized strategies. While this result appeared discouraging at first glance, it was actually quite logical, i.e., the optimization of allocation strategies primarily functions to enhance selection intensity, so when the selection stage already employs high selection intensity, the optimization offers a limited additional benefit. On the other hand, particularly when the selection intensity was low, our proposed gradient-based optimization method demonstrated performance significantly superior to the non-optimized case, albeit slightly inferior to StoSOO-based optimization. This result also suggests that our automatic differentiation framework was functioning properly. Here, the reason why the gradient-based method yielded slightly less favorable results compared to the black-box-based method may be attributed to its limited range of exploration in the optimization process. As evident from the optical flow diagrams (Figure 8 and Figures S8 and S9), even in breeding schemes spanning just one or two generations, the exploration

range of SGD failed to encompass the entire global area, and consequently, the SGD could not have reached the global optimum. One possible reason for the narrow search range of SGD is that the SGD used in this study employed a scheduler, which might have prematurely halted the search process. Therefore, employing an alternative optimization method that can automatically adjust the learning rate, such as Adam [77], without using a scheduler, could resolve this issue and potentially enhance the final genetic gains achieved through gradient-based optimization.

Next, we discuss the impact of different initial values on the performance of the gradient-based methods. Analysis of genetic gains revealed that the gradient-based method with smaller initial values for GVP consistently outperformed the approach using larger initial GVP values (Figure 5 and Figure S6). Additionally, the results of genetic variance demonstrated that larger initial GVP values led to greater genetic diversity in breeding populations in all cases (Figure 7). These initial values significantly influenced the final optimized parameters (Table 1 and Tables S1 – S5), likely explaining the observed variations in both genetic gains and diversity. These results indicate that the initial values had a significant impact on the performance of gradient-based methods, even when applied to the optimization of breeding strategies. Therefore, employing global optimization methods, such as Bayesian optimization [36], to determine suitable initial

values would enhance the effectiveness of our gradient-based method with automatic differentiation.

Finally, the potential applications of automatic differentiation extend far beyond mere gradient-based methods. In particular, black-box optimization methods that utilize derivative information—such as Bayesian optimization [5,6] or genetic algorithm [7] with derivatives—show promise in accelerating optimization processes and enhancing their efficiency. Thus, by applying automatic differentiation to these derivative-oriented optimization methods, we can expect further improvements in genetic gains while reducing computational costs, surpassing the capabilities of our current gradient-based and black-box-based optimization frameworks. Moreover, the gradient information obtained through automatic differentiation can be incorporated into deep learning techniques, such as deep neural networks [8]. By integrating deep neural network structures into the optimization of breeding schemes using automatic differentiation, we could potentially create a more flexible optimization framework that does not rely on human-proposed selection criteria and can handle a larger number of parameters. The automatic differentiation framework developed in this study is expected to serve as a foundation for more sophisticated optimization methods in the future. Therefore, establishing the automatic differentiation of breeding schemes and demonstrating its

potential for optimization using a simple gradient method in this study is of extremely great significance. This research will likely be an important step for future decision-making in breeding strategies and their optimization.

Acknowledgments

This work was supported by JST, ACT-X Grant Number JPMJAX23CL, Japan. The computations in this work were performed in the supercomputer centers at RAIDEN. We would like to thank Dr. Masato Sumita for his assistance in setting up the computational environment such as RAIDEN.

965 # References

966 1. Capper JL. Replacing rose-tinted spectacles with a high-powered microscope: The
967 historical versus modern carbon footprint of animal agriculture. Anim Front. 2011;1:
968 26–32.
969 2. Eggen A. The development and application of genomic selection as a new breeding
970 paradigm. Anim Front. 2012;2: 10–15.
971 3. Meuwissen TH, Hayes BJ, Goddard ME. Prediction of total genetic value using
972 genome-wide dense marker maps. Genetics. 2001;157: 1819–1829.
973 4. Jannink J-L, Lorenz AJ, Iwata H. Genomic selection in plant breeding: from theory
974 to practice. Brief Funct Genomics. 2010;9: 166–177.
975 5. Crossa J, Pérez-Rodríguez P, Cuevas J, Montesinos-López O, Jarquín D, de Los
976 Campos G, et al. Genomic Selection in Plant Breeding: Methods, Models, and
977 Perspectives. Trends Plant Sci. 2017;22: 961–975.
978 6. Goddard ME, Hayes BJ, Meuwissen THE. Using the genomic relationship matrix to
979 predict the accuracy of genomic selection. J Anim Breed Genet. 2011;128: 409–421.
980 7. Hickey JM, Dreisigacker S, Crossa J, Hearne S, Babu R, Prasanna BM, et al.
981 Evaluation of genomic selection training population designs and genotyping
982 strategies in plant breeding programs using simulation. Crop Sci. 2014;54: 1476–

983 1488.

984 8. Rabier C-E, Barre P, Asp T, Charmet G, Mangin B. On the Accuracy of Genomic
985 Selection. PLoS One. 2016;11: e0156086.

986 9. Crossa J, Beyene Y, Kassa S, Pérez P, Hickey JM, Chen C, et al. Genomic prediction
987 in maize breeding populations with genotyping-by-sequencing. G3 . 2013;3: 1903–
988 1926.

989 10. Hayes BJ, Bowman PJ, Chamberlain AJ, Goddard ME. Invited review: Genomic
990 selection in dairy cattle: progress and challenges. J Dairy Sci. 2009;92: 433–443.

991 11. García-Ruiz A, Cole JB, VanRaden PM, Wiggans GR, Ruiz-López FJ, Van Tassell
992 CP. Changes in genetic selection differentials and generation intervals in US
993 Holstein dairy cattle as a result of genomic selection. Proc Natl Acad Sci U S A.
994 2016;113: E3995-4004.

995 12. Yabe S, Hara T, Ueno M, Enoki H, Kimura T, Nishimura S, et al. Potential of
996 Genomic Selection in Mass Selection Breeding of an Allogamous Crop: An
997 Empirical Study to Increase Yield of Common Buckwheat. Front Plant Sci. 2018;9:
998 276.

999 13. Xu Y, Ma K, Zhao Y, Wang X, Zhou K, Yu G, et al. Genomic selection: A
1000 breakthrough technology in rice breeding. The Crop Journal. 2021;9: 669–677.

1001 14. VanRaden PM. Efficient methods to compute genomic predictions. J Dairy Sci.
1002 2008;91: 4414–4423.
1003 15. de Los Campos G, Hickey JM, Pong-Wong R, Daetwyler HD, Calus MPL. Whole-
1004 genome regression and prediction methods applied to plant and animal breeding.
1005 Genetics. 2013;193: 327–345.
1006 16. Habier D, Fernando RL, Kizilkaya K, Garrick DJ. Extension of the bayesian
1007 alphabet for genomic selection. BMC Bioinformatics. 2011;12: 186.
1008 17. Goddard M. Genomic selection: prediction of accuracy and maximisation of long
1009 term response. Genetica. 2009;136: 245–257.
1010 18. Jannink J-L. Dynamics of long-term genomic selection. Genet Sel Evol. 2010;42:
1011 35.
1012 19. Daetwyler HD, Hayden MJ, Spangenberg GC, Hayes BJ. Selection on Optimal
1013 Haploid Value Increases Genetic Gain and Preserves More Genetic Diversity
1014 Relative to Genomic Selection. Genetics. 2015;200: 1341–1348.
1015 20. Goiffon M, Kusmec A, Wang L, Hu G, Schnable PS. Improving Response in
1016 Genomic Selection with a Population-Based Selection Strategy: Optimal Population
1017 Value Selection. Genetics. 2017;206: 1675–1682.
1018 21. Müller D, Schopp P, Melchinger AE. Selection on Expected Maximum Haploid

1019 Breeding Values Can Increase Genetic Gain in Recurrent Genomic Selection. G3 .
1020 2018;8: 1173–1181.
1021 22. Wang L, Zhu G, Johnson W, Kher M. Three new approaches to genomic selection.
1022 Plant Breed. 2018;137: 673–681.
1023 23. Meuwissen TH. Maximizing the response of selection with a predefined rate of
1024 inbreeding. J Anim Sci. 1997;75: 934–940.
1025 24. Grundy B, Villanueva B, Woolliams JA. Dynamic selection procedures for
1026 constrained inbreeding and their consequences for pedigree development. Genet
1027 Res . 1998;72: 159–168.
1028 25. Gorjanc G, Gaynor RC, Hickey JM. Optimal cross selection for long-term genetic
1029 gain in two-part programs with rapid recurrent genomic selection. Theor Appl Genet.
1030 2018;131: 1953–1966.
1031 26. Cowling WA, Castro-Urrea FA, Stefanova KT, Li L, Banks RG, Saradadevi R, et al.
1032 Optimal Contribution Selection Improves the Rate of Genetic Gain in Grain Yield
1033 and Yield Stability in Spring Canola in Australia and Canada. Plants. 2023;12.
1034 doi:10.3390/plants12020383
1035 27. Moeinizade S, Hu G, Wang L, Schnable PS. Optimizing Selection and Mating in
1036 Genomic Selection with a Look-Ahead Approach: An Operations Research

Framework. G3 . 2019;9: 2123–2133.

28. Hunter SR, McClosky B. Maximizing quantitative traits in the mating design problem via simulation-based Pareto estimation. IIE Trans. 2016;48: 565–578.

29. Sakurai K, Hamazaki K, Inamori M, Kaga A, Iwata H. Cross potential selection: A proposal for optimizing crossing combinations in recurrent selection using the usefulness criterion of future inbred lines. G3 (Bethesda). 2024; jkae224.

30. Diot J, Iwata H. Bayesian optimisation for breeding schemes. Front Plant Sci. 2022;13: 1050198.

31. Jannink J-L, Astudillo R, Frazier P. Insight into a two‐part plant breeding scheme through Bayesian optimization of budget allocations. Crop Sci. 2023. doi:10.1002/csc2.21124

32. Moeinizade S, Hu G, Wang L. A reinforcement Learning approach to resource allocation in genomic selection. Intelligent Systems with Applications. 2022;14: 200076.

33. Hamazaki K, Iwata H. AI-assisted selection of mating pairs through simulation-based optimized progeny allocation strategies in plant breeding. Front Plant Sci. 2024;15. doi:10.3389/fpls.2024.1361894

34. Valko M, Carpentier A, Munos R. Stochastic Simultaneous Optimistic Optimization.

In: Dasgupta S, McAllester D, editors. Proceedings of the 30th International Conference on Machine Learning. Atlanta, Georgia, USA: PMLR; 2013. pp. 19–27.

35. Holland JH. Genetic Algorithms. Sci Am. 1992;267: 66–73.

36. Brochu E, Cora VM, de Freitas N. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. arXiv [cs.LG]. 2010. Available: http://arxiv.org/abs/1012.2599

37. Rios LM, Sahinidis NV. Derivative-free optimization: a review of algorithms and comparison of software implementations. J Glob Optim. 2013;56: 1247–1293.

38. Shahriari B, Swersky K, Wang Z, Adams RP, de Freitas N. Taking the human out of the loop: A review of Bayesian optimization. Proc IEEE Inst Electr Electron Eng. 2016;104: 148–175.

39. Rall LB. Automatic Differentiation: Techniques and Applications. 1981st ed. Rall LB, editor. Berlin, Germany: Springer; 1981.

40. Griewank A, Walther A. Evaluating derivatives : principles and techniques of algorithmic differentiation: Principles and techniques of algorithmic differentiation, second edition. 2nd ed. Cambridge, TAS, Australia: Cambridge University Press; 2008.

41. Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: A survey. J Mach Learn Res. 2018;18: 1–43.

42. Margossian CC. A review of automatic differentiation and its efficient implementation. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2019;9: e1305.

43. Goodfellow I, Bengio Y, Courville A. Deep Learning. London, England: MIT Press; 2016. Available: https://books.google.com/books/about/Deep_Learning.html?hl=ja&id=omivDQAAQBAJ

44. Günes B, PearlmutterBarak A, Andreyevich R, Mark S. Automatic differentiation in machine learning. Journal of Machine Learning Research. 2017 [cited 4 Sep 2024]. doi:10.5555/3122009.3242010

45. Callejo A, Narayanan SHK, García de Jalón J, Norris B. Performance of automatic differentiation tools in the dynamic simulation of multibody systems. Adv Eng Softw. 2014;73: 35–44.

46. Luchnikov IA, Krechetov ME, Filippov SN. Riemannian geometry and automatic differentiation for optimization problems of quantum physics and quantum technologies. New J Phys. 2021;23: 073006.

47. Tamayo-Mendoza T, Kreisbeck C, Lindh R, Aspuru-Guzik A. Automatic differentiation in quantum chemistry with applications to fully variational Hartree-Fock. ACS Cent Sci. 2018;4: 559–566.

48. Abbott AS, Abbott BZ, Turney JM, Schaefer HF 3rd. Arbitrary-order derivatives of quantum chemical methods via automatic differentiation. J Phys Chem Lett. 2021;12: 3232–3239.

49. Yoshikawa N, Sumita M. Automatic Differentiation for the Direct Minimization Approach to the Hartree-Fock Method. J Phys Chem A. 2022;126: 8487–8493.

50. Greener JG. Differentiable simulation to develop molecular dynamics force fields for disordered proteins. Chem Sci. 2024;15: 4897–4909.

51. AlQuraishi M, Sorger PK. Differentiable biology: using deep learning for biophysics-based and data-driven modeling of molecular mechanisms. Nat Methods. 2021;18: 1169–1180.

52. Frank SA. Automatic differentiation and the optimization of differential equation models in biology. Front Ecol Evol. 2022;10: 1010278.

53. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available: https://www.tensorflow.org/

54. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, et al. Automatic differentiation in PyTorch. 2017. Available: https://openreview.net/pdf?id=BJJsrmfCZ

55. Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, et al. JAX: Autograd and XLA. 2021.

56. Lauvernet C, Hascoët L, Le Dimet F-X, Baret F. Using automatic differentiation to study the sensitivity of a crop model. Lecture Notes in Computational Science and Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. pp. 59–69.

57. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst. 2019;abs/1912.01703. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

58. R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing; 2024. Available: https://www.R-project.org/

59. Van Rossum G, Drake FL. Python 3 Reference Manual. Scotts Valley, CA: CreateSpace; 2009.

60. Liang L, Zöllner S, Abecasis GR. GENOME: a rapid coalescent-based whole genome simulator. Bioinformatics. 2007;23: 1565–1567.

61. Yabe S, Iwata H, Jannink J-L. A simple package to script and simulate breeding schemes: The breeding scheme language. Crop Sci. 2017;57: 1347–1354.

62. Müller D, Schopp P, Melchinger AE. Persistency of Prediction Accuracy and Genetic Gain in Synthetic Populations Under Recurrent Genomic Selection. G3 . 2017;7: 801–811.

63. Van Inghelandt D, Reif JC, Dhillon BS, Flament P, Melchinger AE. Extent and genome-wide distribution of linkage disequilibrium in commercial maize germplasm. Theor Appl Genet. 2011;123: 11–20.

64. Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K. cluster: Cluster Analysis Basics and Extensions. 2021. Available: https://CRAN.R-project.org/package=cluster

65. Robbins H, Monro S. A Stochastic Approximation Method. aoms. 1951;22: 400–407.

66. Jang E, Gu S, Poole B. Categorical Reparameterization with Gumbel-Softmax. International Conference on Learning Representations. 2017. Available: https://openreview.net/pdf?id=rkE3y85ee

67. Maddison CJ, Mnih A, Teh YW. The Concrete Distribution: A Continuous

Relaxation of Discrete Random Variables. International Conference on Learning Representations. 2017. Available: https://openreview.net/pdf?id=S1jE5L5gl

68. Kosambi DD. The estimation of map distances from recombination values. Ann Eugen. 1943;12: 172–175.

69. Zhao H, Speed TP. On genetic map functions. Genetics. 1996;142: 1369–1377.

70. Lehermeier C, Teyssèdre S, Schön C-C. Genetic Gain Increases by Applying the Usefulness Criterion with Improved Variance Prediction in Selection of Crosses. Genetics. 2017;207: 1651–1661.

71. Allier A, Moreau L, Charcosset A, Teyssèdre S, Lehermeier C. Usefulness Criterion and Post-selection Parental Contributions in Multi-parental Crosses: Application to Polygenic Trait Introgression. G3 . 2019;9: 1469–1479.

72. Allier A, Lehermeier C, Charcosset A, Moreau L, Teyssèdre S. Improving Short- and Long-Term Genetic Gain by Accounting for Within-Family Variance in Optimal Cross-Selection. Front Genet. 2019;10: 1006.

73. Horn BKP, Schunck BG. Determining optical flow. Artif Intell. 1981;17: 185–203.

74. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv [cs.LG]. 2014. Available: http://arxiv.org/abs/1412.6980

75. Wu J, Poloczek M, Wilson AG, Frazier PI. Bayesian optimization with gradients.

arXiv [stat.ML]. 2017. Available: http://arxiv.org/abs/1703.04389

76. Perrin G, Leriche R. Bayesian optimization with derivatives acceleration. Transactions on Machine Learning Research. 2024 [cited 22 Oct 2024]. Available: https://openreview.net/pdf?id=JRjD0YF3Yd

77. Sekhon JS, Mebane WR Jr. Genetic Optimization Using Derivatives. Polit Anal. 1998;7: 187–210.

78. Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw. 2015;61: 85–117.

Data availability statement

All the datasets for initial populations were simulated by R in this study. The simulated datasets in the present study are available from the "KosukeHamazaki/GORA-PT" repository on GitHub, https://github.com/KosukeHamazaki/GORA-PT. The optimized results by PyTorch were not shared in this repository due to the file sizes, so the all datasets, including the optimized ones, will be shared in the "GORA-PT: Gradient-based Optimization of Resource Allocation of progeny in breeding schemes with PyTorch" repository on Zenodo, https://doi.org/10.5281/zenodo.14046522.

The proposed automatic differentiation framework was implemented as the Python module "BST". This module and all the scripts used in this study, including the gradient-based optimization, are available from the "KosukeHamazaki/GORA-PT" repository on GitHub, https://github.com/KosukeHamazaki/GORA-PT.

Supporting information

Appendix S1. Supplementary Note for the detailed theoretical derivation of the derivative of genetic variance of gametes.

Figure S1. The differentiable resource allocation strategy of progenies developed in

this study.

Figure S2. Example of automatic differentiation of the final genetic gain with respect to the allocation parameters in breeding schemes assuming the recurrent selection.

Figure S3. Comparison between automatic differentiation and analytical solutions for the first-order derivatives of the genetic variance of gametes with respect to the parent's marker genotype. Scatter plots of the first-order derivatives of the genetic variance of different numbers of gametes with respect to the parent's marker genotype. The horizontal and vertical axes represent the results derived from the automatic differentiation and analytical solutions, respectively. The red line indicates where the automatic differentiation results perfectly match the analytical solutions. (A), (C), (E) Method 1. (B), (D), (F) Method 2. (A), (B) Results of 100 gametes. (C), (D) Results of 10,000 gametes. (E), (F) Results of 50,000 gametes.

Figure S4. Comparison between two methods of generating gametes for the first-order derivatives of the genetic variance of gametes with respect to the parent's marker genotype. Scatter plots of the first-order derivatives of the genetic variance of different numbers of gametes with respect to the parent's marker genotype. The horizontal and vertical axes represent the results derived from Method 1 and Method 2 of generating gametes, respectively. The red line indicates where Method 1 perfectly matches Method

2. (A) Results of 100 gametes. (B) Results of 1,000 gametes. (C) Results of 10,000 gametes. (D) Results of 50,000 gametes.

Figure S5. Convergence conditions of the SGD algorithm. Change in the function values with different epochs. The horizontal and vertical axes represent the number of epochs of the SGD algorithm and the function values evaluated with 100 breeding schemes, respectively. (A), (C), (E) Scenario 1. (B), (D), (F) Scenario 2. (A), (B) SI1. (C), (D) SI2. (E), (F) SI3.

Figure S6. Genetic gains across different simulation repetitions. The horizontal axis shows the final genetic gain of an individual, whereas the vertical axis represents the percentile of the simulation repetitions. We compared the four allocation strategies under the different selection intensities (SI1 – SI3). The abbreviations of the allocation strategies are the same as those of Figure 5. (A) Scenario 1. (B) Scenario 2.

Figure S7. Improvement of the final genetic gains compared to the equal allocation strategy using ten replications for the phenotype simulation under different selection intensities. Boxplots of the improvement of the final genetic gains compared to the equal allocation strategy using ten replications for the phenotype simulation. The horizontal and vertical axes represent the different allocation strategies and the improvement rate of the final genetic gains, respectively. We compared the following two allocation strategies

under the different selection intensities (SI1 – SI3): GORA2: gradient-based optimized allocation with smaller initial weights on GVP (light purple), ORA: black-box-based optimized allocation (light blue). The weighting parameters were optimized based on 100 and 5,000 function evaluations for GORA2 and ORA, respectively. (A) Scenario 1. (B) Scenario 2.

Figure S8. Optical flow for the gradients calculated by automatic differentiation under SI1. The function values and gradients were evaluated by automatic differentiation via grid search under SI1. The horizontal and vertical axes represent the logit-transformed weighting parameters $\mathbf{z}^{(\tau)}$ in the allocation step. The background color represents the size of the function values for each set of logit-transformed weighting parameters, whereas the arrows reflect the sizes and directions of the gradients evaluated by automatic differentiation. The blue and green points correspond to the initial and optimized values by the GORA2 approach. (A), (B) The gradients were evaluated for the breeding scheme with only one generation considering both WBV and GVP as candidates of $\mathbf{\Omega}^{(\tau)}$. The horizontal and vertical axes represent the logit-transformed weighting parameters for WBV and GVP in generation 0, respectively. (C), (D) The gradients were evaluated for the breeding scheme with two generations considering only WBV as candidates of $\mathbf{\Omega}^{(\tau)}$. The horizontal and vertical axes represent the logit-transformed weighting parameters for

WBV in generations 0 and 1, respectively. (A), (C) Scenario 1. (B), (D) Scenario 2.

Figure S9. Optical flow for the gradients calculated by automatic differentiation under SI3. The function values and gradients were evaluated by automatic differentiation via grid search under SI3. The horizontal and vertical axes represent the logit-transformed weighting parameters $\mathbf{z}^{(\tau)}$ in the allocation step. The background color represents the size of the function values for each set of logit-transformed weighting parameters, whereas the arrows reflect the sizes and directions of the gradients evaluated by automatic differentiation. The blue and green points correspond to the initial and optimized values by the GORA2 approach. (A), (B) The gradients were evaluated for the breeding scheme with only one generation considering both WBV and GVP as candidates of $\mathbf{\Omega}^{(\tau)}$. The horizontal and vertical axes represent the logit-transformed weighting parameters for WBV and GVP in generation 0, respectively. (C), (D) The gradients were evaluated for the breeding scheme with two generations considering only WBV as candidates of $\mathbf{\Omega}^{(\tau)}$. The horizontal and vertical axes represent the logit-transformed weighting parameters for WBV in generations 0 and 1, respectively. (A), (C) Scenario 1. (B), (D) Scenario 2.

Table S1. Optimized weighting parameters $\mathbf{h}^{(\tau)}$ across different generations $\boldsymbol{\tau} = \mathbf{0}, \dots, \boldsymbol{T} - \mathbf{1}$ for Scenario 1 under SI2.

Scenario 1, SI2	GORA1		GORA2		ORA	
Generation (τ)	WBV	GVP	WBV	GVP	WBV	GVP
0	3.25	2.71	3.21	2.40	1.50	0.17
1	2.44	2.20	2.50	0.08	2.83	0.17
2	2.51	1.80	2.59	−0.02	2.83	1.5
3	2.26	1.95	2.47	−0.07	2.83	0.17

Table S2. Optimized weighting parameters $\mathbf{h}^{(\tau)}$ across different generations $\boldsymbol{\tau} = \mathbf{0}, \dots, \boldsymbol{T} - \mathbf{1}$ for Scenario 1 under SI3.

Scenario 1, SI3	GORA1		GORA2		ORA	
Generation (τ)	WBV	GVP	WBV	GVP	WBV	GVP
0	3.29	2.07	3.26	2.06	2.83	0.17
1	2.58	1.87	2.69	0.01	2.83	0.17
2	2.74	1.61	2.85	−0.04	1.50	0.17
3	2.59	1.81	2.69	−0.06	2.83	0.17

Table S3. Optimized weighting parameters $\mathbf{h}^{(\tau)}$ across different generations $\boldsymbol{\tau} = \mathbf{0}, \dots, \boldsymbol{T} - \mathbf{1}$ for Scenario 2 under SI1.

Scenario 2, SI1	GORA1		GORA2		ORA	
Generation (τ)	WBV	GVP	WBV	GVP	WBV	GVP
0	2.36	0.02	2.44	−0.17	1.06	0.17
1	2.55	2.20	2.64	0.56	0.17	0.17
2	2.32	2.36	2.63	0.05	0.17	0.17
3	2.17	2.19	2.57	−0.17	2.83	0.17

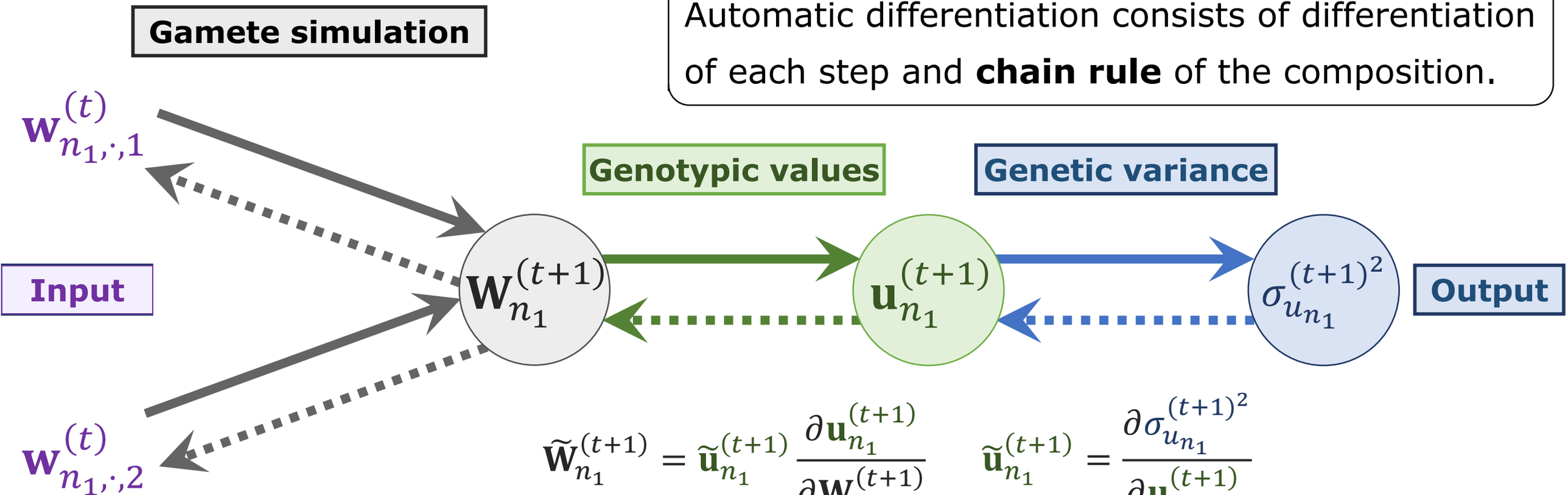
Table S4. Optimized weighting parameters $\mathbf{h}^{(\tau)}$ across different generations $\boldsymbol{\tau} = \mathbf{0}, \dots, \boldsymbol{T} - \mathbf{1}$ for Scenario 2 under SI2.

Scenario 2, SI2	GORA1		GORA2		ORA	
Generation (τ)	WBV	GVP	WBV	GVP	WBV	GVP
0	2.52	0.30	2.23	−0.19	1.06	1.06
1	2.68	1.67	2.68	0.22	1.94	0.61
2	2.51	2.04	2.80	−0.05	1.06	0.61
3	2.35	2.08	2.68	−0.16	3.28	0.17

Table S5. Optimized weighting parameters $\mathbf{h}^{(\tau)}$ across different generations $\boldsymbol{\tau} = \mathbf{0}, \dots, \boldsymbol{T} - \mathbf{1}$ for Scenario 2 under SI3.

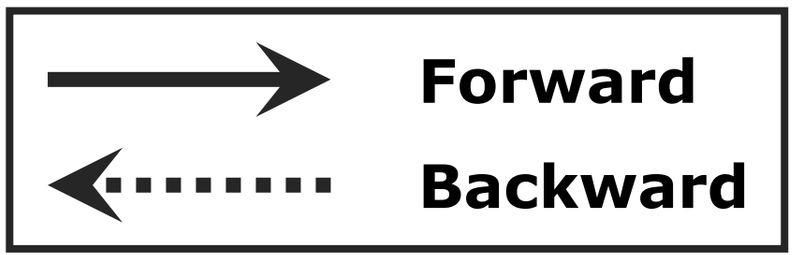
Scenario 2, SI3	GORA1		GORA2		ORA	
Generation (τ)	WBV	GVP	WBV	GVP	WBV	GVP
0	2.78	0.02	2.50	−0.17	1.06	0.17
1	2.81	1.62	2.81	0.10	1.50	0.61
2	2.73	1.69	2.88	−0.06	1.06	0.17
3	2.57	1.82	2.79	−0.14	3.28	0.17

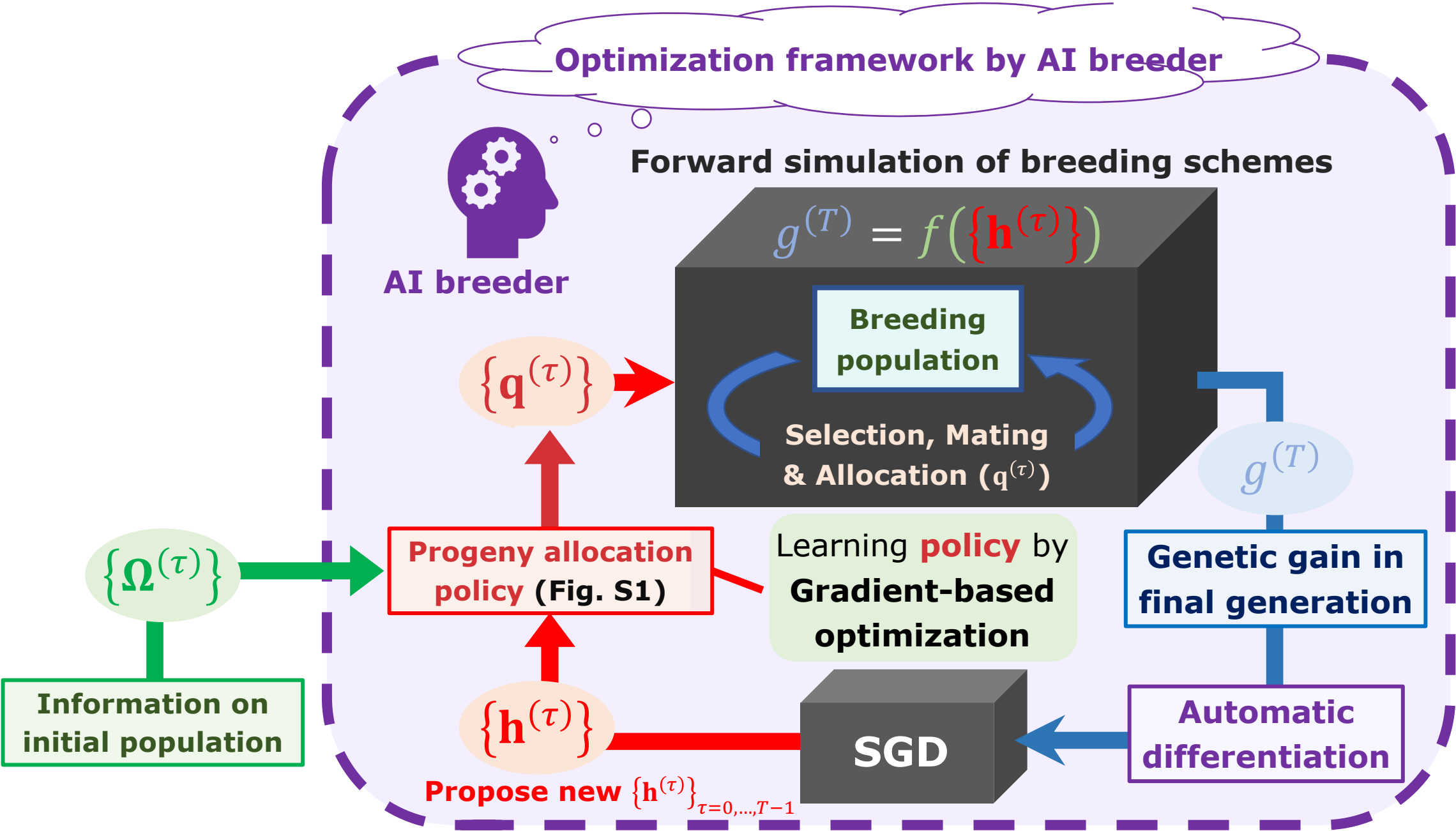
Automatic differentiation consists of differentiation of each step and **chain rule** of the composition.

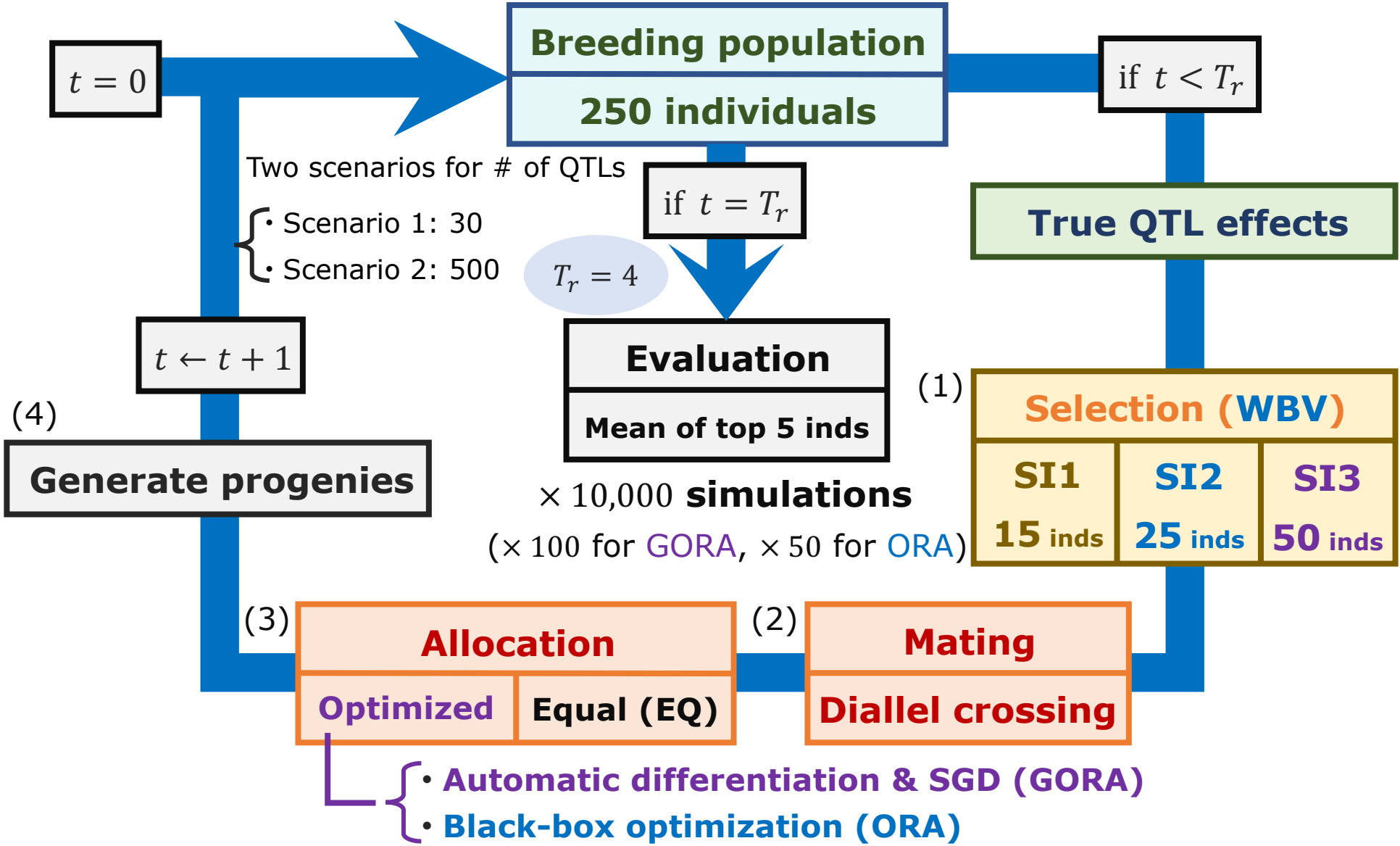


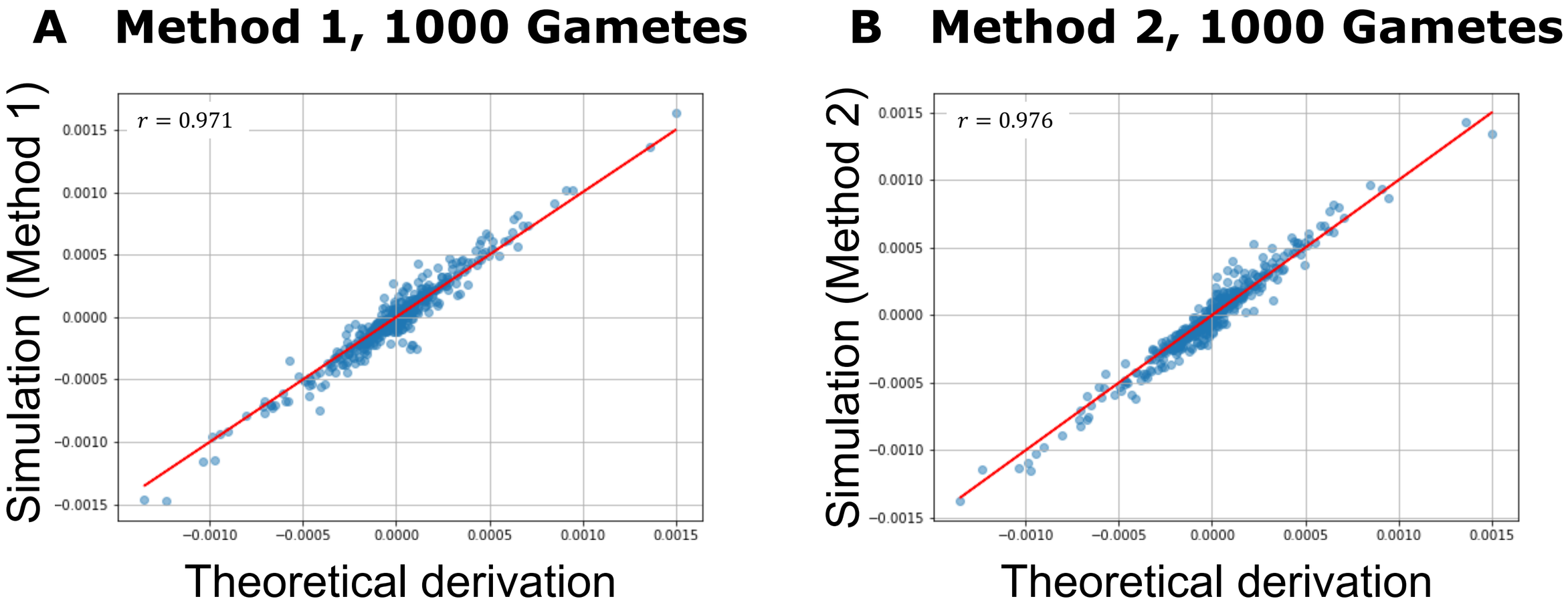
$$\widetilde{\mathbf{W}}^{(t+1)}_{n_1} = \widetilde{\mathbf{u}}^{(t+1)}_{n_1} \frac{\partial \mathbf{u}^{(t+1)}_{n_1}}{\partial \mathbf{W}^{(t+1)}_{n_1}} \qquad \widetilde{\mathbf{u}}^{(t+1)}_{n_1} = \frac{\partial \sigma^{(t+1)^2}_{u_{n_1}}}{\partial \mathbf{u}^{(t+1)}_{n_1}}$$

$$\frac{\partial \sigma^{(t+1)^2}_{u_{n_1}}}{\partial w^{(t)}_{n_1,m,l}} = \widetilde{w}^{(t)}_{n_1,m,l} = \widetilde{\mathbf{W}}^{(t+1)}_{n_1} \frac{\partial \mathbf{W}^{(t+1)}_{n_1}}{\partial w^{(t)}_{n_1,m,l}}$$

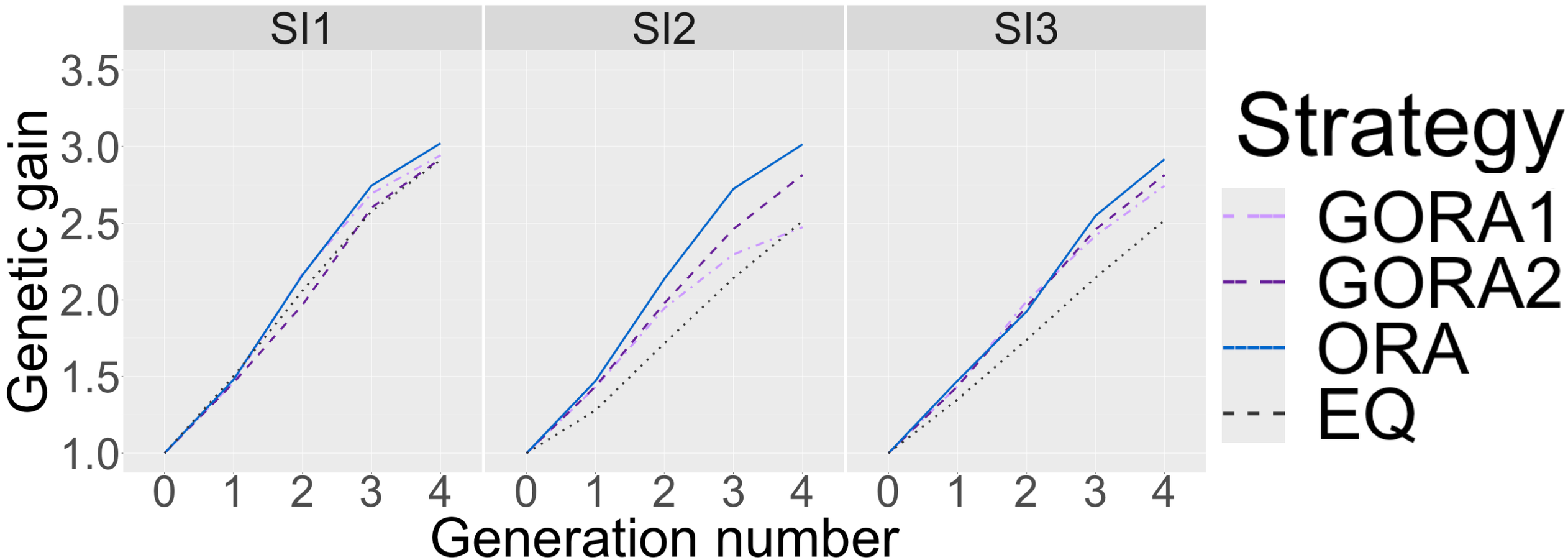



Optimization framework by AI breeder
AI breeder
Forward simulation of breeding schemes
$g^{(T)} = f(\{\mathbf{h}^{(\tau)}\})$
Breeding population
Selection, Mating & Allocation ($\mathbf{q}^{(\tau)}$)
$\{\mathbf{q}^{(\tau)}\}$
$g^{(T)}$
$\{\Omega^{(\tau)}\}$
Progeny allocation policy (Fig. S1)
Learning policy by Gradient-based optimization
Genetic gain in final generation
Information on initial population
$\{\mathbf{h}^{(\tau)}\}$
SGD
Automatic differentiation
Propose new $\{\mathbf{h}^{(\tau)}\}_{\tau=0,\ldots,T-1}$

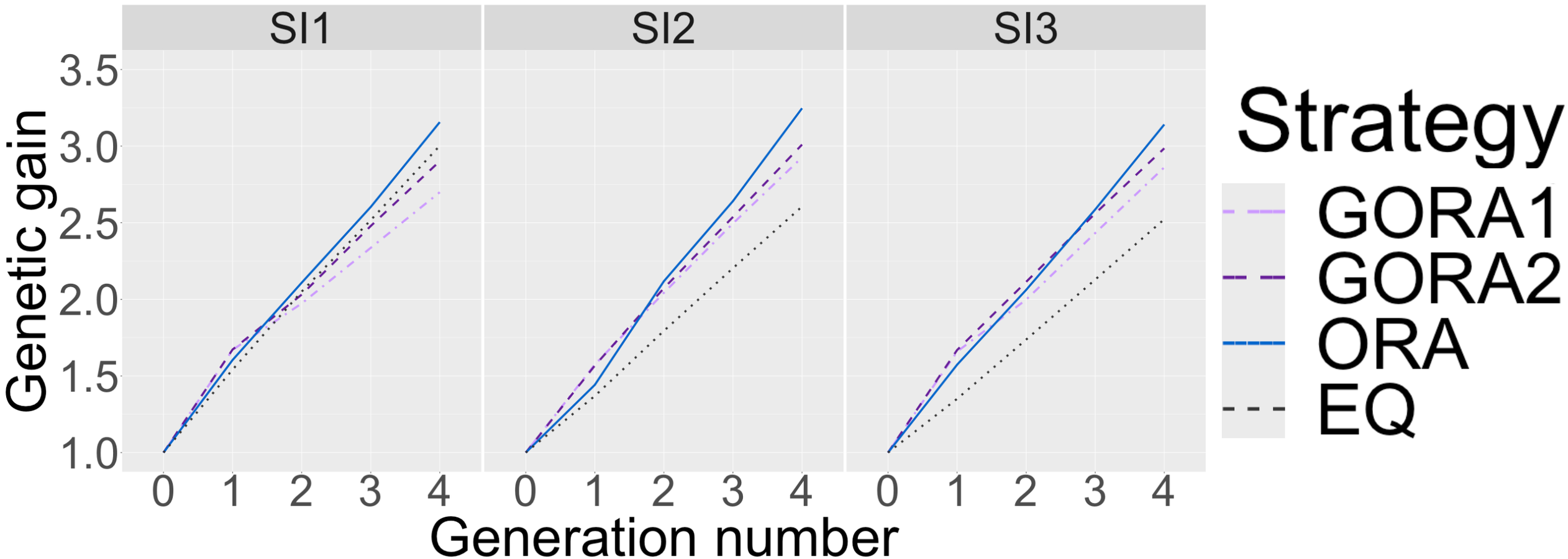

$t = 0$
Breeding population
250 individuals
if $t < T_r$
Two scenarios for # of QTLs
• Scenario 1: 30
• Scenario 2: 500
if $t = T_r$
$T_r = 4$
True QTL effects
$t \leftarrow t + 1$
Evaluation
Mean of top 5 inds
(1)
Selection (WBV)
SI1
15 inds
SI2
25 inds
SI3
50 inds
(4)
Generate progenies
× 10,000 simulations
(× 100 for GORA, × 50 for ORA)
(3)
Allocation
Optimized
Equal (EQ)
(2)
Mating
Diallel crossing
• Automatic differentiation & SGD (GORA)
• Black-box optimization (ORA)


A Method 1, 1000 Gametes
B Method 2, 1000 Gametes
$r = 0.971$
$r = 0.976$
Simulation (Method 1)
Simulation (Method 2)
Theoretical derivation
Theoretical derivation
0.0015
0.0010
0.0005
0.0000
−0.0005
−0.0010
−0.0015

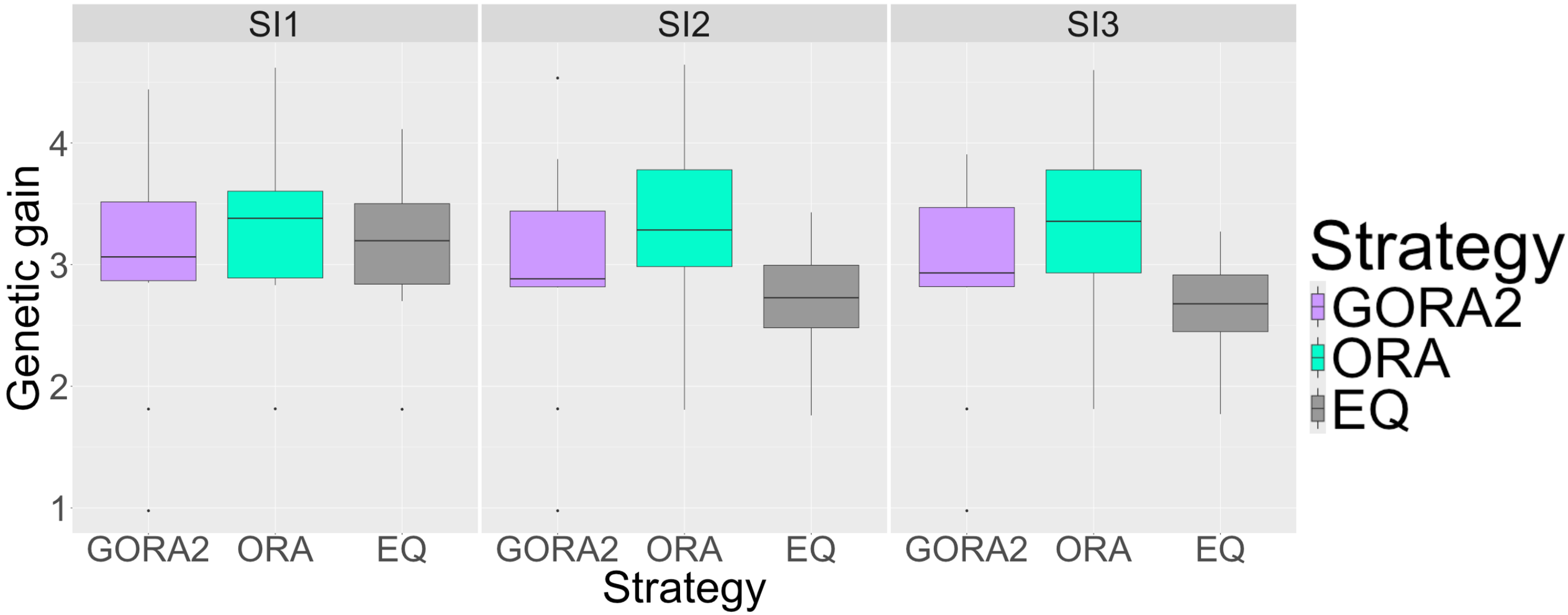
A Scenario 1



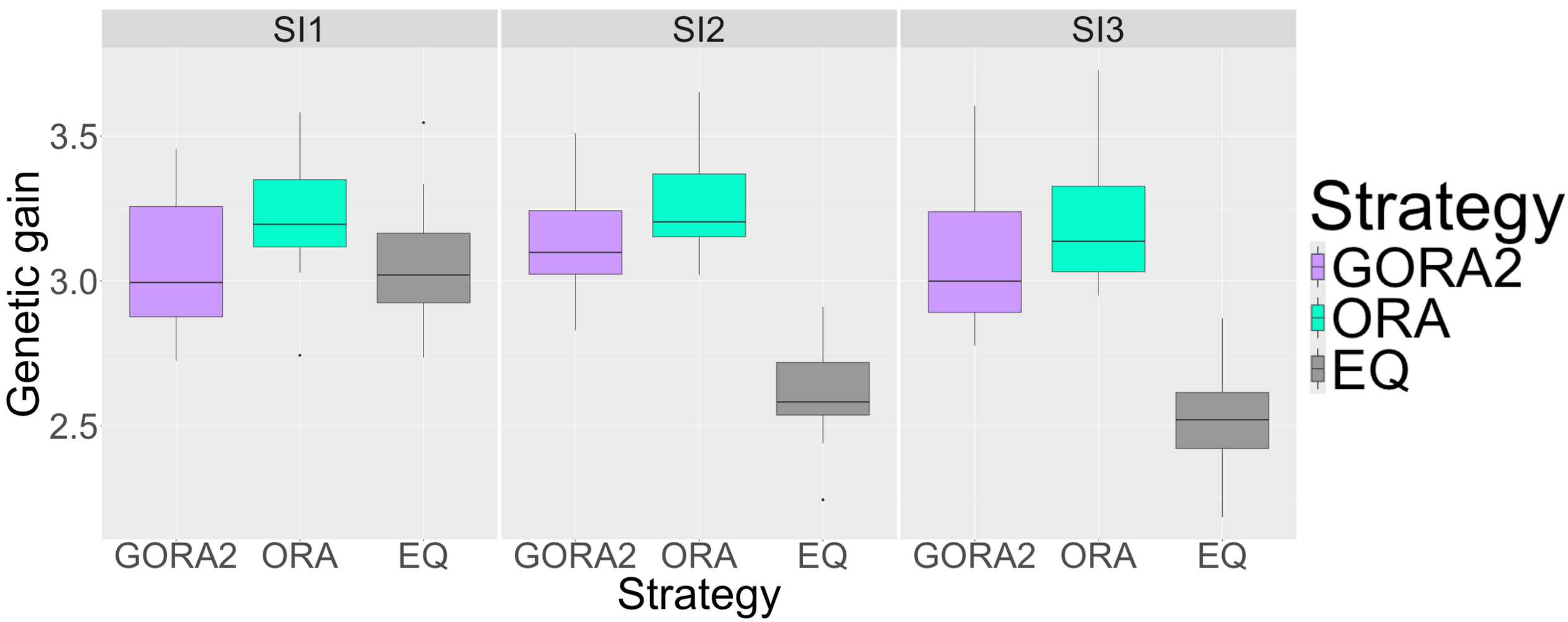
B Scenario 2



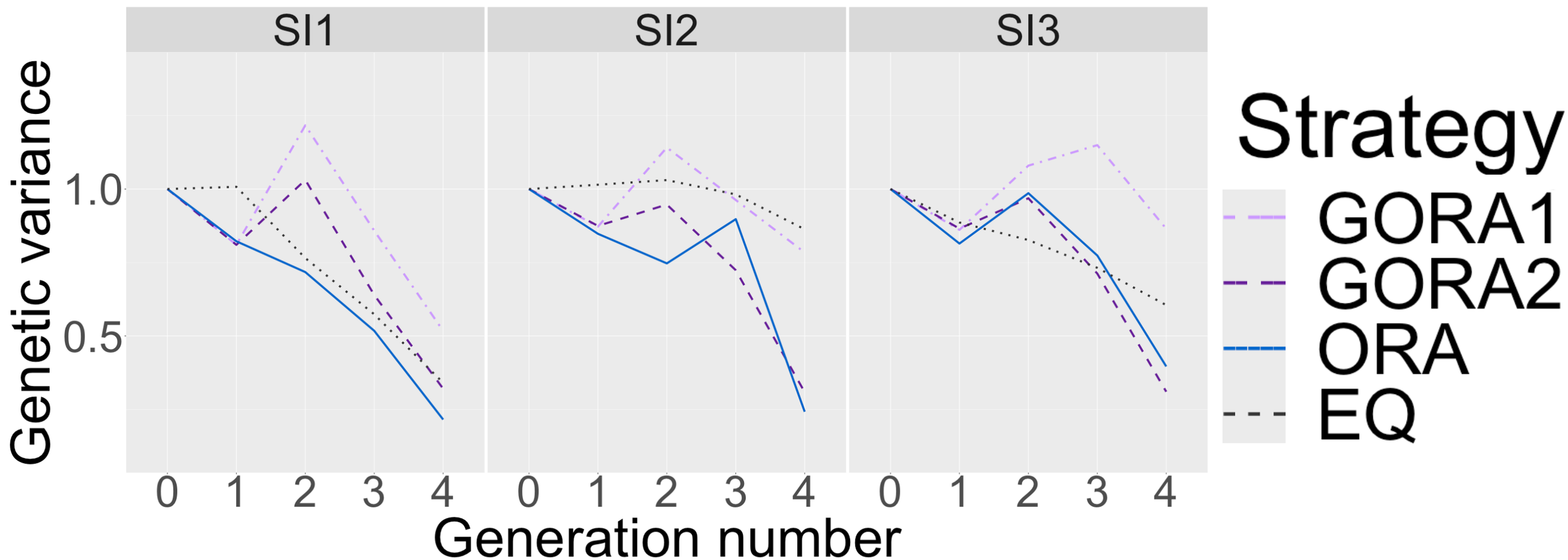
A Scenario 1



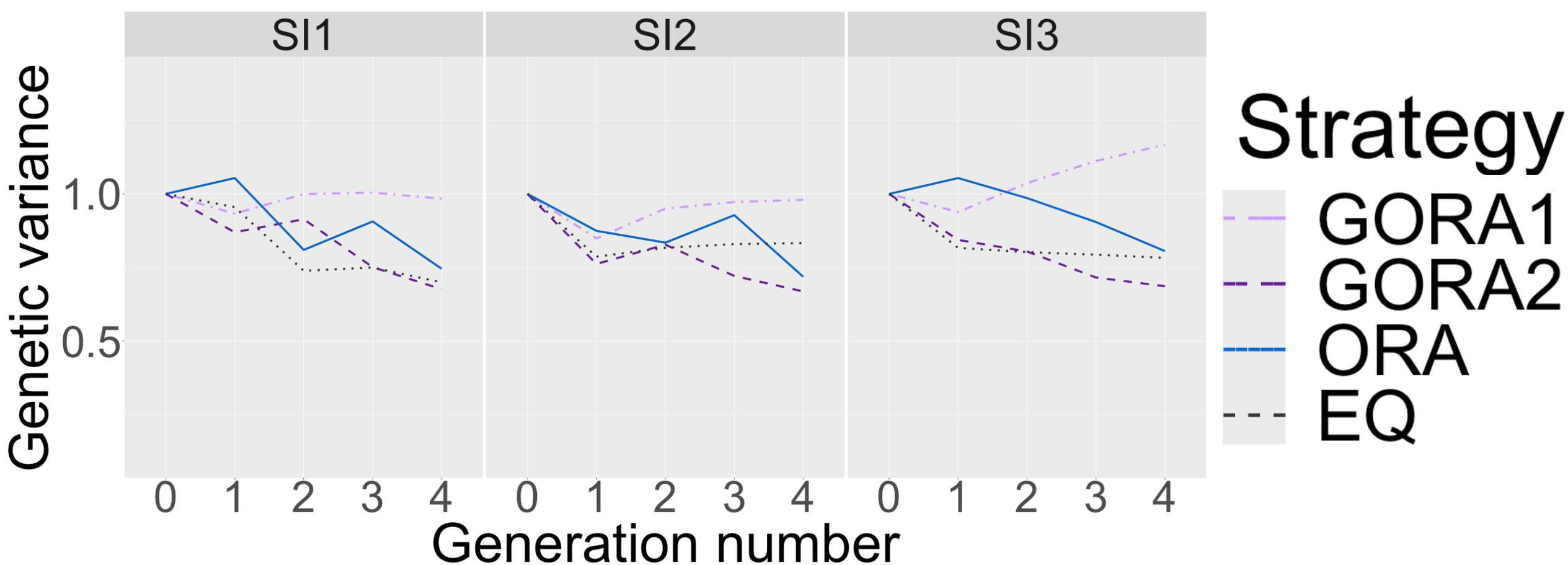
B Scenario 2

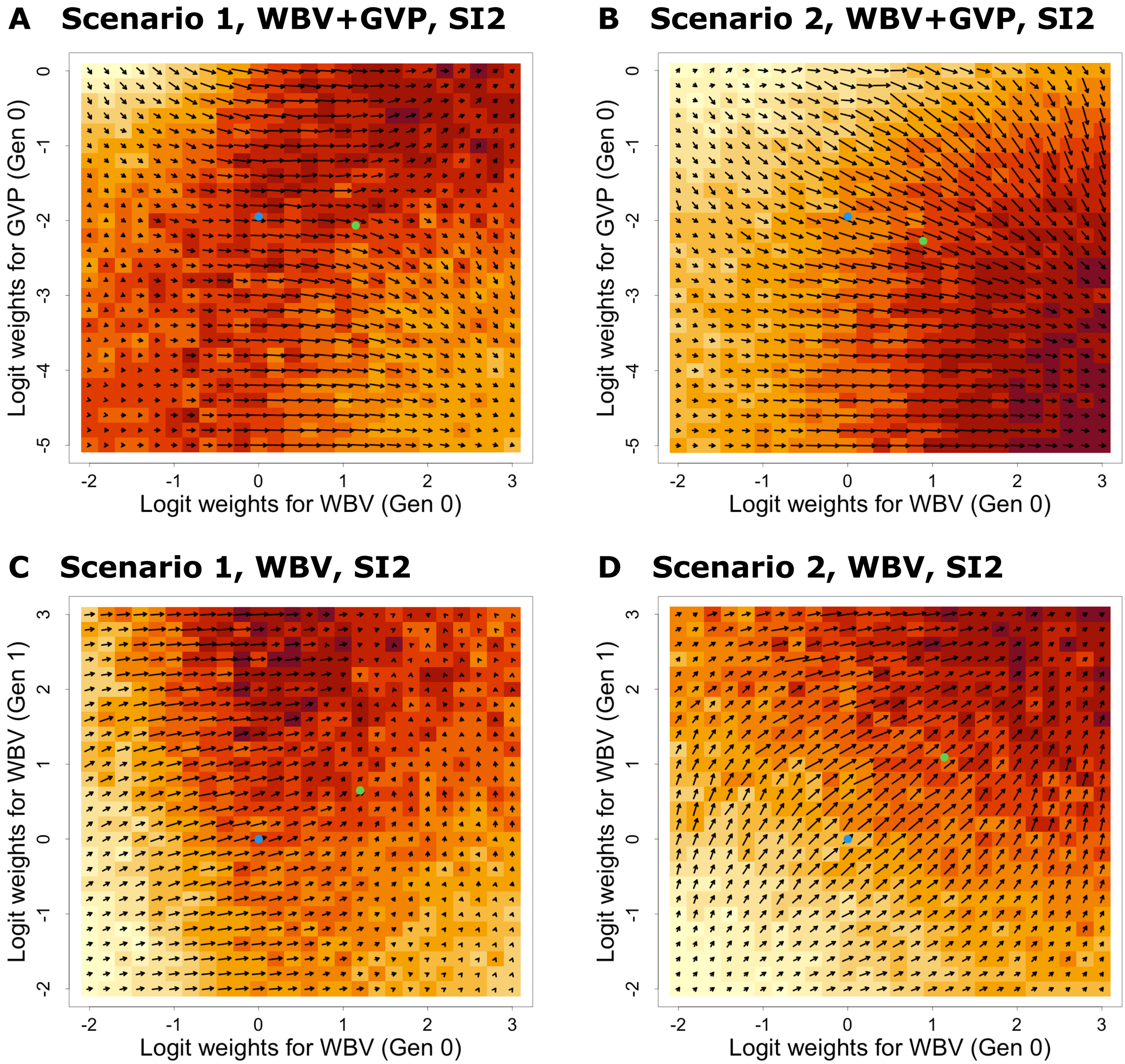


A Scenario 1



B Scenario 2




A Scenario 1, WBV+GVP, SI2
B Scenario 2, WBV+GVP, SI2
C Scenario 1, WBV, SI2
D Scenario 2, WBV, SI2
Logit weights for GVP (Gen 0)
Logit weights for WBV (Gen 1)
Logit weights for WBV (Gen 0)
0
-1
-2
-3
-4
-5
3
2
1
-2
-1
0
1
2
3