An Efficient Way of Introducing Gender Into Evolutionary Algorithms

Christian Kasten, Julian Fahr, and Markus Klein

***Abstract*—Evolutionary algorithms have been extensively used for numerous optimization problems with great success in the past years. They mimic nature's process of evolution to find solutions to a large range of mathematical problems. In this work a new strategy is suggested to introduce gender, defined by a characteristic of an individual, that is easy to implement in evolutionary algorithms, as long as they are based on evaluating a fitness function. The new method outperforms comparable evolutionary approaches without gender for all standard test problems considered. The present study shows that with increasing problem complexity the performance of this gender variant increases to more than double the success rates while keeping the computational effort at about the same level and still being clearly better for easy problems. Additionally, in the mean, the new method results in better fitness values for all presented cases.**

***Index Terms*—Evolutionary algorithm, gender, gene expression programming.**

I. Introduction

As THE basic idea of evolutionary algorithms (EA) [1]–[3] is to copy the natural process of evolution [4] many characteristics of biological evolution are already considered [5] and implemented in EA comparable to the biological processes. With these mechanisms all kind of problems have been tackled in the field of general engineering [6]–[8], turbulence modeling [9]–[13], or health technologies [14]. Additionally, EAs have even been successfully used in arts and music [15]–[17] or software engineering [18]–[20] to fix bugs or to increase the performance of compilers.

The concept of gender on the chromosome level has been known since the early 1900s [21]. It has been introduced in a variety of creative ways in EA by different authors over the last decades [22]–[24], but widely it has been ignored in productive codes due to small performance increases. The question of gender is to some extent (but not exclusively) related to the question of parent selection which has been discussed in the literature by several authors: Świechowski [25] has analyzed the complementary fitness based on splitting the decision tree. Thierens and Bosman [26] discussed if substructures are available in different parents and changed the mating accordingly. Finally, Ryerkerk *et al.* [27] provided a good review on how to ensure diversity.

Despite natures definition of gender by a genetic characteristic, all evolutionary codes known to the authors that deal with gender set the gender by random or at least not by an individual's attributes (e.g., [28]). Having in mind that most of today's successful species evolved with a genetic feature to determine their gender it is of great interest that this concept finds its way into the evolutionary computation. As a matter of fact humankind has always tried to copy processes from nature and to think ahead from there, which already helped while developing the first aircraft by da Vinci or by letting a neural net evolve like a brain evolved in nature [29]. While the concept of nature is to determine the gender by the X- and Y-chromosomes this is not easy to imitate in evolutionary computation. Adding these chromosomes to an individuals DNA just leads to the questions when to assign two X-chromosomes or one X- and one Y-chromosome to an individual and what these chromosomes mean to an individual. On a macroscopic level being male or female has always decided about the behavior of individuals, their strengths and weaknesses, and their overall characteristics. From this point of view, the gender of an individual in evolutionary computation could be identified after its birth based on its characteristics just like it was done in medicine before it was possible to identify a fetus' gender by genetic tests. Consequently, it is suggested to determine the gender of an individual in the sense of evolutionary computation by an individuals characteristic instead of letting its characteristic be decided by its gender. Obviously, an individual in terms of evolutionary computation has considerably less properties than most individuals in real evolution. Additionally, it is desirable that the characteristic that is chosen to determine the gender in the sense of evolutionary computation should assist the convergence of the evolutionary algorithm and should not increase the computational effort tremendously. In the authors opinion the sign of the error satisfies these requirements in an ideal way. Beside the fact that this information is already present in an evolutionary algorithm it does not only divide a population in two completely different groups but it is highly likely that it also increases the convergence and the findings of superb results. Taking this characteristic to determine the gender is the main idea of this article besides splitting the population into male and female which comes along with the adoption of the selection process. After allowing reproduction only between male and female with the different reproduction possibilities there is always a chance that a negative and a positive error

Manuscript received 21 September 2021; revised 23 March 2022; accepted 13 July 2022. Date of publication 19 July 2022; date of current version 1 August 2023. This work was supported by the Universität der Bundeswehr München. *(Corresponding author: Christian Kasten.)*

The authors are with the Fakultät für Luft- und Raumfahrttechnik, Universität der Bundeswehr München, 85577 Neubiberg, Germany (e-mail: christian.kasten@unibw.de; julian.fahr@unibw.de; markus.klein@unibw.de).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TEVC.2022.3192481.

Digital Object Identifier 10.1109/TEVC.2022.3192481

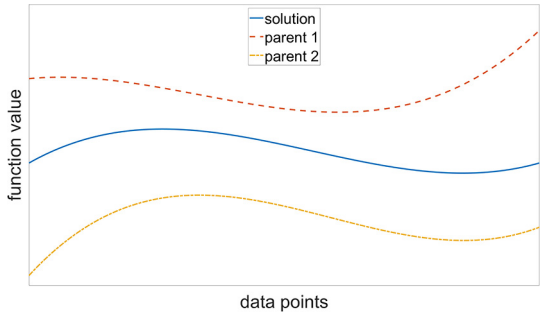

Fig. 1. Definition of male and female with positive (parent 1) and negative (parent 2) error eventually combining to an offspring with smaller error.

partially compensate each other by letting a male with a positive error and a female with a negative error (or the other way round) reproduce by combining their most influential features like illustrated in Fig. 1. This idea can be easily combined with other gender-based ideas like in [30], where different selection schemes for males and females have been introduced with different selection pressures, inspired by the idea of male vigor and female choice. The concept that diversity is increased by this idea should further benefit from the present determination of gender based on an individuals characteristics. Details regarding the implementation as well as "gender specific" modifications of the original EA will be presented and discussed next.

II. Approach

A. Basic Idea

This section first describes the enhancement of EA with gender and subsequently the codebasis for the present work. While the principle of this work can be applied to any EA that provides a measure of an individuals success, such as a cost or fitness function, a gene expression programming (GEP) algorithm is used for illustration. GEP, developed by Ferreira [31], combines the simplicity of chromosomes and their encoding as introduced by Holland [1] with the functional complexity achieved by expressing them as parse trees as suggested by Koza [32]. The fundamental difference in GEP compared to other EAs resides in the nature of the individuals [31]. GEP genes are composed of a head and a tail and despite their fixed length, each gene has the potential to code for expression trees of different sizes and shapes [31]. For a detailed description of how GEP works and how a solution is represented in detail the reader is referred to [31] or the more recent work by Schopplein *et al.* [9]. A central property of every GEP is the definition of the fitness function. For the fitness case j and C_t fitness cases and with F_j and $C_{(i,j)}$ as the target function values and the values returned by the chromosome of the individual i, the fitness of individual i, f_i is defined by

$$f_i = \frac{1}{C_t}\sum_{j=1}^{C_t}\left(F_j - C_{(i,j)}\right)^n, \text{ with } n = 2. \tag{1}$$

Obviously, a variety of other fitness functions could be defined. In the context of this work, based on the fitness function, it should be possible to divide the population, or a subset of it, into two groups with, e.g., positive and negative values. In this analysis, the squared error was chosen as the fitness function due to its simplicity and its wide use in engineering applications. Additionally, to determine the gender as the sign of the error it is required to sum (1) up for $n = 1$ (or another odd number)

$$g_i = \frac{1}{C_t}\sum_{j=1}^{C_t} F_j - C_{(i,j)}. \tag{2}$$

The main idea of this article is to add gender to EA by calculating the error g_i with (2) and defining an individual i with (where this convention is of course arbitrary)

$$\begin{aligned} g_i &> 0 \quad \text{as male} \\ g_i &< 0 \quad \text{as female.} \end{aligned} \tag{3}$$

This has to be done for all individuals while creating the random initial population at the beginning and for all individuals created during later generations. In summary, the proposed method is to group the individuals according to whether they overshoot or undershoot the target function. The intuition of this mechanism is that by combining parts of the program that create the opposite effects, the offspring can balance its prediction. As a result of introducing gender, males only compete with other males and females only compete with other females for reproduction using any selection scheme. In the case of tournament selection half the planned tournaments are held for all the males and half are held for all the females. Note, that it is also imaginable that different selection schemes can be defined for both genders to introduce different selection pressures like for example proposed in [30]. After the males and females that are allowed to reproduce are selected, reproduction only takes place between male and female. For the children created the gender is again calculated according to (3).

For GEP and as well for GEP with gender, henceforth denoted GGEP, it can happen that the fitness function cannot be evaluated for example because a floating point error occurs. In this case, the gender is specified randomly. But as a matter of fact, from the computational point of view, it would also be possible to always add individuals with an error of not a number (NAN) to one specific gender. It could be observed that there is no difference during the optimization between the two options because individuals with an NAN error and consequently with an NAN fitness are and should be removed from the next generation anyways.

B. Codebase for Gene Expression Programming

The used code is a GEP [31] code called EVE written in Python and described in detail in [33]. GEP can be seen as a combination of Genetic Programming and Genetic Algorithms whereby Genetic Programming was first introduced by Cramer [34] and later further developed by Koza [32]. Genetic Algorithms were established by Holland [1] and are according to [35] becoming increasingly attractive for researchers from various disciplines, such as operations research, computer science, industrial engineering, electrical engineering, social

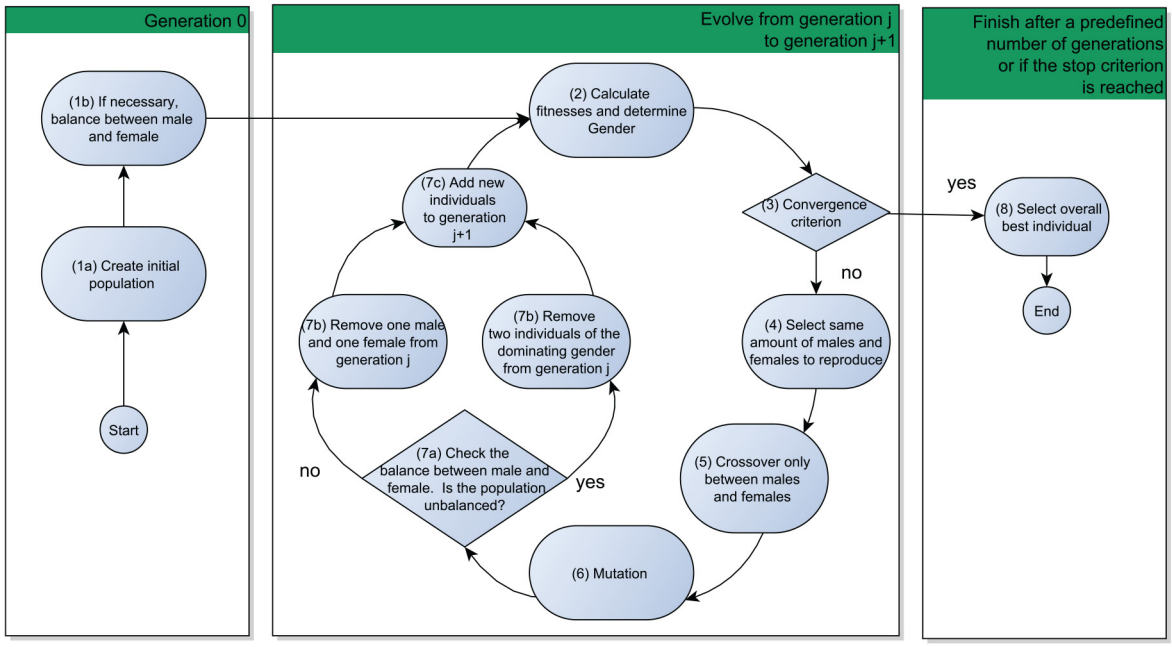

Fig. 2. Workflow of GGEP.

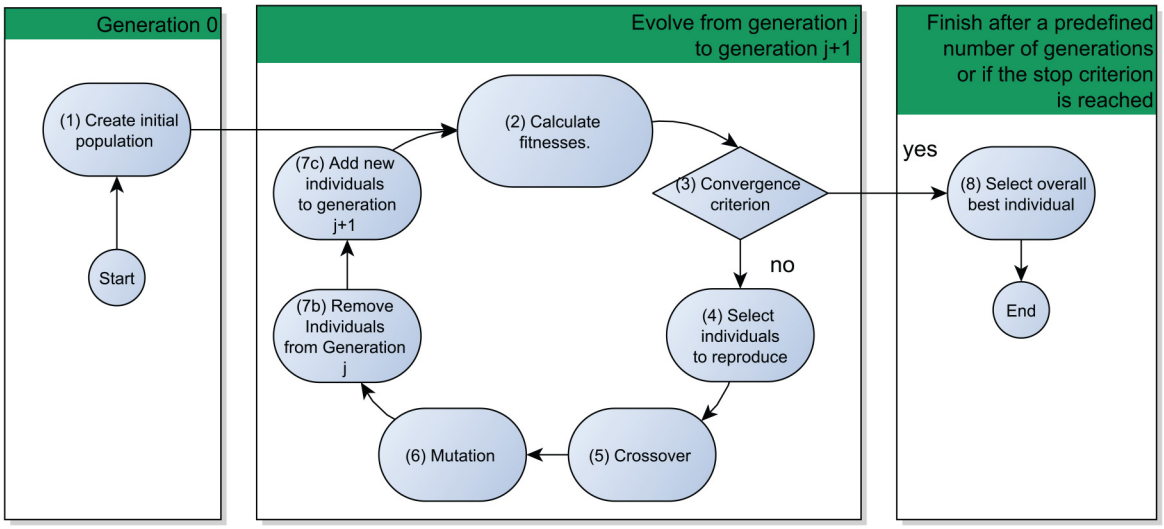

Fig. 3. Workflow of GEP.

science, and economics. The EVE code has been expanded with gender as described in the previous section. The code has the possibility to define different probabilities for crossovers and mutation, is easy to expand and can be used in various engineering and academic problems. After the initialization of the (quasi) random initial population the steps shown in Fig. 2 are repeated for GGEP until a predefined fitness level or the maximum number of generations is reached. The workflow for conventional GEP is shown in Fig. 3. Note that the difference between the workflow of GGEP in Fig. 2 and GEP in Fig. 3 can be seen in steps 1, 2, 4, 5, and 7.

Additionally, EVE uses a diversity mechanism called flood where the standard deviation and mean values of all fitness values of the current population are calculated and if the standard deviation undercuts a configured percentage of the mean value the flood replaces a part of the population with randomly created individuals. For GGEP this procedure is done separately for males and females.

Furthermore, the selection scheme is another important property that impacts the sequence of the optimization. In the current analysis, a tournament selection with elitism was used with the tournament size t_{size} defining how big the tournaments are and the mating size m_{size} defining how many tournaments are held in every generation. To have a fair comparison the same amount of individuals has to be sampled every generation for GEP and GGEP. Therefore, the GGEP code always holds half the mating size tournaments for male and half the mating size tournaments for female individuals.

The amount of available genetic operators is kept low to minimize the impact of other effects while comparing GEP

TABLE I
PROPERTIES OF GEP/GGEP FOR THE DIFFERENT BENCHMARK PROBLEMS

Property	problem 1	problem 2	problem 3
Population size	$[20, 200]$	$[20, 800]$	$[20, 800]$
Number of generations	$[30, 300]$	$[30, 300]$	$[100, 300]$
head length $[h]$	5	4	5
number of genes $[n_{genes}]$	5	6	5
t_{size}	$0.1 \cdot$ population size	$0.1 \cdot$ population size	$0.1 \cdot$ population size
m_{size}	$0.5 \cdot$ population size	$0.5 \cdot$ population size	$0.5 \cdot$ population size
Function set	$[+, -, *, /]$	$[+, -, *, /]$	$[+, -, *, /]$
Terminal set	$[x]$	$[x]$	$[x, y]$
Constant set	[]	[]	$[-1, 1]$
Mutation probability per gene	0.06	$\frac{2}{n_{genes}(2h+1)}$	$\frac{2}{n_{genes}(2h+1)}$
One point crossover probability	0.3	0.7	0.3
Two point crossover probability	0.2	0	0.2
Fitness tolerance	0.01	0.0001	10^{-20}

with GGEP. Consequently, only mutation, one-point-crossover, and two-point-crossover are used for reproduction and their probabilities and further properties can be seen in Table I for the different problems described in the test problem description section.

Note that the differences of success rates between the presented results here and the work of Ferreira [31] for GEP can be lead back to small configuration differences like the absence of insertion methods in this work. This can be especially seen for problem 2, but the focus of this work lies in the comparison between GEP and GGEP. Beside that, the property definition mostly follows the work of Ferreira [31] and has been optimized for GEP to avoid any efficiency bias toward the new method. Moreover, it was found that for most set ups the optimal values of GGEP were close to the optimal values for GEP. One big difference occurs for increasing population sizes. Whereas GEP benefits unremarkably from increasing population size GGEP improves considerably. If one is willing to invest the computational power this is especially useful because GGEP offers the opportunity to get superb results for increasing population sizes as will be demonstrated later on.

C. Selection and Balance Between Male and Female

Two unbalanced populations represent a clear disadvantage for GGEP. Per se there is no evidence that the error has a higher probability to be negative or positive and consequently an individual has about the same chances to become male or female. But especially for small populations (in the current analysis small means ≤ 20) it sometimes happens due to the small amount of individuals that one gender dominates immensely and this leads to a worse convergence for the following generations. The scenarios where this can happen can be divided into two cases.

1) While creating the random initial population.
2) In every generation while creating the offspring.

For both cases, a different strategy has been selected to maintain the balance. For the creation of the random initial population it is recommended to balance them directly at the beginning which increases the convergence over the following generations. Manipulating the initial population has already been shown to increase diversity in the past [36]. The additional computational cost to balance the initial population gender wise at the beginning is marginal compared to the computational effort over the whole optimization. Consequently if the random initial population is unbalanced, for example consisting of 90% male, a fixed amount of them should be removed and replaced with new random individuals until the population is balanced. Such an adequate starting point supports the handling of a potential unbalance during the offspring creation in later generations. The easiest thing is to replace individuals of the dominating gender during reproduction instead of replacing individuals independent of their gender. To be more specific, if the selection scheme decides to replace the losers of both genders' tournaments and males are currently dominating, the selection scheme could be adjusted to replace the loser and the second last male of the male tournaments and therefore does not replace the losers of the female tournaments which is the suggested strategy for case 2. Note that this has to be done for all tournaments held and consequently multiple individuals get replaced. This procedure can be transferred to other selection mechanisms as well.

D. Dealing With the Additional Calculation to Determine the Gender

Python is a widely used programming language in AI and it is known that for mathematical costly functions libraries written in C are used. For large data sets, the evaluation of the fitness function can become relatively expensive. In the context of GGEP the "cost function" has to be evaluated twice. For example $n = 2$ [see (1)] would be a standard choice for the fitness (or alternatively the magnitude of the error) while $n = 1$ [see (2)] would be needed to determine gender. To improve efficiency it is recommended to include both evaluations into one single loop by writing a user-defined C-library, as explained in https://docs.python.org/3/extending/extending.html. In this way calculating both values for a big amount of data points does not take any appreciable additional amount of time.

III. RESULTS

In order to describe the results first the test problems considered here are introduced in the following section.

A. Test Problem Description

Three different symbolic regression problems with increasing complexity were chosen, where the success of an individual can be easily measured. The first two of them represent 1-D symbolic regression problems of Ferreira's paper [31] where GEP was first introduced and the third one is a 2-D problem which is used as an introductory example in the GEP code developed by Weatheritt and Sandberg [33]

$$z = x^4 + x^3 + x^2 + x, \quad \text{for } x \in [2, 20] \tag{4}$$

$$z = 5x^4 + 4x^3 + 3x^2 + 2x + 1, \quad \text{for } x \in [1, 10] \tag{5}$$

$$z = x^2 + xy - 2y^2, \quad \text{for } x, y \in [-1, 1]. \tag{6}$$

With increasing problem complexity it can be observed that the introduction of gender increases the performance of the GEP code. The same phenomenon has been observed for other, more complex, optimization problems not discussed in this work which serves as an introduction of GGEP. While the first two problems consist of ten data points to be as comparable as possible to Ferreira's paper the third problem has 625 data points. Moreover, it will be shown that GGEP works increasingly better with an increasing population but also is superior with populations as small as 20. Additionally, this work shows that GGEP finds solutions in much earlier generations. To prove the supremacy of GGEP over GEP important factors to compare are:

1) the success rate;
2) the computational effort;
3) the ability to find solutions in early generations;
4) the ability to find good solutions in problems where an exact match does not exist in the function space available to GEP.

Here, the dominance of GGEP in regards of the first three factors will be demonstrated in detail using problems 1 to 3. The fourth criterion will be relevant for much more complex optimization problems, such as turbulent heat transfer or combustion modeling in the field of computational fluid dynamics [37], [38].

The success rate is defined as the percentage of runs in which the algorithm is able to undercut the fitness tolerance of this problem during a predefined number of generations. To compare both methods for one setup, both algorithms were run with 100 different random initial populations and the number of populations that end with an individual i in a population p with a fitness smaller than the fitness tolerance T consequently is the success rate. Another measure denoted as success speed includes the run time to compare the methods while giving respect to the time they need to come to a result

$$\text{success rate} = \sum_{k=1}^{100} s(k)$$

$$\text{success speed} = \frac{\text{success rate}}{\text{run time}}$$

$$\text{with } s(k) = \begin{cases} 0, & \text{for } \min(f_i) > T \ \forall i \in p \\ 1, & \text{for } \min(f_i) \leq T \ \forall i \in p. \end{cases} \tag{7}$$

The three problems have largely different function values and hence for the same relative error to be obtained a largely different target absolute error has to be chosen. The T values were chosen roughly in such a way that comparable relative errors will be achieved. A too high tolerance level would force the algorithm to stop too early without even coming close to a maximum, while a too stringent criterion could potentially force the algorithm to run forever. The present T values have been chosen as a compromise in terms of the relative error. These 100 initial populations evolve over many generations and due to the lack of a better word this will be referred to as 100 creations subsequently. Additionally, the run time is tracked and compared for both variants. It is to be expected that GGEP needs additional time to calculate and organize gender. All comparisons presented in the next section can possibly differ while changing the setups. But as a matter of fact most of the configuration parameters had statistically no or little influence on the dominance of GGEP over GEP. As an example, both algorithms have the same optimal value for the tournament size. On the contrary the population size, like mentioned before, seems to be an important factor and is consequently analyzed separately.

B. Comparison of GEP With GGEP

This section compares GEP and GGEP for the three test problems outlined before, for a range of parameters. Fig. 4 shows the success rate of both algorithms versus the population size for four different numbers of generations (i.e., 50, 100, 200, and 300). Whereas for problem 1 both GEP variants reach a 100% success rate (see Fig. 4) if the population size is big enough it can be clearly seen that for the more complex problems 2 and 3 GGEP outperforms GEP notably. Fig. 4 indicates that GGEP reaches higher success rates for all test problems. This trend increases with increasing test problem complexity and increasing population sizes. The only exception can be seen for problem 2 when using 50 generations where GGEP has a comparable or worse performance for smaller population sizes, whereas for population sizes reaching 200, GGEP again outperforms GEP. For 300 generations GGEP reaches nearly 90% success rate for big population sizes, whereas GEP peaks at about 70%. For the 2-D problem 3 the success rate of GGEP ends up being nearly twice the success rate of GEP for all numbers of generations. For problem 3 the results for GGEP after 50 generations are already slightly better than for GEP after 100 generations and GEP after 300 generations is broadly outperformed by GGEP after 200 generations and even more by GGEP after 300 generations overall population sizes.

A similar conclusion can be drawn for an increasing number of generations with fixed population sizes (60, 80, 140, 200) as shown in Fig. 5. Comparing the algorithms for problem 1 it can be seen that GGEP reaches a 100% success rate independent of the population size. On the contrary GEP peaks at a smaller success rate for all populations smaller than 200 and only reaches the 100% value for a population size of 200 (and bigger, but not shown). For problems 2 and 3, where the 100% cannot be reached it still can clearly be seen that GGEP reaches much higher success rates than GEP does. After about 100 generations the success rate of GGEP for a population size of 80 outperforms the success

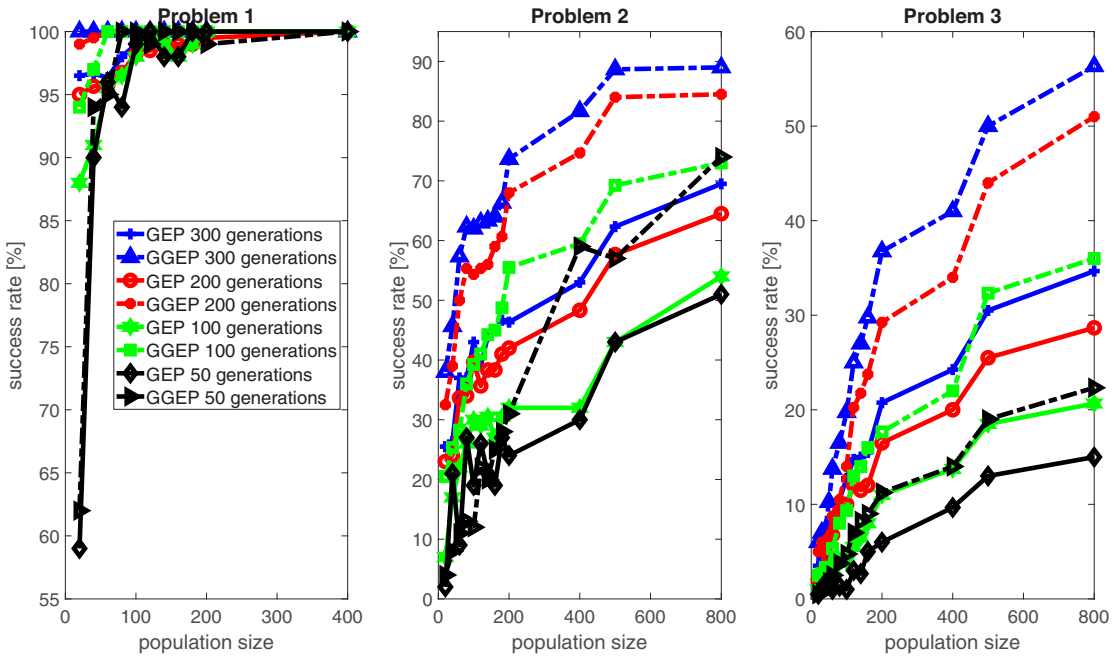

Fig. 4. Success rate of GEP and GGEP for problems 1 [left], 2 [middle], and 3 [right] for different number of generations over population size. Note the axes are adjusted for the different problems and do not have the same ranges for all three problems.

rate of GEP for a population size of 200 for problem 3. For problem 2 GGEP already reaches better results for a population size of 60 than GEP does for any of the shown population sizes. Additionally, it can be seen that the performance difference between the two algorithms increases with the increasing number of generations which suggests that GGEP can reach even better success rates for increasing numbers of generations.

Finally, in Fig. 6 the success speed of GGEP is divided by the success speed of GEP. It is expected that GGEP has, for the same problem, a higher run time than GEP due to the additional organization of gender. It can be seen in Fig. 6 that for all three problems GGEP reaches a higher success speed peaking at 1.5, 3.0, and 6.0 times the success speed of GEP for problems 1, 2, and 3, respectively. Additionally, GGEP reaches a higher success speed for the majority of the presented combinations of population sizes and generations. Only for problem 1 a bigger area occurs for small population sizes and generations, where GEP reaches a slightly higher success speed than GGEP does. For the more complex problems 2 and 3 GGEP outperforms GEP for the wast majority of configurations. For problem 2 (problem 3) GGEP reaches up to a three (six) times higher success speed than GEP does while it reaches about 1.5 (twice) the success speed for the majority of the shown configurations. Finally, note that the regions where GGEP outperforms GEP vary from 1 to 6 green indicating that the difference can be significant. On the contrary, the regions where GEP outperforms GGEP in all three problems only reaches values between 1 and 0.75 which indicates a more insignificant difference.

As a conclusion from analyzing Figs. 4–6 it is evident that GGEP outperforms GEP for all problems considered here, based on the success rate for a wide range of the number of generations and population sizes.

Besides success rates the mean fitness value after a defined number of generations for different population sizes is an important performance indicator and is shown in Fig. 7, where the mean value is taken from 100 runs/creations with different random initial populations of the GEP/GGEP codes. The mean value M is calculated based on the best (smallest) fitness f of all individuals i found in each creation k

$$M = \frac{1}{100}\sum_{k=1}^{100}\min(f_i). \tag{8}$$

As in this analysis, the error is being minimized in the context of survival of the fittest it is obvious that a smaller fitness value dominates a bigger one. Note that in Fig. 7 the y-axis for problems 1 and 2 is shown in a logarithmic scale to be able to recognize the wide range of fitness values reached by GEP. In Fig. 7 it can be especially seen that for problem 1 GGEP has always a mean value close to zero even if it has not a success rate of 100%, whereas GEP only reaches such small values in runs with a 100% success rate. This means that GGEP finds still good solutions in cases where it does not succeed. At the same time, GEP still finds solutions that are really far away from an optimal solution and therefore have a much higher mean fitness. While comparing the mean fitness values for problems 2 and 3 for GEP and GGEP it can be seen that the introduction of gender decreases the mean fitness values considerably. This reaches a level where the mean fitness of GGEP is a half (quarter) of the mean fitness value of GEP for a population of 500 (800). This is further illustrated in Table II in the columns best/worst mean fitness, where the best mean fitness value after 300 generations for a population size of 800 for problem 3 is 0.033 for GEP and 0.008

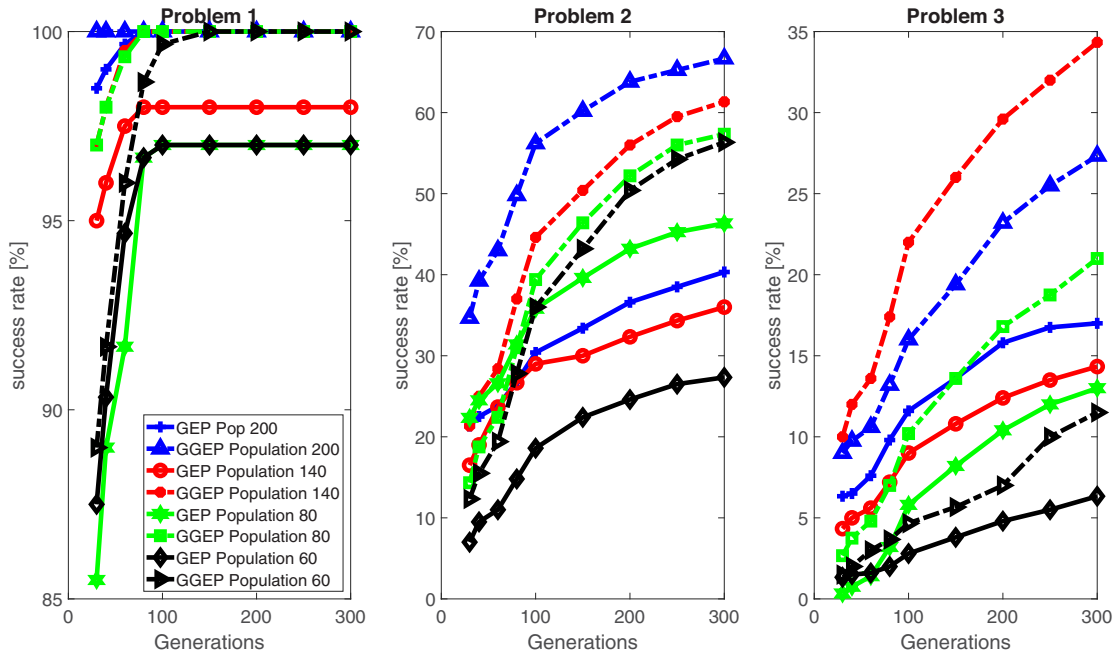

Fig. 5. Success rate of GEP and GGEP for problems 1 [left], 2 [middle], and 3 [right] for different population sizes over number of generations. Note the *y*-axis is adjusted for the different problems and does not have the same range for all three problems.

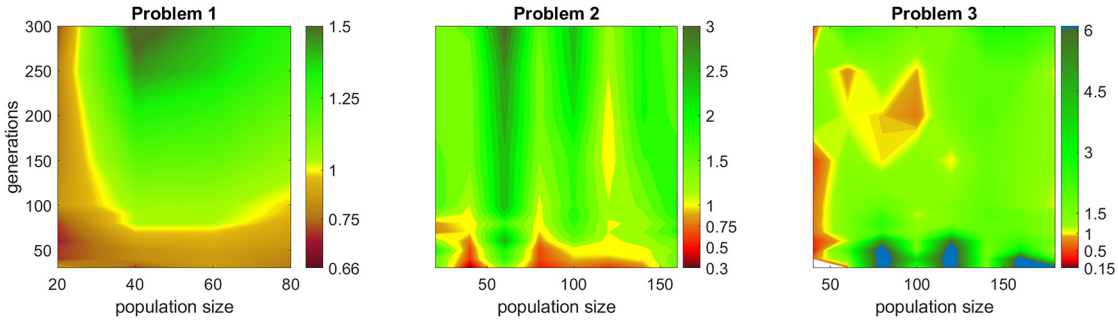

Fig. 6. success speed (GGEP)/success speed (GEP) for different generations and population sizes. Values in green bigger than 1 (red smaller than 1) indicate GGEP (GEP) has a higher success speed. Note the color bar range is adjusted for the different problems as well as the range of the population size. White values indicate that success speed(GEP)=0 and white ones that both methods have zero successes.

for GGEP. This trend again increases with increasing population size. Comparing the row best mean fitness of Table II this trend peaks at 3 to 4 times better mean fitness values at a population of 800 for problem 3 or even a 5000 times better mean fitness for problem 2. Table II summarizes important results of the comparison for all three problems. A not yet mentioned criterion that is presented in Table II is the calculation time increase for GGEP versus GEP. This value fluctuates for different population sizes, number of generations, test problems, and different random initial populations. Therefore, the range of the mean values over 100 creations for a given problem for the different configurations is shown. Note that the highest absolute additional run time was found to be 10 s for problem 2. The highest relative run time increase is seen for problem 1 from 4 to 7 s for GGEP which still can be considered as a small overhead given the performance increase of GGEP. For problem 3, where due to the increased data points the computational cost of the fitness evaluation increases, the additional run time becomes insignificant especially for big population sizes and the number of generations.

IV. Discussion

The previous results suggest that GGEP is able to reach higher success rates for relatively easy symbolic regression problems. However, it is very important that any evolutionary algorithm used in production should be able to find as good solutions as possible in real engineering problems as well. Preliminary tests in the field of turbulent combustion modeling [37], [38], not shown here, indicate that GGEP provides models that are at least 20% better error wise than those from GEP, which can be considered a strong indication that GGEP is ready to be used in productive applications. The question remains why GGEP reaches such high levels of performance and effectiveness. One obvious assumption is that the diversity is increased by splitting the population into male and female. Drezner and Drezner [28] mentioned in

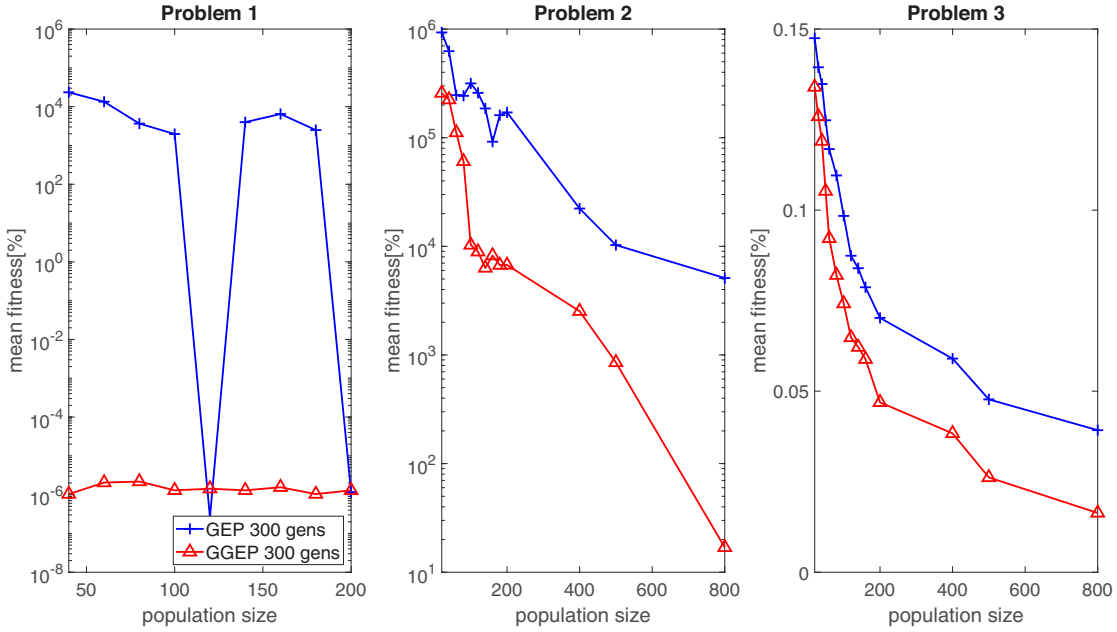

Fig. 7. Mean fitness values of 100 creations after 300 generations of GEP and GGEP for problems 1 [left], 2 [middle], and 3 [right] over populations. Note that for problems 1 and 2 the y-axis is presented in a logarithmic scale.

TABLE II
SUMMARY OF THE MAIN RESULTS FOR THE THREE STANDARD PROBLEMS. IF RANGES ARE DISPLAYED THEY ARE FROM THE WORST TO THE BEST CASE MEANING POPULATION 20 TO 800 FOR 300 GENERATIONS. NOTE THAT IN THIS ANALYSIS THE FITNESS IS MINIMIZED MEANING SMALLER VALUES ARE BETTER IN THIS FITNESS CONTEXT

Property	problem 1		problem 2		problem 3	
	GEP	GGEP	GEP	GGEP	GEP	GGEP
success rate ranges	$59-100\%$	$62-100\%$	$0-70\%$	$0-92\%$	$0-39\%$	$0-67\%$
Best mean fitness	$3\cdot10^{-7}$	$4\cdot10^{-24}$	5158	1	0.03	0.008
Worst mean fitness	23402	10^{-6}	930516	351085	0.17	0.14
Calc time	$2-4s$	$2-7s$	$12-88s$	$13-98s$	$18-423s$	$22-453s$

this regard "it is clear that the gender-specific algorithm slows down the homogenizing process and maintains more diverse populations" where they added gender by random. This should increase the diversity less than adding gender by the sign of the error, because first, GGEP tries to balance the percentage of males and females in the population and this consequently leads to a balance of individuals with positive and negative errors (as explained in detail in Section II). Second, by crossing an individual with a positive and one with a negative error it should be clear that the combination of these two possibly comes closer to the exact solution. This already happens in a classical EA where two good solutions with good fitness values reproduce. But with the introduction of the gender via the sign of the error this effect increases because the most influential features differ more substantially.

Moreover, GGEP, in a certain sense, has some elements of a multiobjective optimization. A population that wants to survive the evolution not only needs to evolve individuals with good fitness values but finally needs to keep individuals from both genders alive. Therefore, one gender needs to make room for the other if the population is unbalanced. This leads to weaker individuals having a possibility to survive because they just have to compete with individuals of their gender. As a result, it is possible to have a high selection pressure for one gender but at the same time a higher diversity and more genetic material.

V. CONCLUSION

In this work, an efficient way of introducing gender into evolutionary algorithms has been proposed and applied to three benchmark problems with an increasing level of complexity on the example of GEP. Gender has been defined via the sign of the error as a property of an individual rather than introducing it via the genome or by random. With this definition, evolutionary computation is not only brought closer to real evolution but it has been also demonstrated, that with the same setups GGEP outperforms GEP overall concerning the success rate, the fitness of the best individuals, and the mean fitness values while keeping the additional computational effort low. This trend increases with increasing problem complexity and increasing population and generation sizes. It has been argued that the performance increase is caused primarily by two mechanisms. At first, by dividing the population into male and female an overall higher diversity is maintained and

second the error is minimized faster by combining individuals with a positive and a negative error during the creation of the offspring. Although, the first results are very promising future work has to confirm the benefit of using gender in EAs when applied to more complex engineering optimization problems.

Furthermore, the introduction of gender in this analysis is not yet brought to its limits and leaves space for additional enhancements that possibly make it even more similar to nature's evolution. An example that leaves room for improvements is how tournament-winning males and females choose their mating partner. Currently there is no particular mechanism which tournament-winning female reproduces with which tournament-winning male. Although this is something that can happen, as a matter of fact in nature mostly the most dominant male or female can choose his or her partner. Alternatively, the algorithm could be changed in a way that the most dominant female or male chooses the partner that fits itself best concerning the value of the error. Therefore, not only the sign of an individual's error would influence the reproduction but also its magnitude. Another idea could be to increase the dimension of gender by splitting the domain in half and calculating the sign of the error in each half of the domain leading to four genders $(+,+)$, $(+,-)$, $(-,+)$, and $(-,-)$ and match them accordingly. The dimension of gender could also be increased by splitting the domain in more than two sections but the computational effort to organize and balance more genders increases with an increasing dimension. Apparently, testing these ideas will need further analysis.

ACKNOWLEDGMENT

The authors thank R.D. Sandberg for providing the EVE code.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence* (A Bradford Book). Cambridge, MA, USA: MIT Press, 1992.

[2] S. Forrest, "Genetic algorithms: principles of natural selection applied to computation," *Science*, vol. 261, no. 5123, pp. 872–878, 1993.

[3] H. Pohlheim, *Evolutionäre Algorithmen: Verfahren, Operatoren und Hinweise für die Praxis*. Heidelberg, Germany: Springer, 1999.

[4] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992.

[5] R. Miikkulainen and S. Forrest, "A biological perspective on evolutionary computation," *Nat. Mach. Intell.*, vol. 3, pp. 9–15, Jan. 2021.

[6] D. Greiner, J. Periaux, D. Quagliarella, J. Magalhaes-Mendes, and B. Galván, "Evolutionary algorithms and metaheuristics: Applications in engineering design and optimization," *Math. Problems Eng.*, vol. 2018, Jan. 2018, Art. no. 2793762.

[7] I. Parmee, *Evolutionary and Adaptive Computing in Engineering Design*. London, U.K.: Springer, 2012.

[8] A. J. Johnson, E. Meyerson, J. de la Parra, T. L. Savas, R. Miikkulainen, and C. B. Harper, "Flavor-cyber-agriculture: Optimization of plant metabolites in an open-source control environment through surrogate modeling," *PLoS One*, vol. 14, no. 4, 2019, Art. no. e0213918.

[9] M. Schoepplein, J. Weatheritt, R. Sandberg, M. Talei, and M. Klein, "Application of an evolutionary algorithm to les modelling of turbulent transport in premixed flames," *J. Comput. Phys.*, vol. 374, pp. 1166–1179, Dec. 2018.

[10] Y. Zhao, H. D. Akolekar, J. Weatheritt, V. Michelassi, and R. D. Sandberg, "RANS turbulence model development using CFD-driven machine learning," *J. Comput. Phys.*, vol. 411, Jun. 2020, Art. no. 109413.

[11] J. Weatheritt, R. Pichler, R. D. Sandberg, G. Laskowski, and V. Michelassi, "Machine learning for turbulence model development using a high-fidelity HPT cascade simulation," in *Proc. Turbo Expo Power Land, Sea, Air*, vol. 2B, 2017, Art. no. V02BT41A015.

[12] R. D. Sandberg *et al.*, "Applying machine learnt explicit algebraic stress and scalar flux models to a fundamental trailing edge slot," *J. Turbomach.*, vol. 140, no. 10, 2018, Art. no. 101008.

[13] M. Reissmann, J. Hasslberger, R. D. Sandberg, and M. Klein, "Application of gene expression programming to a-posteriori LES modeling of a Taylor Green Vortex," *J. Comput. Phys.*, vol. 424, Jan. 2021, Art. no. 109859.

[14] S. H. Ling and H. K. Lam, "Evolutionary algorithms in health technologies," *Algorithms*, vol. 12, no. 10, p. 202, 2019.

[15] J. Secretan *et al.*, "Picbreeder: A case study in collaborative evolutionary exploration of design space," *Evol. Comput.*, vol. 19, no. 3, pp. 373–403, 2011.

[16] J. Lehman *et al.*, "The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities," *Artif. Life*, vol. 26, no. 2, pp. 274–306, 2020.

[17] D. M. Hofmann, "A genetic programming approach to generating musical compositions," in *Evolutionary and Biologically Inspired Music, ound, Art and Design*, C. Johnson, A. Carballal, and J. Correia, Eds. Cham, Switzerland: Springer Int., 2015, pp. 89–100.

[18] A. Arcuri and X. Yao, "A novel co-evolutionary approach to automatic software bug fixing," in *Proc. IEEE Congr. Evol. Comput. IEEE World Congr. Comput. Intell.*, 2008, pp. 162–168.

[19] A. Arcuri, "On the automation of fixing software bugs," in *Proc. Compan. 30th Int. Conf. Softw. Eng.*, 2008, pp. 1003–1006.

[20] P. A. Ballal, H. Sarojadevi, and P. Harsha, "Compiler optimization: A genetic algorithm approach," *Int. J. Comput. Appl.*, vol. 112, no. 10, pp. 9–13, 2015.

[21] N. M. Stevens, "A study of the germ cells of Aphis rosæ and Aphis ænotheræ," *J. Exp. Zool.*, vol. 2, no. 3, pp. 313–333, 1905.

[22] M. Zhang, S. Zhao, and X. Wang, "A hybrid self-adaptive genetic algorithm based on sexual reproduction and baldwin effect for global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 3087–3094.

[23] J. Lis and A. Eiben, "A multi-sexual genetic algorithm for multiobjective optimization," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1997, pp. 59–64.

[24] S. Sodsee, P. Meesad, Z. Li, and W. Halang, "A networking requirement application by multi-objective genetic algorithms with sexual selection," in *Proc. 3rd Int. Conf. Intell. Syst. Knowl. Eng.*, vol. 1, 2008, pp. 513–518.

[25] M. Świechowski, "A crossover that matches diverse parents together in evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf. Compan.*, 2021, pp. 233–234.

[26] D. Thierens and P. A. Bosman, "Optimal mixing evolutionary algorithms," in *Proc. 13th Annu. Conf. Genet. Evol. Comput.*, 2011, pp. 617–624.

[27] M. Ryerkerk, R. Averill, K. Deb, and E. Goodman, "A survey of evolutionary algorithms using metameric representations," *Genet. Program. Evolvable Mach.*, vol. 20, no. 4, pp. 441–478, 2019.

[28] T. Drezner and Z. Drezner, "Gender-specific genetic algorithms," *INFOR Inf. Syst. Oper. Res.*, vol. 44, no. 2, pp. 117–127, 2006.

[29] K. O. Stanleyand, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nat. Mach. Intell.*, vol. 1, pp. 24–35, Jan. 2019.

[30] S. Wagner and M. Affenzeller, "SexualGA: Gender-specific selection for genetic algorithms," in *Proc. 9th World Multi Conf. Systemics Cybern. Inform.*, 2005, pp. 76–81.

[31] C. Ferreira, "Gene expression programming: A new adaptive algorithm for solving problems," *Complex Syst.*, vol. 13, no. 2, pp. 87–129, 2001.

[32] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[33] J. Weatheritt and R. Sandberg, "A novel evolutionary algorithm applied to algebraic modifications of the RANS stress-strain relationship," *J. Comput. Phys.*, vol. 325, pp. 22–37, Nov. 2016.

[34] N. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proc. 1st Int. Conf. Genet. Algorithms*, vol. 183, 1985, p. 187.

[35] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms* (Decision Engineering). London, U.K.: Springer, 2010.

[36] L. Duan *et al.*, "The strategies of initial diversity and dynamic mutation rate for gene expression programming," in *Proc. 3rd Int. Conf. Nat. Comput. (ICNC)*, vol. 4, 2007, pp. 265–269.

[37] C. Kasten, J. Shin, R. D. Sandberg, M. Pfitzner, N. Chakraborty, and M. Klein, "Modelling subgrid-scale scalar dissipation rate in turbulent premixed flames using gene expression programming and deep artificial neural networks," *Phys. Fluid*, to be published.
[38] C. Kasten, J. Shin, M. Pfitzner, and M. Klein, "Modelling filtered reaction rate in turbulent premixed flames using feature importance analysis, gene expression programming and tiny artificial neural networks," *Int. J. Heat Fluid Flow*, to be published.

Christian Kasten received the master's degree in mathematical engineering from the Universität der Bundeswehr München, Neubiberg, Germany, in 2014.

He worked eight years as a lead Software Developer with German Air Force Department. His main research interests are the enhancement of machine learning approaches, especially evolutionary algorithms, and the application of those to engineering problems from the field of computational fluid dynamics.

Julian Fahr received the master's degree in aerospace engineering from the Universität der Bundeswehr München, Neubiberg, Germany, in 2021.

He contributed during his master thesis to this research project.

Markus Klein received the Diploma degree in mathematics from the University of Mannheim, Mannheim, Germany, in 1998, and the Ph.D. and Habilitation degrees in mechanical engineering from the University of Darmstadt, Darmstadt, Germany, in 2002 and 2009, respectively.

He worked six years with GM Powertrain Engine Simulation Department. Since 2012, he has been a Professor for numerical methods in aerospace engineering with the Universität der Bundeswehr München, Neubiberg, Germany. He has authored or coauthored over 130 research papers in journals a dozen book chapters and more than 160 conference papers. His main research interests are the development of efficient, time resolved numerical methods as well as physical closure models for turbulent flow predictions, including recent machine learning techniques. The spectrum of application covers complex flow phenomena, such as multiphase flow, reactive flow, aerodynamic flows, supersonic flows, and gas explosions.