EUROGEN 2021
14th ECCOMAS Thematic Conference on
Evolutionary and Deterministic Methods for Design, Optimization and Control
N. Gauger, K. Giannakoglou, M. Papadrakakis, J. Periaux (eds.)
Streamed from Athens, Greece, 17—19 May 2021

ASSESSMENT OF MULTI-OBJECTIVE AND COEVOLUTIONARY GENETIC PROGRAMMING FOR PREDICTING THE STOKES FLOW AROUND A SPHERE

Heiner Zille[1], Fabien Evrard[1], Julia Reuter[1], Sanaz Mostaghim[1], Berend van Wachem[1]

[1]Otto von Guericke University Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany
e-mail: {firstname.lastname}@ovgu.de

Keywords: Genetic Programming, Fluid Dynamics, Multi-objective Optimization, Cooperative Coevolution

Abstract. *Genetic Programming (GP) has been used in a variety of fields to learn the relationships between physical measurements of real-world problems. In this article, we combine different techniques from the area of evolutionary optimization and, particularly, GP to solve a fluid-dynamics problem: the Stokes flow around a rigid sphere. This serves as the starting point to explore the potential of applying different GP techniques to such complex physical problems. From the definition of the considered fluid-dynamics problem, six benchmark instances with different lengths and complexities are derived. We use single- and multi-objective GP methods and compare their performance using different objective functions. More precisely, we study how model complexity, correlation and the consistency with physical laws (i.e. physical units of measurement) can be included as different objective functions, and whether their inclusion is beneficial to the overall search process. In addition, we include the concept of Cooperative Coevolution, which maintains multiple independent populations of solutions, into our GP implementations and explore the capabilities and limitations of such coevolution-based optimization. We further propose a novel multi-phase approach, which alternates in the GP process between phases of traditional optimization and a mutation-only phase to reduce the model complexity. The results indicate that using multi-objective optimization is beneficial to the search process and can help finding numerically correct solutions to the problems, especially when including a transformed Spearman correlation as an additional optimization goal. Furthermore, the inclusion of the physical units of measurement also helps guide the GP toward numerically correct and physically meaningful equations. While the concept of coevolution did not lead to a superior performance in all cases, the precomputation of additional features, and the resulting reduction of the function set of the GP, leads to a drastic increase in performance and enables the algorithms to solve even the most complex of our benchmark instances. In the future, we aim to extend this research to more complex flows with multiple spheres and at higher Reynolds numbers, which involve a large number of input features.*

1 INTRODUCTION

GP has been applied to a variety of applications to relate the physics of real-world phenomena to their experimental measurements or numerical simulations. Typically in physics and engineering applications, the goal of the optimization is not only to numerically predict variables accurately, but also to gain insight into the underlying physics and relations between input variables. In that regard, the fact that GP produces human-readable mathematical models is an advantage compared to other approaches such as Neural Networks (NN). In this article, we combine different techniques from the area of multi-objective evolutionary optimization and GP to solve a classical fluid-dynamics problem: the Stokes flow around a rigid sphere.

The fluid flow around a fixed rigid sphere is governed by a set of non-linear partial differential equations, the Navier-Stokes (NS) equations, and its nature depends upon the Reynolds number, Re, a dimensionless quantity characterizing the ratio of inertial effects over viscous effects within the fluid. Owing to the non-linearity of the NS equations, there is no general analytical solution to this fluid-dynamics problem. However, when $\mathrm{Re} \to 0$ (also referred to as the Stokes limit, or Stokes flow) the NS equations can be approximately linearized, and an analytical solution to the steady-state flow over a sphere of radius a, subject to the far-field velocity $\mathbf{u}_\infty$, can be derived [20]. The streamlines of this flow are shown in Fig. 1.

The application of GP to this fundamental flow problem serves as a starting point for exploring the capabilities of different GP techniques in predicting the behavior of complex flows, such as particle-laden flows. To that end, we derive in this work six test problem instances with different lengths and complexities, as to benchmark and compare different GP approaches and their ability to recover the analytical expression of this flow. Four different algorithm variants using classical methods as well as techniques from the field of coevolutionary optimization methods are compared. In the field of evolutionary optimization, the co-evolution of different components of the solutions has become increasingly popular in recent years, and this concept has also been applied to GP [e.g. 19]. In addition to using the concept of Cooperative Coevolution (CC) and a special multi-Phase GP approach, we investigate the use of multiple objective functions for the optimization. The single-objective fitness optimization is compared with 2-, 3- and 4-objective versions of the different algorithms, resulting in 32 different algorithms to solve the benchmark instances. As far as physical or engineering applications are concerned, a trained model that explains the connection between input and output data should also be consistent with the laws that govern their evolution. This means that, even though numerically correct predictions can be obtained, for instance, by adding a length and a time, such solutions are not in line with physical units and therefore of little value. For this reason, we additionally formalize the consistency with the law of physics as an objective. We furthermore include the correlation and complexity of the model as two objective functions. Our experiments also discuss the possible influence of incorporating expert knowledge and the precomputation of features on the performance of the GP process. The insights provided by this study serve as a basis for exploring the capabilities of GP in solving complex fluid-dynamics problems.

The remainder of this article is organized as follows. In Section 2, we derive the Stokes flow around a rigid sphere and discuss its utility for the modeling of particle-laden flows. In Section 3, we briefly outline the basics of Genetic Programming and the approaches that have been used in the literature so far with regard to Cooperative Coevolution and the inclusion of physical measurements in the GP process. Subsequently, in Section 4, we introduce our proposed GP variants and the objective functions we use in the optimization process. The benchmark instances and their corresponding function- and terminal sets are outlined in Section 5. Finally, we shows the results of our experimental evaluation in Section 6, and the article is concluded in Section 7.

2 THE STOKES FLOW AROUND A SPHERE

Flows that are laden with particles are at the heart of many natural processes and engineering applications, ranging from the flow of sediments in river beds to the combustion of fuel and biomass in furnaces. Understanding and predicting such processes typically require the development of reduced models to estimate the hydrodynamics forces acting on a particle carried within a flow, based on the averaged properties of that flow. When viscous effects dominate inertial effects, the Reynolds number of a rigid spherical particle of radius a, subject to a far-field flow with velocity $\mathbf{u}_\infty$ ($\|\mathbf{u}_\infty\| = u_\infty$), satisfies

$$\mathrm{Re} = \frac{2\,a\,\rho\,u_\infty}{\mu} \ll 1\,, \tag{1}$$

where ρ and μ are the density and viscosity of the fluid, respectively. It this regime, it is possible to derive an analytical solution of the flow around the sphere, which is axi-symmetric about the direction of the far-field velocity. In a spherical coordinate system whose origin is the center of the sphere, and whose zenith is aligned with $\mathbf{u}_\infty$, this flow can be expressed in terms of the stream-function

$$\psi(r,\theta) = u_\infty \sin^2\theta \left(\frac{r^2}{2} + \frac{a^3}{4\,r} - \frac{3\,a\,r}{4}\right)\,, \tag{2}$$

which results in the velocity field

$$u_r(r,\theta) = \frac{1}{r^2\sin\theta}\frac{\partial\psi}{\partial\theta} = u_\infty\cos\theta\left(1 + \frac{a^3}{2\,r^3} - \frac{3\,a}{2\,r}\right)\,, \tag{3}$$

$$u_\theta(r,\theta) = -\frac{1}{r\sin\theta}\frac{\partial\psi}{\partial r} = -u_\infty\sin\theta\left(1 - \frac{a^3}{4\,r^3} - \frac{3\,a}{4\,r}\right)\,. \tag{4}$$

This, in turn, enables the direct calculation of the resultant of the fluid forces acting on that particle [20], which reads

$$\mathbf{F} = 6\,\pi\,a\,\mu\,\mathbf{u}_\infty\,. \tag{5}$$

As soon as inertial effects become significant or if other particles are present in the vicinity of the particle under consideration, it becomes impossible to derive such an analytical solution, and one must rely on the results of experiments or numerical simulations to estimate the fluid forces acting on the particle. Owing to the human-readable nature of the models it produces, GP has the potential to provide a new insight into the non-linear physics of the flow through ensembles of particles at varying Reynolds number, and can therefore lead to the development of better reduced models for studying the behavior of particle-laden flows. In the following sections, and with the aim to explore this potential, we apply GP to the case of the Stokes flow around a fixed rigid sphere and study its ability to recover the analytical expressions of its velocity field, as given in Eqs. (3) and (4). The six benchmark instances that we derive from the Stokes problem are described in detail in Section 5.

3 GENETIC PROGRAMMING AND RELATED WORK

Genetic Programming (GP) is a subfield of Evolutionary Algorithms (EA) and was popularized by the work of Koza [9]. EA makes use of the Darwinian principle of survival of the fittest

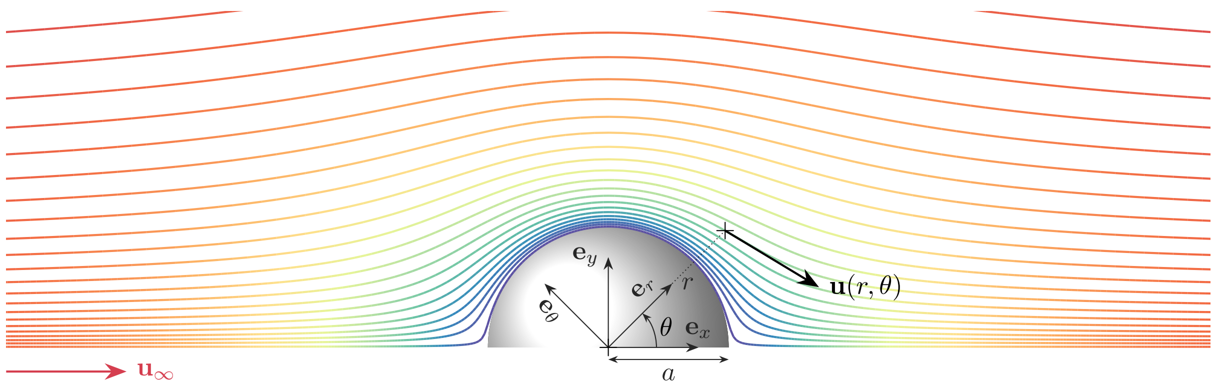


Figure 1: Streamlines of the Stokes flow around a rigid sphere, colored by the magnitude of velocity.

in order to optimize a given problem. Hence, a problem-dependent fitness function is used to evaluate the quality of a solution, usually in terms of how well the prediction of the GP model matches the training data that serves as input. Inside the GP process, evolutionary principles (selection, recombination of models, mutation operators) are used to perform a guided search towards optimal solutions. This way, promising individuals are created and refined iteratively until a stopping criterion is reached.

Evolutionary Multi-objective Optimization (EMO) is an important subfield of EA. Multi-objective Optimization Problems (MOOP) have not only one, but multiple objective conflicting functions that are to be optimized simultaneously. Mathematically, a multi-objective optimization problem can be formulated as follows:

$$\begin{aligned} \min \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x}))^T \\ s.t. \quad & \mathbf{x} \in \Omega \end{aligned} \tag{6}$$

A MOOP maps the search space Ω to the objective space $\mathcal{M}$ of dimension m. The definition of Ω depends on the encoding used for the problem. While in classical EAs, such encodings are often vectors of binary or real numbers, in the field of GP, the search space is the set of all possible models that can be created using the provided terminal symbols $\mathcal{T}$ and functions $\mathcal{F}$.

Since the problem consists of more than one objective, one single optimal solution cannot be ascertained anymore. Instead, the concept of Pareto-dominance is used in many of the existing EMO algorithms. A solution $\mathbf{x}_1$ is a solution that dominates the solution $\mathbf{x}_2$ if the following conditions are met [4]: (1) The solution $\mathbf{x}_1$ is no worse than $\mathbf{x}_2$ in all objectives, i.e. $f_j(\mathbf{x}_1) \leq f_j(\mathbf{x}_2)$ for all $j = 1 \ldots m$. (2) The solution $\mathbf{x}_1$ is strictly better than $\mathbf{x}_2$ in at least one objective, i.e. $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2)$ for at least one $j = 1 \ldots m$. The solution of a MOOP is a set of so-called Pareto-optimal solutions which are not dominated by any other solution in the search space Ω. By using the concept of domination together with other principles, EMO algorithms aim to provide an approximated set of Pareto-optimal solutions.

Applying the principles of evolution, GP aims at identifying a relation between a given input and output. A solution in GP is represented by a syntax tree consisting of terminals (input variables) and functions that are applied to them. This tree can be parsed into an executable program or equation. Special mechanisms for recombination and mutation on a syntax tree during the evolutionary process are required: For example, crossover can be performed by exchanging branches of two tree individuals; Mutation can be performed by replacing one node or branch by an other randomly chosen node or branch. For the fitness function, a distance measure between the produced and the desired output is often used such as the rooted mean squared error.

In the basic version of GP, the execution of the program manipulates numerical values without considering their physical units. This leads to situations in which non-physical solutions can be produced; a result which is of little interest for physics or engineering applications. The consideration of physical units inside GP was first proposed in [8], where special operators were applied in order to repair incompatible units. As an example: an operation adding a length and a time is repaired by artificially transforming a time into a length. This concept inspired multiple other works and has, since then, been implemented in single- as well as multi-objective GP algorithms [e.g. 1, 15, 6]. A different development line in dimensionally aware GP focuses on grammar-based GP, where constraints given by physical laws are already included in the rules of the programming grammar [e.g. 14, 18]. Another noteworthy concept was presented in [21], where strongly typed GP was used to generate viable method call sequences for software testing. This approach limits the search space only to feasible solutions and could be transferred to our problem by only allowing for dimensionally meaningful operations. A recent work on dimensionally aware GP was presented in [12], where an approach similar to ours is proposed: For each dimensionally incompatible GP operation, a penalty is calculated and aggregated throughout an individual. Minimizing this penalty value is an additional objective of the algorithm. Unlike the previously mentioned works, this method allows the initialization of physically non-meaningful solutions and enables them to develop towards physically meaningful solutions.

In most applications, GP has been used on a relatively small feature space, and research on high-dimensional GP for regression problems remains sparse. Some work has been done recently to include concepts like Cooperative Coevolution (CC) into the field of GP [e.g. 19], which has been used in many evolutionary algorithms as a successful optimization technique [e.g. 2, 23]. While CC, and more specifically the separation of solutions into multiple components, has mostly been used in the EA community as a technique to solve high-dimensional problems [22], CC-based approaches have been used in GP mostly as a way to boost performance, irrespective of the number of features. Applying Cooperative co-evolutionary GP (CCGP) on a problem requires (1) splitting the original feature space into smaller sets to create subsolutions as well as (2) merging partial solutions into one combined solution. In [19], the authors identified three different levels on which the merging of partial solutions in CCGP can be performed: genotype, feature and ensemble level. While fusion on a genotype level is the equivalent of the method originally proposed by Potter and De Jong [17] for evolutionary algorithms, fusion on feature [10, 11] and ensemble level [5, 13, 16] have been studied more extensively in the literature. The evaluation of the three proposed approaches on an image denoising problem showed that the ensemble method performed best on the given task. The authors also demonstrated that Cooperative coevolution can boost the performance of GP on high-dimensional problems compared to other modern, non-CC GP approaches.

4 PROPOSED METHODS

In this work, we explore the potential of various combinations of GP-techniques for solving Stokes Flow-related benchmarks. We use CC-based single- and multi-objective GP methods and compare their performance using different objective functions. More precisely, we investigate how length, correlation and the consistency with physical laws (i.e. physical units of measurement as, for instance, introduced in [8]) can be included as different objective functions, and whether their inclusion is beneficial to the overall search process. In total, we use four different variants of GP, along with 8 combinations of objective functions, which results in 32 different optimization methods. In the following paragraphs, we start by describing the four base-algorithm variants, before going into the details of the proposed objective function combinations.

4.1 Algorithm Variants

Standard GP: The first algorithm used in this article is a standard GP algorithm, abbreviated as *S-GP*, which is based on a classical $(\mu + \lambda)$-reproduction scheme and follows its implementation in the `deap`-framework[1]. The algorithm produces in each generation from a population of size μ a set of λ offspring solutions, which are then evaluated. The best out of the $(\mu + \lambda)$ solutions are taken over to the next generation of the algorithm. This environmental selection follows the procedure of the NSGA-II algorithm [3], while the selection of the parent-solutions for the reproduction are chosen at random. The new λ solutions are created either by repeatedly recombining two random solutions from the current population, or by applying mutation to a random individual. The mutation and recombination operators are selected at random from different operators every time a recombination/mutation is carried out.

For recombination, the algorithm chooses between two crossover operators: One-point crossover randomly selects a cut-off point in two individuals and swaps the selected branches. Leaf-based one-point crossover does so as well, while giving higher probability to leaves being exchanged between individuals rather than entire branches. For mutation, the algorithm chooses between uniform mutation, node replacement, insertion mutation and shrink mutation, all of which are chosen with equal probability. The uniform mutation replaces a randomly selected subtree by a randomly generated new branch. Node replacement, as the name suggests, chooses one node at random and replaces this function- or terminal-symbol with another one of the same arity. The insertion mutation creates a new subtree and inserts it at a random position, using the previous branch on that position as a subcomponent of the created subtree. Shrink mutation aims at decreasing the size of an individual by replacing a branch with one of its arguments.

Multi-Phase GP: The second algorithm is a Multi-Phase GP (*MP-GP*). The MP-GP is a variation of the S-GP above, which alternates between two phases of optimization behavior. In the first phase, the algorithm behaves identically to the S-GP. In the second phase, the crossover operators are disabled and the probabilities of the mutation operators to choose from are changed. More precisely, in this second phase of the optimization, the algorithm only applies node replacement mutation and shrink mutation, where the former is chosen with a probability of $2/3$ and the latter with a probability $1/3$. This two-stage approach avoids extensive growing of the models and allows refinements through a frequent mutation of the population. This proves particularly useful since not all of the algorithm variants employ the length of the individual as an objective, which in turn can lead to extensive growing of the trees. Another issue which was often observed in initial experiments is that solutions which only differ from an optimal model in one node may obtain a bad fitness, and using crossover is not likely to produce the required small change. Moreover, the replacement of one node may have a large impact on the actual fitness (for instance, changing an addition into a subtraction), which results in a less smooth fitness landscape. Therefore, enabling the algorithm to perform this second phase with a focus on small adjustments and shrinking the tree can be beneficial to find optimal solutions.

[1]`https://github.com/DEAP/deap`

Algorithm 1 Pseudocode of the Coevolutionary Genetic Programming (CC-GP)

```
Input: Training Data $X$, Terminals $\mathcal{T}$, Functions $\mathcal{F}$, Crossover probability $p_c$, Mutation probability $p_m$, Number of subpopulations $v$
Output: Set of non-dominated Solutions $S$
 1: $S_0 \leftarrow$ Random initial population of Solutions for upper-level GP
 2: $r_0 \leftarrow$ Choose random solution from $S_0$ as representative
 3: $A_0 \leftarrow$ Empty Pareto-dominance-based Archive
 4: for $i = 1$ to $v$ do
 5:     $S_i \leftarrow$ Random initial population of Solutions for $i-$th lower-level GP
 6:     $r_i \leftarrow$ Choose random solution from $S_i$ as representative
 7:     $A_i \leftarrow$ Empty Pareto-dominance-based Archive
 8: end for
 9: repeat
10:     for $i = 1$ to $v$ do
11:         if $i$ is part of $r_0$ then
12:             $S_i, A_i \leftarrow$ Optimize $S_i$ using standard GP for $k$ generations
13:             $r_i \leftarrow$ updateRepresentative($A_i$)
14:         else
15:             $r_i \leftarrow$ Choose random solution from $S_i$ as representative
16:         end if
17:     end for
18:     $S_0, A_0 \leftarrow$ Optimize $S_0$ using standard GP for $k$ generations
19:     $r_0 \leftarrow$ updateRepresentative($A_0$)
20: until total #Evaluations used
21: return $\bigcup_{i=0}^{v} A_i$
```

Cooperative-Coevolution-based GP: Our CC-based approach, abbreviated *CC-GP*, divides the problem into multiple, independent populations, with the intent that each population could, in theory, evolve different aspects of the original problem. As a result, multiple transformed features are created which serve as inputs to another, higher-level GP process. Lower- and higher-level GP populations are evolved cooperatively to enable the prediction of complex physical relationships in the data. This means that function evaluations are only carried out cooperatively, by taking a representative from each of the lower-level and upper-level populations and building a combined individual which is then evaluated using the training data.

The structure of the CC-GP algorithm is shown in Algorithm 1. In the main loop of the algorithm, each lower-level population (sometimes also referred to as subpopulation) is evolved using the standard GP method for a fixed amount of generations. After the last iteration of each population's optimization process, a representative solution is chosen from the current population, and this representative is used in the subsequent function evaluations of the other populations. After all lower-level populations have been evolved, the higher-level GP is used to optimize models that represent combinations of the representatives of the lower-level populations. The upper-level population uses only these representatives as input features, while the lower-level populations operate on the terminal set and input features that are given by the problem (i.e. that are also used in the above S-GP and MP-GP).

In order to use the computational budget more efficiently, the conditional statement of line 11 in Algorithm 1 is used to check whether the current, about-to-be-optimized subpopulation is part of the current representative of the upper-level population. If that is not the case, then the optimization of the lower-level population would be meaningless, since all function evaluations would return the same values (as they only depend on the values of all other representatives). Therefore, we assume that if a population i was not used in the current upper-level GP, the feature that was given by the representative of subpopulation i may not have been useful, and we replace it with a new, random representative.

Multi-Phase CC-GP: The last algorithm considered in this work, *MP-CC-GP*, is a combination of the two methods described above. It follows the same structure as the CC-GP, with the addition that it also alternates between the two optimization phases. The switch between the normal optimization (including crossover) and the mutation-only phase is done at the beginning of the main loop, i.e. the algorithm optimizes each lower-level and the upper-level population once with crossover and mutation, followed by one iteration with mutation only, as described above.

4.2 Objective Function Specification

To examine the effect of different objective functions, and the benefit of using multi-objective GP as opposed to a single-objective version, we define four different fitness functions. They are related to the quality of the prediction, the correlation with the training data, the physical correctness of the produced equations, and the complexity of the model. These four fitness functions are used in different combinations in the experiments below.

Maximum Absolute Error (f_1): The first objective function is related to the error between the prediction of the GP-produced model and the actual, expected output in the data set. Since we are interested in an accurate prediction of the physical relationship between the data, we decided to use the maximum absolute error over the data set, i.e. during the training, the value of f_1 is the absolute error of the worst data point within the training data set. By using the maximum error, we obtain a solution which guarantees that all predictions have at least a certain quality. Preliminary experiments have shown that this leads more often to exact and physically correct models compared to the rooted mean squared error.

Transformed Spearman Correlation (f_2): The second objective function is a transformed version of the Spearman Correlation ρ between the data set and the predictions. By using this correlation coefficient to guide the optimization, the GP algorithm may favor models which themselves are not numerically accurate, but are highly correlated with the desired output. Such models can, for instance, be multiplicative or additive components within a numerically correct model. A multi-objective EA optimizing f_2 has the ability to keep such models in the population over the generations, which can then be used in subsequent recombination steps to approximate numerically correct results. As an example, the term $u_\infty \cdot \cos(\theta)$ in Benchmark Instance 3 (see below) would show a high correlation with the data set, even though the actual prediction of the outcome u_r may still be poor. An EA which optimizes with regard to f_2, however, may consider this as a good solution and keep it in the population for further refinement.

To utilize the correlation coefficient in the optimization, we transform its value to fit our requirements. The range of ρ lies between -1 and 1, with a high coefficient indicating a strong positive (non-linear or linear) correlation, and a low value of ρ indicates a strong negative correlation. Both cases are, however, useful for the optimization, as building blocks with a strong negative correlation may still be helpful when used in a subtraction, for instance. In addition, our objective function are always minimized, which leads to a transformed objective $f_2 := 1 - |\rho|$. The worst possible value of f_2 is 1.0, which indicates no correlation at all, while smaller values of f_2 indicate some kind of connection (positive or negative) between the model and the data set.

In the CC-based approaches, a small adjustment is made to this objective: When the optimization of one of the lower-level populations is carried out, the value of f_2 is computed using the output of only the current individual of the subpopulation (which represents only a branch within the whole, combined tree model). In contrast, the upper-level GP uses the actual prediction from the combined model (i.e. the model that incorporates all lower-level representatives into the upper-level one). This way, the GP judges the features developed by the lower-level GP processes based on how correlated this feature is with the target data, and therefore obtains a more accurate measure of how useful the developed feature of this subpopulation is to the whole, combined, CC-based optimization.

Dimension Penalty (f_3): As explained previously in this article, it may be useful to guide the optimization as to find physically correct models, which (1) do not violate the laws of physics in their calculations and (2) predict the same unit of measurement as the unit of the target variable. To achieve both of these goals, we model the third objective function f_3 in the following way.

In our approach, contrary to the method used in [8], we do not use a special operator to adjust units to each other. Instead, whenever incompatible units are used in an operator, the operation is still carried out on the numerical values of the arguments, and the unit of the first input argument is used as the unit of the result of the node (e.g. in case a length and a time are added, the operation is carried out and the result is considered to be a length). To guide the search towards evolving physically meaningful equations, a penalty value of 1.0 is added every time such a nonphysical operation is carried out. This penalty value is accumulated along the tree together with the outputs of each function node. As a result, the root of the tree returns a value for the prediction of the training data, its physical units and a value for the total penalty occurring in the execution of that tree. Since we know that the expected result is a model that outputs, for instance, a velocity, any physical unit that is not expressed in `meters/second` is not meaningful for the purpose of the application. Therefore, an additional penalty based on the difference between the obtained units of the tree and the expected units of the output is added. This additional penalty is based on the distances between the exponents of the SI-base units (plus the unit radian for angles in this work) of which the final unit is composed. As an example, velocity can be written as $\mathtt{m}^1 \cdot \mathtt{s}^{-1}$. If our model output a result in the units $\mathtt{m}^2 \cdot \mathtt{s}^3$, the distances in the respective exponents are summed up as $|1-2| + |-1-3| = 1 + 4 = 5$. The value of the objective function f_3 is defined as the sum of the accumulated penalty during tree executing and the dimension-penalty for not matching the correct output units.

Length of Individual (f_4): The last of the four objective functions considers the length of the model, i.e. the number of nodes (function and terminal symbols) that are used to represent the model in its tree-based representation. This can be seen as a measure of the complexity of the model in terms of necessary mathematical operations. In the literature, the length of the solutions is often used in a weighted sum together with the fitness / error to avoid extensive growth of the trees. In our approach, instead, we employ the length as its own objective function. In the CC-based algorithms, the length refers to the length of the complete, combined individual, i.e. the model that incorporates all lower-level trees into the upper-level one.

4.3 Conflicts of Objective Functions

In contrast to the usual assumption in multi-objective optimization that the objective functions are somewhat in conflict with each other, this does not have to be the case between f_1 and f_2. It is clear that the model which perfectly optimizes f_1 will also produce an optimal value for f_2. This property may, however, not be true in earlier generations of the GP, since there exist optimal solutions with regard to f_2 with very poor performance in f_1. A similar expectation can be formulated for the relation between f_1 and f_3, although conflict may be stronger between these two functions. It is, on the one hand, clear that the optimal model for f_1 may, most probably, also be physically correct. This however is only true if there exists an actual physically correct relationship between the variables, which does not have to be the case for all applications. On the other hand, it is easy to construct solutions which are physically correct and match the units of the target variable, but perform horribly in a numerical sense with regard to f_1 and f_2. In conclusion, we can assume that there is, at least for sub-optimal solutions, a conflict between all four objective functions, although we can expect the conflict to vanish, the closer the models get to an optimal prediction of the data. It should also be noted that a vanishing conflict is, for our application, desirable, since the objectives f_2 and f_4 are merely tools to guide the algorithm to better solutions, while the objectives that are actually important for the application are f_1 and f_3.

5 BENCHMARK INSTANCES, FUNCTION SET AND TERMINAL SET

From the definition of the fluid-dynamics problem described in Section 2, we derive six problem instances with varying complexities, alongside with the used function and terminal sets. In the following, besides the variables and derived features of the problem, the terminal sets $\mathcal{T}$ include a set of constants, which we set as $\mathcal{C} = \{4, 3, 2, 1, \frac{1}{2}, \frac{1}{4}\}$.

Benchmark Instance 1: Predict $|\mathbf{u}|$ from u_x and u_y.

$$|\mathbf{u}| = \sqrt{u_x^2 + u_y^2} \tag{7}$$

Instance 1 is a rather simple equation and involves only 2 variables. To solve this instance, 4 computational operations are necessary (two multiplications or square-operations, one addition and one square root operation). The function set for this instance is given as $\mathcal{F} = \{+, -, \times, \circ^2, \circ^3, \frac{1}{\circ}, \sin(\circ), \cos(\circ), \sqrt{\circ}\}$ and the terminal set as $\mathcal{T} = \mathcal{C} \cup \{u_x, u_y\}$, where $\circ$ denotes the input of the unary functions.

Benchmark Instance 2: Predict u_x from u_r, u_θ and θ.

$$u_x = u_r \cdot \cos(\theta) - u_\theta \cdot \sin(\theta) \tag{8}$$

Instance 2 is more complex, first due to its length (3 variables and 5 computational operations), but also due to the inclusion of trigonometric functions. Compared to the other unary functions such as the square root, the power function or the multiplicative inverse, the trigonometric functions $\sin(\circ)$ and $\cos(\circ)$ are not monotonously increasing or decreasing with their input argument. Therefore, the fitness landscape in this problem instance may be more complex, and finding the optimal model to solve it may be more challenging for the GP variants. The function set for this instance is given as $\mathcal{F} = \{+, -, \times, \circ^2, \circ^3, \frac{1}{\circ}, \sin(\circ), \cos(\circ)\}$ and the terminal set as $\mathcal{T} = \mathcal{C} \cup \{u_r, u_\theta, \theta\}$.

Benchmark Instance 3: Predict u_r from u_∞, θ, a and r.

$$u_r = u_\infty \cdot \cos(\theta) \cdot \left(1 + \frac{a^3}{2 \cdot r^3} - \frac{3 \cdot a}{2 \cdot r}\right) \tag{9}$$

Instance 3 is the most complex and involves 4 variables and 13 computational operations (given its function set $\mathcal{F}$, defined below) in its simplified form as given in Equation (9). However, the GP may not necessarily end up with the simplification of multiplying the first two terms after the addition and subtraction in the parenthesis, and the expanded form of this equation needs 19 operations. Instance 3 further involves a trigonometric function, which adds the same additional complexity as described in Instance 2. The function and terminal sets for this instance are $\mathcal{F} = \{+, -, \times, \circ^2, \circ^3, \frac{1}{\circ}, \sin(\circ), \cos(\circ)\}$ and $\mathcal{T} = \mathcal{C} \cup \{a, r, u_\infty, \theta\}$.

Benchmark Instance 4: The basic relation between input and output variables in Instance 4 is identical to the one in Instance 1, i.e. the same equation needs to be learned. However, to examine the effect of preprocessing variables using expert knowledge or other computational techniques, we change the terminal and function sets compared to Instance 1. We use some of the unary function operators in the original set $\mathcal{F}$ and perform their operation of the input variables u_x and u_y. As a results, we obtain a smaller set of functions, but in exchange a much larger set of terminal symbols for the GP process. The sets for Instance 4 are $\mathcal{F} = \{+, -, \times, \sqrt{\circ}\}$ and $\mathcal{T} = \mathcal{C} \cup \{u_x, u_y\} \cup \{u_x^2, u_x^3, \frac{1}{u_x}, u_y^2, u_y^3, \frac{1}{u_y}\}$. Since we assume that the functions sin(∘) and cos(∘) can only be applied to angles in order to be physically meaningful, these two functions are not used on the velocities u_x and u_y. While the effect of a much larger terminal set is explored in our experiments (see Section 6.4), from a theoretical point of view Instance 4 is less complex than Instance 1, since only two operations are now necessary to assemble the correct equation.

Benchmark Instance 5: Instance 5 follows the same equation as Instance 2, but with changed function and terminal sets, as done in Instance 4. Again, trigonometric functions are applied to angles, while the other functions are applied to the velocities. As a result, we obtain $\mathcal{F} = \{+, -, \times\}$ and $\mathcal{T} = \mathcal{C} \cup \{u_r, u_\theta, \theta\} \cup \{u_r^2, u_r^3, \frac{1}{u_r}, u_\theta^2, u_\theta^3, \frac{1}{u_\theta}, \sin(\theta), \cos(\theta)\}$.

Benchmark Instance 6: As done in Instances 4 and 5, we derive Instance 6 from the equation of Instance 3 and change the function and terminal sets to $\mathcal{F} = \{+, -, \times\}$ and $\mathcal{T} = \mathcal{C} \cup \{a, r, u_\infty, \theta\} \cup \{a^2, a^3, \frac{1}{a}, r^2, r^3, \frac{1}{r}, u_\infty^2, u_\infty^3, \frac{1}{u_\infty}, \sin(\theta), \cos(\theta)\}$. As a result, while the terminal set is much larger than that of Instance 3, the complexity in terms of the necessary number of operations is smaller (10 operations).

For the above-mentioned benchmark instances, we create a data set of 366 data points using a far-field velocity $u_\infty = 1.0$ and a radius of the sphere of $a = 1.0$. The data set is randomly split in an 80/20 ratio, i.e. into 292 data points in the training set and 74 data points in the test set.

6 EVALUATION

In conducting experiments, we aim to compare various aspects of the algorithms and to compare the problems with each other. More precisely, we aim to answer the following questions:

1. How are the numerical performance and physical correctness of the results affected by the inclusion of additional objectives, especially of correlation and dimension-penalty? (Section 6.1)

2. How do the Cooperative Coevolution-based GP methods perform compared to the classical approaches? (Section 6.2)

3. How is performance affected by the multi-phase approach with additional mutation-only phases? (Section 6.3)

4. How is performance affected by the precomputation of features, and the corresponding changes in the function- and terminal-sets of the GP? (Section 6.4)

In this section, we answer these questions in detail by comparing the results of 32 combinations of algorithms and fitness functions on the six benchmark instances described in Section 5. The first objective function f_1 is always part of the optimization, and is paired with all possible combinations of the other three objectives, as seen in Table 1. For each algorithm variant and benchmark instance, we perform 31 independent runs. The parameters are set as follows: The population size is $\mu = \lambda = 2000$ for all algorithms. The probabilities for crossover and mutation are set to 0.5 and 0.5 respectively, except in the Multi-phase approaches, where no crossover is used and the mutation probability is set to 1.0 instead. The number of generations before a switch between normal optimization and mutation-only phase is set to 20. In the CC-based algorithms, all populations are optimized for 20 generations before moving on to the next one, and a switch of the phase in the CC-MP-GP is done once all lower-level and the upper-level GPs have been optimized once. The number of lower-level populations in CC-GP and CC-MP-GP is set to 2. In the leaf-based one-point crossover the probability to select a terminal symbol is set to 0.9. The maximum depth of the initial random solutions is set to 4. During the optimization, created solutions are restricted to a maximum length of 50 for the S-GP and the MP-GP, and a maximum length of 15 for the CC-based variants. The smaller limit in the CC-based algorithms applies to each population, but the combination of the representatives to form one final individual can, of course, still create much longer models. All experiments stop after $1,600,000$ function evaluations, i.e. after 800 generations of optimization. The algorithms are implemented using the `deap`-framework[2] version 1.3.1 [7] and the `pint` package[3] version 0.16.1.

In order to examine the quality of solutions, Table 1 shows the results of the experiments in terms of success rates, i.e. the numbers show how many out of the 31 runs achieved a certain goal. We compare two different types of success rates:

1. The first number represents the physical and numerical success rate. It shows how many runs have achieved a perfect numerical results, which we consider as $f_1 < 1\text{e}{-15}$ and at the same time delivered a model that satisfies $f_3 = 0$ and therefore delivered the correct units of measurement in a physically correct way.

2. The second number in each cell shows the numerical success rate only, i.e. how many models delivered a maximum absolute error of $f_1 < 1\text{e}{-15}$, regardless of an existing dimension penalty in f_3.

[2]`https://github.com/DEAP/deap`
[3]`https://github.com/hgrecco/pint`

	Objectives	1	2	3	4	5	6
S-GP	f_1	- / -	1 / 1	- / -	30 / 31	2 / 4	- / 17
	f_1, f_2	30 / 31	1 / 1	- / 5	31 / 31	3 / 5	- / 31
	f_1, f_3	2 / 2	- / -	- / -	31 / 31	18 / 18	7 / 7
	f_1, f_4	- / -	- / -	- / -	31 / 31	10 / 10	- / -
	f_1, f_2, f_3	31 / 31	8 / 8	13 / 13	31 / 31	25 / 25	27 / 28
	f_1, f_2, f_4	31 / 31	- / -	- / -	31 / 31	11 / 13	- / 4
	f_1, f_3, f_4	- / -	- / -	- / -	31 / 31	13 / 14	- / -
	f_1, f_2, f_3, f_4	31 / 31	1 / 1	- / 1	31 / 31	29 / 29	- / 6
MP-GP	f_1	- / -	- / -	- / -	28 / 30	3 / 4	- / 17
	f_1, f_2	30 / 30	3 / 3	- / 3	31 / 31	3 / 5	- / 31
	f_1, f_3	- / -	- / -	- / -	31 / 31	17 / 18	7 / 7
	f_1, f_4	- / -	- / -	- / -	31 / 31	3 / 3	- / -
	f_1, f_2, f_3	30 / 30	6 / 6	4 / 7	31 / 31	25 / 25	22 / 22
	f_1, f_2, f_4	28 / 28	- / -	- / -	31 / 31	4 / 4	- / -
	f_1, f_3, f_4	1 / 1	- / -	- / -	31 / 31	2 / 2	- / -
	f_1, f_2, f_3, f_4	31 / 31	- / -	- / -	31 / 31	14 / 14	- / -
CC-GP	f_1	1 / 1	- / -	- / -	16 / 17	9 / 11	- / 1
	f_1, f_2	29 / 31	- / -	- / -	29 / 31	5 / 7	- / 8
	f_1, f_3	4 / 4	- / -	- / -	26 / 26	14 / 14	- / -
	f_1, f_4	- / -	- / -	- / -	28 / 28	2 / 2	- / -
	f_1, f_2, f_3	27 / 27	6 / 6	- / -	29 / 29	24 / 24	2 / 3
	f_1, f_2, f_4	31 / 31	- / -	- / -	31 / 31	5 / 8	- / -
	f_1, f_3, f_4	1 / 1	- / -	- / -	22 / 22	11 / 12	- / -
	f_1, f_2, f_3, f_4	28 / 28	- / 1	- / -	29 / 29	15 / 17	- / -
MP-CC-GP	f_1	- / -	- / -	- / -	19 / 20	8 / 9	- / 3
	f_1, f_2	30 / 31	- / -	- / -	30 / 30	4 / 7	- / 9
	f_1, f_3	2 / 2	- / -	- / -	28 / 28	20 / 20	- / 1
	f_1, f_4	2 / 2	- / -	- / -	28 / 28	1 / 1	- / -
	f_1, f_2, f_3	28 / 28	5 / 5	- / -	29 / 30	21 / 21	- / 4
	f_1, f_2, f_4	30 / 30	2 / 2	- / -	29 / 29	3 / 4	- / -
	f_1, f_3, f_4	2 / 2	- / -	- / -	21 / 21	4 / 4	- / -
	f_1, f_2, f_3, f_4	29 / 29	- / -	- / -	29 / 29	12 / 12	- / -

Table 1: Success rates for the six benchmark instances in terms of numerically and physically correct results (first number) and numerically correct results only (second number). A value of 0 is shown as a dash.

In addition, we take a close look at the actual numerical results. Table 2 shows the average values obtained for f_1 over the 31 independent runs. For each of the benchmark instances (each of the columns), the respective best performance is marked in bold. A one-sided Mann-Whitney-U statistical test was performed for each algorithm to test whether its performance was significantly worse than that of the respective best algorithm for that benchmark. A difference is considered significant for $p < 0.05$ and is marked with an asterisk in Table 2, while all non-significant results are printed in bold font.

All analyses in this section are performed on the test set, which was not available to the algorithms during the optimization process. In general, the observed performance in all experiments does not differ significantly between the test set and the training set, which indicates that none of the algorithms suffers from a large amount of overfitting.

	Objectives	1	2	3	4	5	6
S-GP	f_1	4.37e-03 * (± 4.23e-03)	3.85e-02 * (± 2.09e-02)	2.18e-02 * (± 1.39e-02)	**0.00e+00** (± 0.00e+00)	5.00e-02 * (± 3.00e-02)	6.08e-03 * (± 7.48e-03)
	f_1, f_2	**0.00e+00** (± 0.00e+00)	3.62e-02 * (± 1.93e-02)	2.09e-03 * (± 2.52e-03)	**0.00e+00** (± 0.00e+00)	5.03e-02 * (± 2.99e-02)	**1.92e-16** (± 3.78e-17)
	f_1, f_3	1.09e-02 * (± 7.99e-03)	6.54e-02 * (± 3.04e-02)	1.86e-02 * (± 1.28e-02)	**0.00e+00** (± 0.00e+00)	2.96e-02 * (± 4.00e-02)	1.44e-02 * (± 9.99e-03)
	f_1, f_4	1.86e-02 * (± 1.49e-03)	1.35e-01 * (± 3.08e-02)	1.55e-01 * (± 2.67e-02)	**0.00e+00** (± 0.00e+00)	7.55e-02 * (± 5.41e-02)	8.53e-02 * (± 4.00e-02)
	f_1, f_2, f_3	4.03e-18 * (± 1.37e-17)	**1.46e-02** (± 2.00e-02)	**9.70e-04** (± 1.45e-03)	**0.00e+00** (± 0.00e+00)	**1.84e-03** (± 4.50e-03)	**5.78e-04** (± 1.78e-03)
	f_1, f_2, f_4	**3.58e-18** (± 1.96e-17)	8.61e-02 * (± 2.43e-02)	1.95e-02 * (± 9.48e-03)	**3.58e-18** (± 1.96e-17)	6.09e-02 * (± 5.43e-02)	3.70e-02 * (± 2.11e-02)
	f_1, f_3, f_4	1.90e-02 * (± 2.07e-03)	1.22e-01 * (± 2.19e-02)	1.33e-01 * (± 3.69e-02)	**0.00e+00** (± 0.00e+00)	5.52e-02 * (± 5.52e-02)	8.63e-02 * (± 3.16e-02)
	f_1, f_2, f_3, f_4	**7.16e-18** (± 3.92e-17)	3.03e-02 * (± 1.62e-02)	9.34e-03 * (± 6.90e-03)	**0.00e+00** (± 0.00e+00)	**1.01e-03** (± 4.31e-03)	2.10e-02 * (± 1.62e-02)
MP-GP	f_1	5.27e-03 * (± 4.12e-03)	5.21e-02 * (± 2.56e-02)	2.26e-02 * (± 2.91e-02)	**2.63e-05** (± 1.44e-04)	5.05e-02 * (± 2.60e-02)	6.47e-03 * (± 8.68e-03)
	f_1, f_2	**4.44e-17** (± 2.24e-16)	3.92e-02 * (± 2.57e-02)	2.58e-03 * (± 2.58e-03)	**0.00e+00** (± 0.00e+00)	5.42e-02 * (± 2.73e-02)	**1.98e-16** (± 3.51e-17)
	f_1, f_3	1.51e-02 * (± 7.37e-03)	1.00e-01 * (± 3.24e-02)	6.54e-02 * (± 4.94e-02)	**0.00e+00** (± 0.00e+00)	3.09e-02 * (± 4.11e-02)	1.65e-02 * (± 1.38e-02)
	f_1, f_4	2.03e-02 * (± 3.81e-03)	1.56e-01 * (± 3.64e-02)	1.57e-01 * (± 2.42e-02)	**0.00e+00** (± 0.00e+00)	1.11e-01 * (± 4.31e-02)	1.17e-01 * (± 4.82e-02)
	f_1, f_2, f_3	5.97e-05 * (± 3.27e-04)	2.52e-02 * (± 2.49e-02)	2.58e-03 * (± 3.07e-03)	**0.00e+00** (± 0.00e+00)	**3.39e-03** (± 9.65e-03)	1.96e-03 * (± 3.53e-03)
	f_1, f_2, f_4	1.02e-03 * (± 3.81e-03)	1.06e-01 * (± 2.65e-02)	3.41e-02 * (± 1.36e-02)	**0.00e+00** (± 0.00e+00)	1.00e-01 * (± 4.09e-02)	5.12e-02 * (± 1.37e-02)
	f_1, f_3, f_4	1.86e-02 * (± 4.15e-03)	1.42e-01 * (± 2.98e-02)	1.57e-01 * (± 1.34e-02)	**0.00e+00** (± 0.00e+00)	1.08e-01 * (± 4.06e-02)	9.24e-02 * (± 4.22e-02)
	f_1, f_2, f_3, f_4	**0.00e+00** (± 0.00e+00)	5.52e-02 * (± 2.82e-02)	1.60e-02 * (± 6.73e-03)	**0.00e+00** (± 0.00e+00)	3.72e-02 * (± 4.37e-02)	4.20e-02 * (± 1.70e-02)
CC-GP	f_1	9.16e-03 * (± 5.36e-03)	7.27e-02 * (± 3.30e-02)	4.18e-02 * (± 2.28e-02)	2.35e-02 * (± 1.04e-01)	9.93e-02 * (± 1.28e-01)	8.34e-02 * (± 1.16e-01)
	f_1, f_2	**2.64e-17** (± 1.35e-16)	4.31e-02 * (± 1.96e-02)	9.52e-03 * (± 6.18e-03)	**1.07e-17** (± 4.33e-17)	4.74e-02 * (± 3.32e-02)	1.61e-02 * (± 3.78e-02)
	f_1, f_3	1.40e-02 * (± 6.63e-03)	1.07e-01 * (± 4.94e-02)	8.54e-02 * (± 3.87e-02)	5.98e-04 * (± 1.81e-03)	6.59e-02 * (± 7.91e-02)	7.12e-02 * (± 9.03e-02)
	f_1, f_4	2.07e-02 * (± 3.55e-03)	1.53e-01 * (± 3.47e-02)	1.44e-01 * (± 3.36e-02)	2.15e-03 * (± 7.43e-03)	1.13e-01 * (± 5.18e-02)	8.77e-02 * (± 5.28e-02)
	f_1, f_2, f_3	6.74e-04 * (± 2.17e-03)	6.53e-02 * (± 6.11e-02)	1.92e-02 * (± 1.30e-02)	1.87e-05 * (± 1.03e-04)	1.08e-02 * (± 2.78e-02)	4.81e-02 * (± 7.53e-02)
	f_1, f_2, f_4	1.61e-17 * (± 4.72e-17)	9.85e-02 * (± 2.08e-02)	3.69e-02 * (± 1.60e-02)	**3.58e-18** (± 1.96e-17)	5.74e-02 * (± 4.23e-02)	5.23e-02 * (± 3.97e-02)
	f_1, f_3, f_4	1.70e-02 * (± 7.19e-03)	1.44e-01 * (± 5.49e-02)	1.27e-01 * (± 3.28e-02)	6.13e-03 * (± 1.19e-02)	1.01e-01 * (± 1.16e-01)	1.04e-01 * (± 6.34e-02)
	f_1, f_2, f_3, f_4	4.24e-04 * (± 1.38e-03)	5.14e-02 * (± 4.04e-02)	3.07e-02 * (± 1.48e-02)	1.15e-04 * (± 5.75e-04)	2.28e-02 * (± 3.38e-02)	7.93e-02 * (± 1.03e-01)
MP-CC-GP	f_1	1.15e-02 * (± 4.37e-03)	6.92e-02 * (± 2.74e-02)	4.97e-02 * (± 2.92e-02)	4.05e-02 * (± 1.45e-01)	8.34e-02 * (± 1.45e-01)	4.07e-02 * (± 6.11e-02)
	f_1, f_2	**1.07e-17** (± 4.33e-17)	4.23e-02 * (± 1.44e-02)	1.29e-02 * (± 6.70e-03)	1.75e-04 * (± 9.57e-04)	5.71e-02 * (± 7.75e-02)	1.07e-02 * (± 9.32e-03)
	f_1, f_3	1.65e-02 * (± 6.45e-03)	1.08e-01 * (± 4.66e-02)	9.18e-02 * (± 4.35e-02)	2.13e-03 * (± 9.95e-03)	4.46e-02 * (± 1.02e-01)	6.89e-02 * (± 9.01e-02)
	f_1, f_4	1.88e-02 * (± 6.60e-03)	1.53e-01 * (± 4.79e-02)	1.57e-01 * (± 2.08e-02)	1.75e-03 * (± 5.36e-03)	1.43e-01 * (± 9.91e-02)	1.20e-01 * (± 7.41e-02)
	f_1, f_2, f_3	4.49e-04 * (± 1.83e-03)	4.64e-02 * (± 4.10e-02)	2.30e-02 * (± 1.62e-02)	**1.43e-07** (± 7.82e-07)	4.93e-02 * (± 1.19e-01)	3.34e-02 * (± 5.81e-02)
	f_1, f_2, f_4	**5.93e-04** (± 3.25e-03)	9.11e-02 * (± 3.69e-02)	4.44e-02 * (± 1.97e-02)	**3.56e-04** (± 1.89e-03)	7.84e-02 * (± 4.43e-02)	8.29e-02 * (± 9.04e-02)
	f_1, f_3, f_4	1.82e-02 * (± 5.59e-03)	1.72e-01 * (± 4.21e-02)	1.51e-01 * (± 1.95e-02)	7.12e-03 * (± 1.12e-02)	1.33e-01 * (± 1.15e-01)	1.26e-01 * (± 9.17e-02)
	f_1, f_2, f_3, f_4	1.10e-03 * (± 4.18e-03)	8.58e-02 * (± 3.52e-02)	3.61e-02 * (± 1.50e-02)	7.93e-04 * (± 3.97e-03)	3.15e-02 * (± 7.69e-02)	8.20e-02 * (± 1.06e-01)

Table 2: Numerical results (average and standard deviation) over 31 runs for the six benchmark instances. Best results for each instance are shown in bold. A statistically significant difference ($p < 0.05$) is indicated with an asterisk.

6.1 The Effect of Adding Additional Objectives

First, we take a look at how the final performance, numerically and physically, is impacted by the inclusion of additional objectives.

The effect of f_2: For the S-GP, we can see in Table 1 that the (f_1, f_2)-version achieves much higher success rates than the single-objective version which only uses f_1, especially in Instances 1, 2 and 6. This effect is even larger when looking at the other combinations with vs. without f_2. S-GP with (f_1, f_3) can, for instance, only solve Instance 1 twice, and cannot solve Instances 2 and 3 at all, while the same algorithm with f_2 as an additional objective solves these instances 31, 8 and 13 times, respectively. Adding f_2 to the 3-objective version (f_1, f_3, f_4) increases the performance from 0 to 31 successful runs in Instance 1 and from 14 to 29 numerically successful runs in Instance 5. For both the S-GP and the MP-GP we observe that in none of the objective combinations and for none of the benchmarks has the inclusion of f_2 led to a decrease in the success rates, both numerically and physically. For the CC-based algorithms, we can observe a similar picture, with the only exceptions in Instance 5, where the (f_1, f_2) version does not perform better than using f_1 only. We can further observe that the effect of only adding f_2 to the f_1-version of any of the algorithms in Instances 2 and 5 (which share the same characteristics) is not as powerful as for the other problem Instances. This may indicate that the positive effect of optimizing the correlation is weaker when non-monotonous, trigonometric functions are involved. In conclusion, we can clearly see the benefit of optimizing correlation in almost all of the experiments, which supports our hypothesis that this enables the algorithm to appreciate and reuse smaller, correlated building blocks within the optimization process.

The effect of f_3: The impact of the inclusion of physical meaning in the optimization process is most interesting. In principle, this adds another layer of difficulty to the problem, as the expectations to an optimal solution are higher than only finding a numerical optimum. When we compare the (f_1)-only algorithms with their (f_1, f_3) counterparts, we can see that there are a number of instances for which there are positive effects, but the boost in performance is not as strong as the one observed for f_2. In some cases, the consideration of f_3 even has a negative effect on outcome, both numerically and physically. An interesting observation in this case is that the effect in Instance 5 is usually positive, and that the same is true for the most complicated Instances 3 and 6. Here, the (f_1, f_2) variants, especially, often did not find any physically correct equations, even though they were able to find numerical ones. After the inclusion of f_3, the (f_1, f_2, f_3) versions were able to find more numerically, and also physically correct models. A reason for this may lie in the created training data. In our generated data samples, the value for u_∞ is constant. Therefore, a purely numerical optimization does not need to multiply it to the model (see Eq. (9)) to obtain numerically correct results. A physically-aware algorithm however, can discover this relationship in the data based on the units of measurement, even though the data of the application may be biased in such a way. In general, even though a strictly positive effect can not be seen in all cases, the inclusion of physical units into the algorithm can be seen as an advancement. If the user is interested in gaining insight on the workings of the application and what the physical relations are between the input parameters, the slightly lower success rates may be a small price to pay for ultimately obtaining physically meaningful models.

The effect of f_4: Using the length of a model as an optimization criteria has, in most cases, a negative effect on the success rates of the algorithms. We can, however, observe a positive effect in some instances for the S-GP and the CC-GP. This makes sense since the additional mutation phase in the other two algorithms already aims to reduce the size of the models during the optimization, while the S-GP and CC-GP may still suffer from an excessive growth of the trees, and therefore have a higher potential to actually benefit from using f_4.

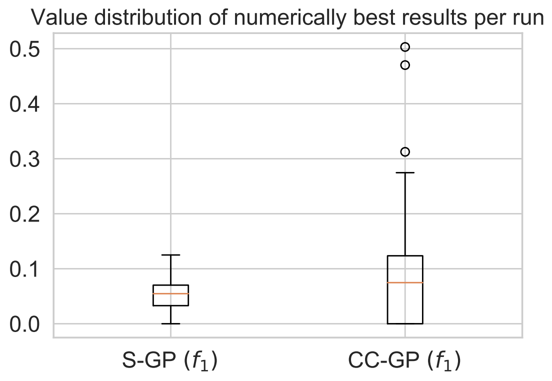
In summary, we can conclude that solving GP problems with multiple objectives is certainly superior to the single-objective version which only uses f_1. It is not strictly better to use any additional objective in all cases, but for every instance and for every algorithm there exists at least one of the multi-objective versions that performed strictly better in both success rates compared to the single-objective version. The largest benefit came from using f_2, which improved performance dramatically in almost all of the experiments.

6.2 Performance of Coevolutionary Algorithms

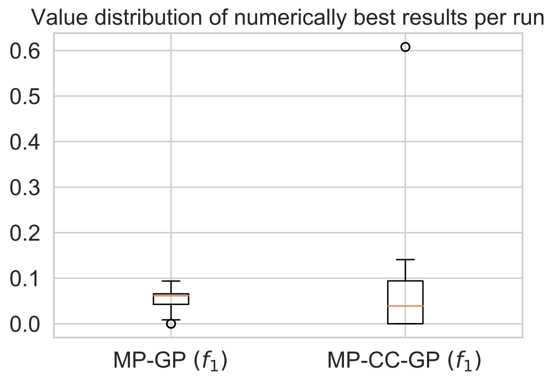
Next, we take a close look at the performance of the two CC-based algorithms in comparison with their traditional counterparts. Overall, we can not observe a strictly positive effect of using the coevolutionary approach in our experiments. For Instances 3, 4 and 6, all success rates of the CC-GP are lower than the ones of the S-GP, and similarly the results of the MP-CC-GP are worse than the ones of the MP-GP. We can, however, observe a certain impact for the Instances 1 and 5. In Instance 1 and one case of Instance 2, the success rates are slightly higher in the CC-based approaches. This impact, however, is not particularly strong, and the small changes in the success rates (e.g. 2 vs. 0 solved runs and 4 vs. 2 solved runs in the (f_1, f_3) algorithms) can also be caused by statistical effects. This is also supported by the fact that the average performances for these cases (see Table 2) do not deviate much, and in some cases the average performance is slightly lower even though less instances are solved correctly. Therefore, we can not observe a negative effect for Instance 1 as for some of the other instances, but the slightly higher success rates of the CC-algorithms can also not reliably indicate a superior performance. One case where we can observe a larger positive effect is Instance 5 when we perform a single-objective optimization with only f_1. The CC-GP variant solves $9/11$ instances (physically/numerically), while the normal S-GP only achieves success rates of $2/4$. On the other hand, we can see in Table 2 that the average fitness of S-GP is actually smaller than the one of CC-GP. To examine this effect in more detail, we show the boxplots of the numerically best results for this case in Fig. 2a. We can observe clearly that on the one hand, the CC-GP is able to find more results closer to the optimum, while on the other hand suffering from a larger spread of the solutions. The overall performance of S-GP is more robust over the independent runs, with a slightly smaller median and smaller spread. CC-GP is more successful overall, but in turn also produces results in some runs with a very poor performance. Looking at Fig. 2b, we observe a similar picture, with a higher success rate but larger spread and overall lower median performance in the CC-MP-GP compared to its counterpart MP-GP. This trade-off between higher success rates but lower robustness can be seen as an interesting property and, depending on the application, may be beneficial to the overall goal of the application.

6.3 Performance of the Multi-Phase Algorithms

Third, we analyze the effect of using the multi-phase approach in the algorithms. Overall, we can observe that adding the mutation-only phase does not increase the performance by large amounts in terms of success rates, but in most cases also does not deteriorate the performance



(a) Standard GP and CC-GP



(b) MP-GP and MP-CC-GP

Figure 2: Distribution of numerical results for benchmark instance 5 and the single-objective f_1-variants of the four algorithms.

of the algorithms. An interesting observation regards the combination with the length of the individuals, i.e. the optimization of f_4. In Instances 5 and 6, we observe that the success rates of MP-GP drop largely for all the algorithms that optimize f_4. This can be explained by the fact that the additional mutation phase, as described above, used only operators that help to reduce the size of the trees and therefore prevent excessive growth. If f_4 is one of the objectives, however, such operations result in a direct increase of one of the objective functions regardless of the quality in terms of the other objectives. This may lead to a premature convergence of the population to very short and simple individuals, from where it is more difficult to find the optimal solutions. The same behavior can also be observed when comparing CC-GP with MP-CC-GP. With the exception of these algorithms, a negative effect can not be observed for the problem instances 1, 4, 5 and 6. This is also reflected in the average performance values in Table 2, where in these instances the best performance of the S-GP is matched by the same algorithm configuration using the multi-phase approach (i.e. the difference between the same algorithm with and without this approach is not statistically significant). In Instances 2 and 3, which are the most difficult to solve in our experiments, the best performance is, however, only achieved by the S-GP, both in terms of success rates as well as average performance values.

6.4 The Effect of Precomputing Features

Lastly, we analyze the effect of feature transformation before the start of the GP process, i.e. we compare the Instances 1, 2 and 3 with the Instances 4, 5 and 6 respectively. As described in Section 5, the equations that need to be approximated are identical, but the last three instances contain a much larger terminal set in exchange for a smaller function set. Such derived features may, depending on the application, be obtained using expert knowledge or simply, as in this work, by taking physical properties of a problem into account. The results in Table 1 show that this precomputing step improves the outcomes significantly. For Instance 1, only algorithms which included f_2 in their optimization were able to solve this instance, with numerical success rates of 27 to 31 solved runs, while all other algorithms were often not able to even solve a single run. Once we precompute the features and solve the same problem with changed function- and terminal sets, all algorithm versions of the S-GP and MP-GP are able to solve Instance 4 perfectly with success rates of 30 or 31 runs. For the CC-based algorithms, we observe the same increase in performance, which enables all algorithms to perform superior or on par in Instance

4 compared to Instance 1. The difference between Instances 2 and 5 is even more prominent, as we see that Instance 2 was rarely solved by any of the algorithms, while Instance 5 is solved reasonably often by various algorithm combinations, most notably with an almost perfect score of 29 solved instances for the S-GP with (f_1, f_2, f_3, f_4). As a last result, we can confirm this trend also for the instance pair 3 and 6. The precomputation of the features enables the S-GP and the MP-GP (using (f_1, f_2)) to solve all 31 instances numerically correctly in Instance 6, while they achieved success rates of only 5 and 3 respectively in Instance 3. Based on these results, an interesting observation is that even though the feature transformation prior to the optimization is a very simple and computationally inexpensive measure, it changes the outcome of all GP algorithms drastically, and the magnitude of the performance enhancement is maybe only matched by the inclusion of the correlation f_2 into the optimization process.

7 CONCLUSION

In this article we have explored the possibilities of various different techniques in GP algorithms to solve the regression problem for a fluid-dynamics problem: the Stokes flow around a rigid sphere. We have derived six benchmark instances as examples of relationships that occur in this sort of problems. Apart from numerically correct predictions, the goal of this research was to produce models which can help researchers and engineers understand the underlying physical mechanisms governing such applications. Therefore, we included the physical correctness of the models, in terms of penalties for nonphysical operations as well as for not matching the desired output units of measurement, as an objective of the optimization. The inclusion of additional objective functions such as correlation and complexity of the model was further used to aid the GP process in reaching numerically and physically correct results. Moreover, we also explored different optimization techniques, such as the principle of Cooperative Coevolution and a multi-phase approach. Finally, the effect of a feature transformation prior to the optimization was investigated. The results of our experiments indicate that the ability to solve the benchmarks varies greatly with the usage of different optimization techniques. The greatest benefits in terms of numerical results were observed first, when the correlation f_2 was included in the optimization and second, when features were precomputed and the function-set $\mathcal{F}$ was reduced at the same time. Using these techniques, we observed a drastic increase in physically meaningful and numerically correct models produced by the GP. The usage of CC-based approaches and the multi-phase approaches, on the other hand, were not as beneficial to the overall outcomes. It must be noted however, that the CC-based algorithms in our experiments were only configured with two subpopulations, and other aspects such as the divisions of the terminal and function sets between the populations may require further investigation. In the future, we plan to apply the developed methods to fluid-dynamics problems with higher dimensionality – involving multiple spheres and varying Reynolds numbers – with the aim to provide helpful tools for understanding the complex physics of particle-laden flows.

References

[1] Sunith Bandaru and Kalyanmoy Deb. A dimensionally-aware genetic programming architecture for automated innovization. In *International conference on evolutionary multi-criterion optimization*, pages 513–527. Springer, 2013.

[2] Huangke Chen, Ran Cheng, Jinming Wen, Haifeng Li, and Jian Weng. Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations. *Information Sciences*, 509:457 – 469, 2020.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[4] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., USA, 2001.

[5] John A Doucette, Andrew R McIntyre, Peter Lichodzijewski, and Malcolm I Heywood. Symbiotic coevolutionary genetic programming: a benchmarking study under large attribute spaces. *Genetic Programming and Evolvable Machines*, 13(1):71–101, 2012.

[6] Marko Durasevic, Domagoj Jakobovic, Marcella Scoczynski Ribeiro Martins, Stjepan Picek, and Markus Wagner. Fitness landscape analysis of dimensionally-aware genetic programming featuring feynman equations. *arXiv preprint arXiv:2004.12762*, 2020.

[7] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, July 2012.

[8] Maarten Keijzer and Vladan Babovic. Dimensionally aware genetic programming. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*, pages 1069–1076, 1999.

[9] John R Koza and John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[10] Krzysztof Krawiec and Bir Bhanu. Coevolution and linear genetic programming for visual learning. In *Genetic and Evolutionary Computation Conference*, pages 332–343. Springer, 2003.

[11] Krzysztof Krawiec and Bir Bhanu. Visual learning by coevolutionary feature synthesis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(3):409–425, 2005.

[12] D. Li and J. Zhong. Dimensionally aware multi-objective genetic programming for automatic crowd behavior modeling. *ACM Trans. Model. Comput. Simul.*, 30(3), July 2020.

[13] Peter Lichodzijewski and Malcolm I Heywood. Coevolutionary bid-based genetic programming for problem decomposition in classification. *Genetic Programming and Evolvable Machines*, 9(4):331–365, 2008.

[14] Robert I Mckay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11(3-4):365–396, 2010.

[15] Yi Mei, Su Nguyen, and Mengjie Zhang. Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 435–447. Springer, 2017.

[16] John Park, Yi Mei, Su Nguyen, Gang Chen, Mark Johnston, and Mengjie Zhang. Genetic programming based hyper-heuristics for dynamic job shop scheduling: Cooperative coevolutionary approaches. In Malcolm I. Heywood, James McDermott, Mauro Castelli, Ernesto Costa, and Kevin Sim, editors, *Genetic Programming*, pages 115–132, Cham, 2016. Springer International Publishing.

[17] Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer, 1994.

[18] Alain Ratle and Michèle Sebag. Grammar-guided genetic programming and dimensional consistency: application to non-parametric identification in mechanics. *Applied Soft Computing*, 1(1):105 – 118, 2001.

[19] Lino Rodriguez-Coayahuitl, Alicia Morales-Reyes, Hugo Jair Escalante, and Carlos A Coello Coello. Cooperative co-evolutionary genetic programming for high dimensional problems. In *International Conference on Parallel Problem Solving from Nature*, pages 48–62. Springer, 2020.

[20] G. G. Stokes. On the effect of the internal friction of fluids on the motion of pendulums. *Transactions of the Cambridge Philosophical Society*, 9:8, January 1851.

[21] Stefan Wappler and Joachim Wegener. Evolutionary unit testing of object-oriented software using strongly-typed genetic programming. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, page 1925–1932, New York, NY, USA, 2006. Association for Computing Machinery.

[22] Heiner Zille. *Large-scale multi-objective optimisation: new approaches and a classification of the state-of-the-art*. PhD thesis, Otto-von-Guericke-Universitaet Magdeburg, Fakultaet für Informatik, 2019.

[23] Heiner Zille, Hisao Ishibuchi, Sanaz Mostaghim, and Yusuke Nojima. Mutation operators based on variable grouping for multi-objective large-scale optimization. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016.