An intelligent optimised estimation of the hydraulic jump roller length

Antonio Agresta$^{1[0000-0003-4037-7403]}$, Chiara Biscarini$^{1[0000-0003-2279-7244]}$,
Fabio Caraffini$^{2[0000-0001-9199-7368]}$, and Valentino
Santucci$^{1[0000-0003-1483-7998]}$

[1] University for Foreigners of Perugia, Perugia, Italy
{antonio.agresta,chiara.biscarini,valentino.santucci}@unistrapg.it

[2] Department of Computer Science
Swansea University, Computational Foundry, Swansea SA1 8EN, UK
{fabio.caraffini@swansea.ac.uk}

Abstract. In this paper, we address a problem in the field of hydraulics which is also relevant in terms of sustainability. Hydraulic jump is a physical phenomenon that occurs both for natural and man-made reasons. Its importance relies on the exploitation of the intrinsic energy dissipation characteristics and on the other hand the danger that might produce on bridges and river structures as a consequence of the interaction with the large vortex structures that are generated. In the present work, we try to address the problem of estimating the hydraulic jump roller length, whose evaluation is inherently affected by empirical errors related to its dissipative nature. The problem is approached using a regression model and exploiting a dataset of observations. Regression is performed by minimising the loss function using ten different black-box optimisers. In particular, we selected some of the most used metaheuristics, such as Evolution Strategies, Particle Swarm Optimisation, Differential Evolution and others. Furthermore, an experimental analysis has been conducted to validate the proposed approach and compare the effectiveness of the metaheuristics.

Keywords: Metaheuristics · Hydraulic jump roller length · Regression · Continuous optimisation · Sustainability

1 Introduction

River flow dynamics [1, 2, 10] falls under the category of very complex flow evolution, which includes the widest diversity of flow. Such processes are the main damage to river crossing structures.

In the 16th century, the great *Leonardo Da Vinci* observed a visible increase in height in a flowing liquid and, while documenting this phenomenon, associated its occurrence with an abrupt velocity change of its flow from high speed to lower speed. Only later, in the 1800s, Professor *Giorgio Bidone* (University of Turin) gave a mathematical formulation to the generic description of his illustrious predecessor.

The phenomenon *Hydraulic Jump* is a physics problem that has been studied for centuries due to the societal, optical, and technological implications that its solution can bring in terms of sustainability and resilience. Italy is a perfect case study to show this potential, as this country has a territory rich in large rivers and smaller watercourses, creeks, and brooks. However, structures such as bridges and embankments that often date back to decades had poor maintenance over the years. Recently, a consortium of universities named “FABRE” has been created to map and assess the health status, as well as to monitor the procedural schemes of hundreds of bridges to implement emergency recovery interventions.

Among the many dangers that affect bridge piers, the hydraulic jump is one of the most common, as it is intrinsically linked to the nature of the flow and its regime. In this article, we propose an approach based on a learning algorithm optimised with a heuristics technique to evaluate the hydraulic jump characteristics and tackle this problem.

Hydraulic jump is a phenomenon generated by a change in the flow regime from supercritical to subcritical, accompanied by high energy dissipation leading to an increase in the depth of the flow, as shown in Figure 1a. The increase in dissipation is due to the development of a complex multi-fold turbulent flow structure that causes significant energy losses. Often, hydraulic jump characteristics, analogously to what happens in compressible flow dynamics with the occurrence of shock waves, are affected by the presence of external elements such as obstacles downstream, typically bridge piles, and above all, the characterisation of the bed roughness (see Figures 1a and 1b).

As in the premisis, hydraulic jumps have been broadly investigated both due to their frequent occurrence in nature and their potential risk for man-made fluvial structures and, on the opposite front, for their possible use as energy dampers for hydraulic structures themselves [15].

From the seminal work between the end and the beginning of the 19^{th} century in terms of both analytical [4, 27] and experimental [31] aspects, the analytical form of the characteristics of the hydraulic jump originates from the equation of momentum balance. The early works cited considered the friction forces negligible compared to the others. In subsequent investigations [24] a more generalised study was conducted that included consideration of a more realistic velocity distribution and, therefore, partially took into account the resistance of the boundaries. Rajaratnam [23], Hughes and Flack [19] and Hager and Bremen [16, 14] went a little further by implementing the bed shear stress directly from the expression of the momentum equation. Since then, the main issue still is represented by the definition of this ”friction term” related in a general sense to the bottom shear stress, but as a matter of fact strictly depending on the different feature of the bottom surface and the flow regime. Quite recently researchers have made a specific effort trying to assess the implementation of the effects of the bed roughness both empirically [11, 5, 6] and by using teaching-learning-based optimisation techniques [20, 13] always distinguishing between the two main characteristics of the hydraulic jump, namely the ”Sequent Depth ratio”, which is the ratio between the two cross sections with successive depths h_1 and

h_2 as in fig. 1b and the "Roller Length", defined as the horizontal distance between the toe section with the flow depth h_1 and the roller end corresponding to the cross section h_2. A more specific and mathematically detailed description of those hydraulic jump flow characteristics will be given in the next section.

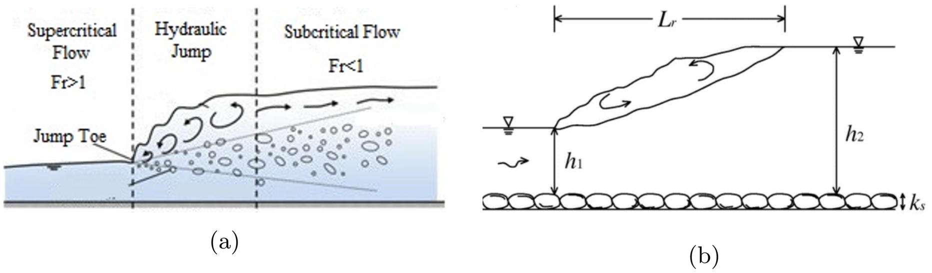
The scope of the present work is to focus on the Roller Length evaluation, estimating it via a novel form of the roughness contribution, modelled from the "shape" of the roughness function derived from the turbulence charts of the Nikuradze-type diagram [3]. Practically, we design a regression function with unknown parameters, which will be fitted to a dataset of observations by minimising a loss function through a black-box optimiser. In particular, we considered ten popular metaheuristics for black-box continuous optimisation and experimentally compared them through a repeated cross-validation approach.

2 Hydraulical aspects

The description given in section 1 shows how, due to the inherently dissipative nature of the hydraulic jump phenomenon, the analytical definition of some of its peculiar features is very difficult to obtain and empirical relations are needed based on the principles of the dimensional analysis. In what follows, a brief description of the characteristics of this phenomenon is presented from the hydraulic point of view.

2.1 The hydraulic jump over a rough surface

The hydraulic jump phenomenon generates a characteristic large vortex, or more correctly a *roller* whose length is directly proportional to the intensity of the phenomenon itself. The length of the roller can be defined as the horizontal distance between the toe section with height h_1 and the section where the roller ends with height h_2 as shown in Figure 1b.



(a)

(b)

Fig. 1: (a) Hydraulic jump physical schematics and (b) hydraulic jump roller length main characteristics.

Its length can be fairly easily evaluated in experimental tests visualising the flow and its free surface and measuring it, but it is still hard to determine it *a priori* by modelling a functional relation, especially considering the effects of the roughness of the bed. More rigorously, we can identify a dependence of the roller length L_r of the following type:

$$L_r = F(k_s, g, h_1, h_2, U_1, \mu), \tag{1}$$

where: k_s represents the roughness height at the bottom, expressed in cm; g is the gravitational acceleration; h_1 and h_2 are, respectively, the depths of the upstream and downstream flow, considering the transition from the supercritical (upstream) to subcritical (downstream) regime; U_1 is the mean flow velocity of the upstream cross section; while μ is the kinematic viscosity of the fluid.

Taking advantage of the bases of dimensional analysis, we can reduce the number of independent variables obtaining the following:

$$\frac{L_r}{h_1} = f_0\left(\frac{k_s}{h_1}, \frac{U_1 h_1}{\mu}, \frac{h_2}{h_1}, \frac{U_1}{\sqrt{g h_1}}\right). \tag{2}$$

The a-dimensional group $\frac{U_1}{\sqrt{g h_1}}$ is called the "Froude number" (Fr) and represents the relationship between the flow inertial forces and the external gravitational field. It serves as an indicator of the transition between the different flow regimes, namely subcritical, critical, and supercritical, respectively, for $Fr < 1$, $Fr = 1$, and $Fr > 1$. Another ratio between forces acting within the flow, and precisely against inertial forces and viscous ones, is represented by the "Reynolds number" (Re), indicated by the fraction $\frac{U_1 h_1}{\mu}$ which defines the transition from laminar to turbulent regime. In this case, since the Reynolds number is relative to the upstream cross section by the velocity U_1, it is taken for granted that it is in the presence of a fully developed turbulent flow, leading to the possibility of ignoring the viscous effects and hence the dependence on the Reynolds number itself. Therefore, the final form of the functional relation will be:

$$\frac{L_r}{h_1} = f\left(\frac{k_s}{h_1}, \frac{h_2}{h_1}, Fr_1\right), \tag{3}$$

where the ratio $\frac{h_2}{h_1}$ is defined as "Sequent Depth Ratio" (SDR) and we directly denote the Froude number with the subscript "1", pointing out that it refers to the upstream flow.

2.2 Definition of a roughness height modelling function

Many have tried for almost a century now to propose valid analytical forms for the relation (3), starting from the very well-known equation for SDR:

$$\frac{h_2}{h_1} = \frac{1}{2}\left(-1 + \sqrt{1 + 8Fr_1^2}\right), \tag{4}$$

obtained in 1828 by the French hydraulic engineer Jean-Baptiste Bélanger from the momentum balance equation. It appears clear right away how this equation does not imply the presence of a rough bottom, hence explicitly, of a shear stress.

Its generalisation including a direct dependence from the velocity distribution was given by Govinda et al. in 1966 [24]:

$$\frac{h_2}{h_1} = \frac{1}{2}\left(-1 + \sqrt{1 + \alpha Fr_1^2}\right), \tag{5}$$

where α also takes into account the condition of zero velocity at the bottom.

To integrate the action of bed shear stress, some authors (Carollo and Ferro, in 2004 [5, 6]) have included empirical parameters and, starting from the momentum balance equation, also proposed the following equation

$$\frac{h_2}{h_1} = \frac{1}{2}\left(-1 + \sqrt{1 + 8(1-\beta) Fr_1^2}\right), \tag{6}$$

where β is empirically defined as $\beta = 0.42\frac{k_s}{h_1}$, Although both α and β are related to the friction of the bed, the first through the velocity profile and the latter through the presence of the height of the roughness of the bed k_s.

Regarding the roller length L_r, Smetana in 1937 was the first to propose a direct correlation with the subsequent depth.

$$\frac{L_r}{h_1} = 6\left(\frac{h_2}{h_1} - 1\right). \tag{7}$$

Hager et Al. [14], via an experimental campaign, suggested the following relation for highly supercritical flows:

$$\frac{L_r}{h_1} = 8(Fr_1 - 1.5). \tag{8}$$

In both the above studies, the proportionality with *SDR* and *Fr* is suggested, but in neither case is there any dependence on the roughness of the bed. Hughes and Flack in 1984 [19], Ead and Rajaratnam in 2002 [11] and Carollo and Ferro in 2004 [5], verified the possibility to include such a relation via an empirical coefficient. In particular, taking into consideration both the sequent depth and the Froude number, Carollo and Ferro proposed the following.

$$\frac{L_r}{h_1} = \frac{a}{\left(\frac{h_1}{h_2}\right)^{1.272}} \tag{9}$$

$$\frac{L_r}{h_1} = b(Fr_1 - 1). \tag{10}$$

where the constants a and b, as expected, directly depend on the roughness of the bed. In 2007 the same authors, as a result of an experimental campaign, were appointed. found specific values for empirical coefficients, explicitly linking the roller length L_r to Fr_1 and k_s and simultaneously L_r to *SDR*, h_2/h_1.

Karbasi followed similar relationships, but with a different approach, in 2016 [20]. He proposed a teaching-learning-based optimisation algorithm to test different regression functions in order to find optimised values for empirical parameters to relate, in this case, the roller length L_r with Fr_1 and h_2/h_1, leaving out the explicit dependence on the roughness of the bed k_s.

In this work, we propose a dimension-based approach where the relation (3) is fully represented using both the results reported above and an evaluation of the roughness of the bed derived from considerations relative to the transition process between a smooth regime and a fully rough one. In the former condition, completely hydraulically smooth, viscosity acts as a damper and cancels out any perturbation caused by the roughness, while in the latter condition, it is the pressure drag on the rough surface that totally leads the process and produces friction. In the middle of these two regimes, a transitional phase occurs where the two processes are present. In order to grossly capture the complex physics described, a modelling function is suggested by Andersson [3] who basically reports the Nikuradse turbulence diagrams in the form of a wall function.

The original idea was to implement such a function, adjusting some of its terms, to the general form of the relation (3). The proposed roughness function equation is as follows:

$$\phi = \begin{cases} 0 & \text{if } k_s^+ \leq K_{SM} \\ \frac{1}{k_s} \ln[C_s k_s^+] \sin\left(\frac{\pi}{2} \frac{\ln k_s^+ - \ln K_{SM}}{\ln K_R - \ln K_{SM}}\right) & \text{if } K_{SM} \leq k_s^+ \leq K_R \\ \frac{1}{k_s} \ln[C_s k_s] & \text{if } k_s^+ > K_R, \end{cases} \tag{11}$$

where the new constants introduced here are K_{SM} and K_R (respectively, “K smooth” and “K rough”) that represent the lower and upper bound of the transitionally rough regime. According to literature they have been set as $K_{SM} = 2.25$ and $K_R = 90$. The variable calculated k_s^+ is the so called “Roughness Reynolds number” defined as $k_s^+ = (k_s U_1)/\nu$ with the viscosity of water $\nu = 0.00131$ kg/ms. The constant C_s, called the “Roughness constant” is a numerical re-tuning coefficient and was set at $C_s = 0.5$. To retrieve the mean flow velocity of the cross section U_1, we used the experimental data set extracted from the 2007 work of Carollo and Ferro [7]. In particular, from the values of the Froude number we can compute U_1 as follows:

$$U_1 = Fr_1 \cdot \sqrt{gh_1} \tag{12}$$

to be inserted in Equation (11).

The idea behind the choice of the type of function (11) was made to best resemble the shape of the functional trend of the roughness function in the three ranges defined by the Froude number, emulating the behaviour of the wall friction function.

By plugging Equation (11) into Equation (3), also taking into consideration Equations (9) and (10), we can derive our estimate for the roller length as follows.

$$\frac{L_r}{h_1} = a_1 \cdot \phi + a_2 \cdot \frac{h_2}{h_1} + a_3 \cdot (Fr_1 - 1.5), \tag{13}$$

where the vector $a \in \mathbb{R}^3$ represents three parameters imposed by the design and whose values can be learnt from the data.

3 The Learning Scheme

The roller length estimation function introduced in Equation (13) requires one to identify the values for the parameter vector $a \in \mathbb{R}^3$. With this aim, we can exploit the dataset of observed values provided in [7] in order to learn the a parameters. Therefore, we treat Equation (13) as a regression function and we apply the definition of the parameters a to a regression problem.

We denote by $x \in \mathbb{R}^4$ the vector of the following four values: $x_1 = k_s$, $x_2 = h_1$, $x_3 = h_2$, and $x_4 = Fr_1$. By also noting that ϕ, as defined in Equation (11), is a function of k_s, h_1 and Fr_1, we will use the notation $\phi(x_1, x_2, x_4)$. We can now define the regression function for the roller length as

$$g_a(x) = a_1 \cdot \phi(x_1, x_2, x_4) + a_2 \cdot \frac{x_3}{x_2} + a_3 \cdot (x_4 - 1.5). \tag{14}$$

Furthermore, we denote by $y \in \mathbb{R}$ the roller length values contained in the data set, that is, $y = L_r/h_1$. Therefore, the data set D considered in this work [3] is a set of pairs (x, y) as follows.

$$D = \left\{(x, y) : x \in \mathbb{R}^3 \text{ and } y \in \mathbb{R}\right\}. \tag{15}$$

It is now clear that the parameters a can be learnt by minimising the following loss function.

$$f(a) = \sum_{(x,y)\in T\subseteq D} \left(g_a(x) - y\right)^2, \tag{16}$$

where the training samples in T are a suitable subset of the entire data set D, which contains a total of 367 observed data samples.

4 Black-Box Optimisers

The loss function introduced in Equation (16) can be minimised by any black-box optimiser. In this section, we describe the ten metaheuristics considered in this work.

4.1 Random and Quasi-random Search

Trivial random search procedures are considered in this work as baseline methods. In particular, we denote by RS the random search procedure which generates a given number of solutions uniformly at random and, after evaluating them all, selects the best one. Similarly, we also consider the Scrambled Hammersley Search (SHS) method [8] which generates a quasi-random sample of low-discrepancy vectors that homogeneously cover the search space. Although being trivial, RS and SHS are interesting baseline methods for two reasons: all the solutions can be evaluated in parallel (at least in principle), and they allow us to derive indications about the smoothness of the search landscape at hand by comparing their effectiveness with that of other smarter algorithms.

[3] The data set is available with the supplementary data at `https://doi.org/10.5281/zenodo.7595510`.

4.2 Simple Evolution Strategies

Evolution strategies [33] are a family of evolutionary algorithms that evolve one or more incumbent solutions by means of one or more genetic operators. In this work we consider two simple evolutionary strategies which adopt the so-called (1+1) search scheme, i.e., they maintain a single incumbent solution which is iteratively mutated and the generated mutant becomes the new incumbent solution if and only if it is fitter than it.

Given the incumbent solution $x \in \mathbb{R}^d$, a mutant $y \in \mathbb{R}^d$ is generated as $y_i \leftarrow x_i + \varepsilon_i$, for any dimension $i = 1, \ldots, d$. In this work we term with the acronyms ES and CES the evolution strategies which perform the perturbation according to, respectively, a normal distribution or a Cauchy distribution (which is a fat-tailed variant of the normal distribution). More formally, we have $\varepsilon_i \sim N(0, \sigma_i)$ in ES, and $\varepsilon_i \sim C(0, \gamma_i)$ in CES. In our experiments we adopt the default configurations of the ES and CES implementations provided in the Nevergrad library [25], which sets $\sigma_i = \gamma_i = 1$ for any dimension $i = 1, \ldots, d$.

4.3 Covariance Matrix Adaptation Evolution Strategies

One of the most sophisticated forms of evolution strategies is the evolution strategy of covariance matrix adaptation (CMA) [17].

Unlike the simple evolution strategies previously described, the CMA maintains three entities: a mean vector $m \in \mathbb{R}^d$, a covariance matrix $C \in \mathbb{R}^{d \times d}$, and a step-size vector $\sigma \in \mathbb{R}^d$. In each iteration, N solutions are sampled from the (possibly) multivariate normal distribution $N(m, \sigma C)$. The solutions are then weighted on the basis of their fitness, and then used to update m, C and σ. For further details, we point the interested reader to [17].

In this work, we consider two standard implementations of this algorithm, namely CMA and DCMA: The former uses a multivariate covariance matrix, while the latter maintains a simpler diagonal covariance matrix. Furthermore, the CMA and DCMA implementations provided in the Nevergrad library [25] are adopted with their default configurations.

4.4 Differential Evolution

Differential Evolution (DE) [29, 28] is a population-based evolutionary metaheuristic which was originally proposed in [32].

The DE population is made up of N $d-$dimensional vectors $x_1, \ldots, x_N$ that can be uniformly initialised at random or by using a quasi-random generator such as the Scrambled Hammersley procedure that produces a low discrepancy sample of vectors [8]. In this work we adopt both initialization variants, which are referred to as, respectively, DE and QrDE.

The key operator of the algorithm is the differential mutation that, for each population individual $x_i \in \mathbb{R}^d$, produces a mutant vector $v_i \in \mathbb{R}^d$ as a linear combination of some other population individuals. One of the most popular DE mutation strategy is named "current-to-best" and is defined as

$$v_i \leftarrow x_i + F_1(x_{best} - x_i) + F_2(x_{r_1} - x_{r_2}), \tag{17}$$

where: x_{best} is the best population individual so far, F_1 and F_2 are the two scale factor hyperparameters of DE, and x_{r_1}, x_{r_2} are two randomly selected population individuals which are different between them and with respect to x_i.

After the mutation, the vector v_i is recombined with x_i. The most common recombination operator is *binomial crossover* that generates a trial vector $y_i \in \mathbb{R}^d$ by selecting each vector component from either v_i, with probability CR, or x_i, with probability $1 - CR$. The crossover probability $CR \in [0, 1]$ is a hyperparameter of the algorithm. Finally, if v_i is fitter than x_i, it replaces x_i in the next iteration population.

In our experimentation we adopted the default configuration of the DE implementation provided in the Nevergrad library [25], i.e., $N = 30$, $F_1 = F_2 = 0.8$, $CR = 0.5$.

4.5 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [30] is one of the most famous metaheuristic based on swarm intelligence principle, firstly proposed in [21].

PSO maintains a population of N particles. Each particle has a position $x_i \in \mathbb{R}^d$ in the solution space and a velocity $v_i \in \mathbb{R}^d$. The particles are statically connected among them, usually by means of a global topology.

As in DE, the population is evolved during a given number of iterations. At each iteration, the j-th component of x_i is updated according to

$$v_{i,j} \leftarrow \omega v_{i,j} + c_1 r_{1,j}(p_{i,j} - x_{i,j}) + c_2 r_{2,j}(g_j - x_{i,j}), \tag{18}$$

$$x_{i,j} \leftarrow x_{i,j} + v_{i,j}, \tag{19}$$

where: $p_i, g \in \mathbb{R}^d$ are, respectively, the best position ever visited by particle i and the global best position ever visited by the whole swarm; $\omega, c_1, c_2 \in \mathbb{R}^+$ are the so-called *intertial*, *cognitive* and *social* hyperparameters of PSO; while $r_{1,j}, r_{2,j} \in \mathbb{R}$ are randomly generated numbers in $[0, 1]$.

In our experimentation we adopted the default configuration of the PSO implementation provided in the Nevergrad library [25], i.e., $N = 40$, $\omega = 0.72$, $c_1 = c_2 = 1.19$.

4.6 Nelder-Mead

Nelder-Mead (NM) [12] is a mathematical optimization methodology which iteratively updates a simplex of $d+1$ vertices in the solution space. At each iteration, the simplex is updated by trying to replace the worst vertex with a new better solution obtained by means of four operations termed as *reflection*, *expansion*, *contraction*, and *shrink*. We refer the interested reader to [12] for their definitions.

In our experimentation we adopt the standard implementation of NM as provided in the Nevergrad library [25].

5 Experiments

5.1 Experimental setup

In order to analyse the proposed methodology, we conducted an experimental comparison among the ten black-box optimisers described in Section 4. Following the learning scheme described in Section 3, we have performed a k-fold repeated cross-validation [22], where the number of repetitions is $r = 10$ and the number of folds is $k = 3$. Then, for all $r \cdot k = 30$ cross-validation rounds, each algorithm is executed $s = 10$ times with a budget of 10 000 objective evaluations. Therefore, a total of $r \cdot k \cdot s = 300$ executions per algorithm were performed.

In any single execution, an algorithm evaluates the loss function by accessing only the data from the training set for the current cross-validation round. However, the optimised a values returned by the algorithm are used to evaluate the regression function on the data of the current test set, thus allowing one to compute the mean percent relative error (MPRE) as follows.

$$MPRE = 100 \cdot \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{y_i}, \tag{20}$$

where: $\hat{y}_i = g_a(x_i)$ is the predicted value computed based on the learnt values a and the observed input data x_i from the test set, y_i is the observed output value of the test set, while n is the number of samples in the test set. With slight abuse of notation, we also denote by MPRE the average mean percentage relative error of an algorithm in all cross-validation rounds. Moreover, we may also use the same measure related to the training set data, but in these cases we will clearly specify that.

The implementations of the selected algorithms available in the popular Python library called *Nevergrad* [25, 26] (version 0.5.0, the latest one at the time of writing) were adopted with their default parameters' settings. For the sake of reproducibility, in Table 1 we provide for each algorithm the name of the corresponding Nevergrad class.

Table 1: Names of the Nevergrad classes corresponding to the algorithms adopted in this work. The acronyms of the algorithms are defined in Section 4.

Algorithm's acronym	**Nevergrad's class**
ES	OnePlusOne
CES	CauchyOnePlusOne
CMA	CMA
DCMA	DiagonalCMA
DE	DE
QrDE	QrDE
PSO	RealSpacePSO
NM	NelderMead
RS	RandomSearch
SHS	ScrHammersleySearch

5.2 Experimental results

First, to validate our approach, we compare the training and test errors of the a values learnt by any single execution of all algorithms considered. In this regard, in Figure 2 we provide a scatter plot where: the abscissa and ordinate axes represent the percentage relative error on, respectively, the training and test data, any blue point in the graph is the result of a single algorithm execution, while the regression line through the points is drawn in black.

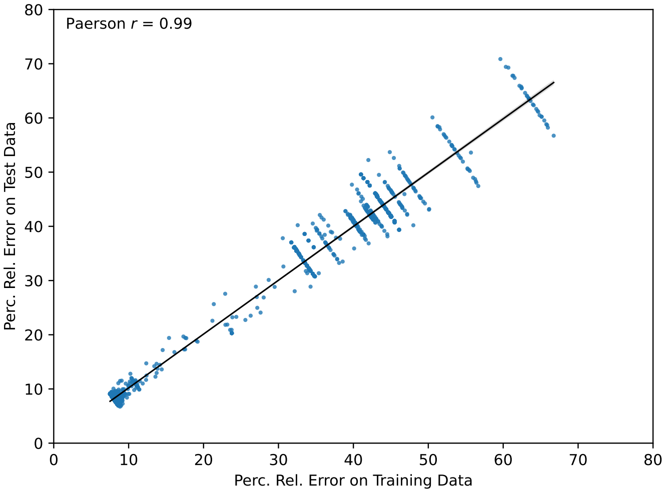


Fig. 2: Correlation between the mean percentage relative training and test errors.

Figure 2 clearly shows that training and test errors are strongly correlated – their Paerson correlation coefficient is 0.99 –, thus validating our approach for learning the regression parameters.

In Table 2 we provide all the statistics about the MPREs – evaluated on the test data – of the best solutions obtained by the ten algorithms in all their executions. The algorithms are ordered by median MPRE and the best results, for each statistic, are provided in bold.

To statistically validate the results, we also performed a statistical analysis on the test MPREs obtained by the algorithms. The omnibus Kruskal-Wallis test [9] rejects the equivalence of effectiveness among the five algorithms with a practically zero p-value (around 10^{-303}). Therefore, a Conover post hoc test has been performed considering the Benjamini-Hochberg adjustment scheme to mitigate the statistical family error rate [18]. The Conover p-values of the comparison between ES and all other algorithms are also provided in Table 2.

Table 2 shows that the eight proper algorithms are clearly better than the two baselines RS (random search) and SHS (a quasi-random sampling method).

Table 2: Statistics about the MPREs evaluated on the test data.

	Statistics on the MPREs					
Algorithm	**median**	**pvalue**	**mean**	**std**	**min**	**max**
ES	**8.27**	–	8.18	0.64	6.92	9.18
CES	8.28	0.98	8.18	0.63	6.95	9.16
PSO	8.29	0.90	**8.17**	**0.62**	7.10	**9.10**
NM	8.30	0.14	8.64	2.26	7.12	20.30
QrDE	8.30	0.53	8.24	0.71	**6.73**	11.26
CMA	8.31	8e-03	9.48	5.38	7.12	53.61
DCMA	8.32	0.02	9.44	5.00	7.01	40.21
DE	8.37	2e-03	8.88	2.72	6.89	30.11
SHS	41.84	6e-172	40.01	5.34	30.76	49.58
RS	42.13	6e-184	42.42	13.00	9.88	70.87

Moreover, the largest group of algorithms that are not significantly better – under the usual threshold of $\alpha = 0.05$ – than each other is made up of five algorithms, namely: ES, CES, PSO, NM and QrDE. Among these five algorithms, ES, CES, and PSO have a higher degree of robustness, as indicated by the standard deviation, minimum, and maximum of their MPREs. Their average MPRE is around 8.17% / 8.18%. The other statistics also show that these three algorithms are practically equivalent among them. Therefore, our main conclusion is that the two variants of the simple evolution strategy (ES and CES) and the PSO algorithm are to be preferred over the other competitors for the task under examination.

To provide a clearer picture, Figure 3 shows the box plot graph of the test MPREs obtained by the 300 executions of each algorithm.

Figure 3 mostly confirms the indications previously discussed. In fact, ES, CES and PSO are the only three algorithms which do not have outliers in the graph. This aspect is of particular interest by considering that the ordinate axis is on log scale. Moreover, it is also interesting to observe that the two baseline random search schemes (SHS and RS) are significantly worse than all the other approaches. This suggests that the search landscape of the optimised optimisation problem designed has a structure that is not chaotic.

For the sake of completeness, all pairwise comparisons among the algorithms considered are synthesised in the heat map of Figure 4. The entries are greenish or reddish when the row-algorithm is, respectively, better or worse than the column-algorithm. The green or red grades are set on the basis of the adjusted Conover p-values, which are also provided in the entries.

Figure 4 is a further confirmation of the conclusions derived, but also provides some additional indications as follows. The two covariance-based evolution strategies CMA and DCMA appear to be less effective than their simpler "cousins" ES and CES. We conjecture that this is probably due to the small search-space dimensionality of the problem under investigation. The plain DE

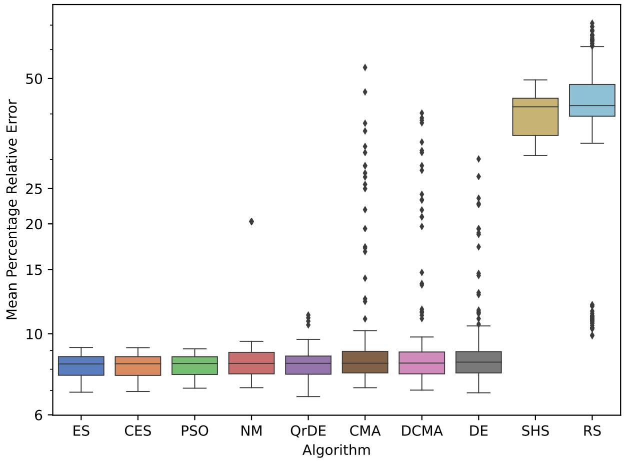


Fig. 3: Boxplot showing the distribution of the MPREs evaluated on the test data.

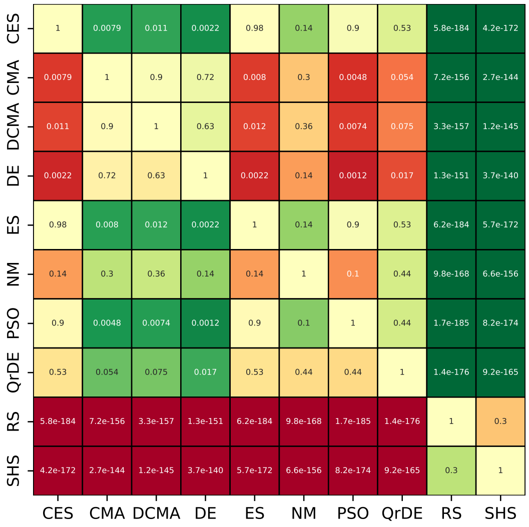


Fig. 4: Conover pvalues of all the pairwise comparisons among the algorithms.

is less effective than QrDE, suggesting that a quasi-random initialisation may improve the resilience to local optima in the DE search scheme.

As a last analysis, for each algorithm – with the exception of the two random search methods RS and SHS – we computed all pairwise Euclidean distances among the 300 a vectors obtained by the different executions carried out in the experimentation. Hence, the maximum Euclidean distances of all algorithms are provided in Table 3.

Table 3: Maximum Euclidean distance observed among the 300 solutions obtained by the different executions.

Algorithm	**CES**	**CMA**	**DCMA**	**DE**	**ES**	**NM**	**PSO**	**QrDE**
Max distance	0.67	4.48	4.19	13.39	0.85	6.47	**0.48**	3.81

These data confirm that the most robust methods in terms of MPRE, i.e., ES, CES and PSO, are also the most robust methods when analysed in the search space dimensions. In fact, also noting that the search space range was $[-50,+50]^3$, the maximum distance between the solutions produced by the 300 independent executions is much less than 1.

Finally, for reproducibility purposes, the source code, the full experimental results and the scripts used for the analysis were made publicly available at the following location: `https://doi.org/10.5281/zenodo.7595510`.

6 Conclusion and future work

We proposed a computational approach to a challenging hydraulic problem with important implications in terms of sustainability.

A contribution of this study is the newly proposed function for estimating the roller length of the typical hydraulic jump problem in water courses. This is designed so that the required parameters for evaluating can be learnt from observed data following a regression approach.

Furthermore, we show the minimisation process of the corresponding loss function with the aid of ten tested state-of-the-art black-box optimisation algorithms. To facilitate the activity of hydrology practitioners and researchers who may not be familiar with these methods, we used established algorithms with open implementation and ready to use from a well-known library [25]. The selected algorithms are from different families, such as simple and covariance-based evolution strategies, differential evolution variants, particle swarm optimisation, mathematical optimisation methodologies, and simpler random or quasi-random one-shot optimisers.

With a thorough experimental and validation phase, we show that the proposed approach is robust and effective in terms of the mean percentage of relative error.

Moreover, this investigation allows us to recommend the use of (1+1) evolution strategies (based on both normally and Cauchy distributed mutations) and particle swarm optimisation algorithms over other methods when using the proposed framework. Indeed, amongst the algorithm under investigation, these two classes guaranteed the best balance of both effectiveness and robustness for the problem at hand.

We believe that this is a promising starting point for further research, and we envisage investigating different regression functions, possibly exploiting interpretable machine learning models, in the future. Also, potential ways forward are in the direction of analysing the fitness landscape induced by the regression function and extending the approach to other related problems in the hydraulic field.

Acknowledgement

This work was partially supported by the following research grants from "Università per Stranieri di Perugia": (i) "Finanziamento Dipartimentale alla Ricerca FDR 2022", (ii) "Artificial Intelligence for Education, Social and Human Sciences", (iii) "Progettazione e sviluppo di strumenti digitali per la formazione a distanza".

References

1. Agresta, A., Baioletti, M., Biscarini, C., Caraffini, F., Milani, A., Santucci, V.: Using optimisation meta-heuristics for the roughness estimation problem in river flow analysis. Applied Sciences **11**(22), 10575 (2021)
2. Agresta, A., Baioletti, M., Biscarini, C., Milani, A., Santucci, V.: Evolutionary algorithms for roughness coefficient estimation in river flow analyses. In: Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24. pp. 795–811. Springer (2021)
3. Andersson, J., Oliveira, D.R., Yeginbayeva, I., Leer-Andersen, M., Bensow, R.E.: Review and comparison of methods to model ship hull roughness. Applied Ocean Research **99**, 102119 (2020)
4. Bélanger, J.: Essay on the Numerical Solution of Some Problems relative to Steady Flow of Water. Carilian-Goeury: Paris, France (1828)
5. Carollo, F., Ferro, V.: Contributo allo studio della lunghezza del risalto libero su fondo liscio e scabro. Rivista di Ingegneria Agraria **35**(4), 13–20 (2004), (in Italian)
6. Carollo, F., Ferro, V.: Determinazione delle altezze coniugate del risalto libero su fondo liscio e scabro. Rivista di Ingegneria Agraria **35**(4), 1–1 (2004)
7. Carollo, F.G., Ferro, V., Pampalone, V.: Hydraulic jumps on rough beds. Journal of Hydraulic Engineering **133**(9), 989–999 (2007)
8. Cauwet, M.L., Couprie, C., Dehos, J., Luc, P., Rapin, J., Riviere, M., Teytaud, F., Teytaud, O., Usunier, N.: Fully parallel hyperparameter search: Reshaped space-filling. In: Int. Conf. on Machine Learning. pp. 1338–1348. PMLR (2020)
9. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Comp. **1**(1), 3 – 18 (2011)
10. Di Francesco, S., Biscarini, C., Manciola, P.: Characterization of a flood event through a sediment analysis: The tescio river case study. Water **8**(7), 308 (2016)

11. Ead, S.A., Rajaratnam, N.: Hydraulic jumps on corrugated beds. Journal of Hydraulic Engineering **128**(7), 656–663 (2002)
12. Gao, F., Han, L.: Implementing the nelder-mead simplex algorithm with adaptive parameters. Computational Optimization and Applications **51**(1), 259–277 (2012)
13. Gul, E., Dursun, O.F., Mohammadian, A.: Experimental study and modeling of hydraulic jump for a suddenly expanding stilling basin using different hybrid algorithms. Water Supply **21**(7), 3752–3771 (05 2021)
14. Hager, W.H., Bremen, R., Kawagoshi, N.: Classical hydraulic jump: length of roller. Journal of Hydraulic Research **28**(5), 591–608 (1990)
15. Hager, W.: Energy Dissipators and Hydraulic Jumps, vol. 8. Kluwer Academic Publication, Dordrecht, The Netherlands (1992)
16. Hager, W.H., Bremen, R.: Classical hydraulic jump: sequent depths. Journal of Hydraulic Research **27**(5), 565–585 (1989)
17. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**(2), 159–195 (2001)
18. Hollander, M., Wolfe, D.A., Chicken, E.: Nonparametric statistical methods, vol. 751. John Wiley & Sons (2013)
19. Hughes, W.C., Flack, J.E.: Hydraulic jump properties over a rough bed. Journal of Hydraulic Engineering **110**(12), 1755–1771 (1984)
20. Karbasi, M.: Estimation of classical hydraulic jump length using teaching–learning based optimization algorithm. J. of Mater. and Envir. Sci. **7**, 2947–2954 (08 2016)
21. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of IEEE International Conference on Neural Networks. vol. 4, pp. 1942–1948 (1995)
22. Kim, J.H.: Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Comp. stat. & data analysis **53**(11), 3735–3745 (2009)
23. Rajaratnam, N.: The hydraulic jump as a well jet. Journal of the Hydraulics Division **91**(5), 107–132 (1965). https://doi.org/10.1061/JYCEAJ.0001299
24. Rao, N.S.G., Ramaprasad: Application of momentum equation in the hydraulic jump. La Houille Blanche **52**(4), 451–453 (1966)
25. Rapin, J., Teytaud, O.: Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad (2018)
26. Rapin, J., Bennet, P., Centeno, E., Haziza, D., Moreau, A., Teytaud, O.: Open source evolutionary structured optimization. In: Proc. of the 2020 Genetic and Evolutionary Computation Conference Companion. pp. 1599–1607 (2020)
27. Safranez, K.: Wechselsprung und die energievernichtung des wassers. Bauingenieur **8**(49), 898–905. (08 1927)
28. Santucci, V., Baioletti, M., Di Bari, G.: An improved memetic algebraic differential evolution for solving the multidimensional two-way number partitioning problem. Expert Systems with Applications **178**, 114938 (2021)
29. Santucci, V., Baioletti, M., Milani, A.: An algebraic differential evolution for the linear ordering problem. In: Proc. of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 1479–1480 (2015)
30. Santucci, V., Milani, A.: Particle swarm optimization in the edas framework. Soft Computing in Industrial Applications **96**, 87–96 (2011)
31. Smetana, J.: Studi sperimentali sul salto di Bidone libero e annegato. Energ. Elettr. **24**(10), 829–835 (1937)
32. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization **11**(4), 341–359 (1997)
33. Yao, X., Liu, Y.: Fast evolution strategies. In: International Conference on Evolutionary Programming. pp. 149–161. Springer (1997)