Welding in the World (2024) 68:805–817
https://doi.org/10.1007/s40194-023-01641-0

RESEARCH PAPER

Reinforcement learning as data-driven optimization technique for GMAW process

Giulio Mattera[1] · **Alessandra Caggiano**[2] · **Luigi Nele**[1]

Received: 25 August 2023 / Accepted: 17 November 2023 / Published online: 21 December 2023

Abstract
Welding optimization is a significant task that contributes to enhancing the final welding quality. However, the selection of an optimal combination of various process parameters poses different challenges. The welding geometry and quality are influenced differently by several process parameters, with some exhibiting opposite effects. Consequently, multiple experiments are typically required to obtain an optimal welding procedure specification (WPS), resulting in the waste of material and costs. To address this challenge, we developed a machine learning model that correlates the process parameters with the final bead geometry, utilizing experimental data. Additionally, we employed a reinforcement learning algorithm, namely stochastic policy optimization (SPO), with the aim to solve different optimization tasks. The first task is a setpoint-based optimization problem that aims to find the process parameters that minimize the amount of deposited material while achieving the desired minimum level of penetration depth. The second task is an optimization problem without setpoint in which the agent aims to maximize the penetration depth and reduce the bead area. The proposed artificial intelligence-based method offers a viable means of reducing the number of experiments necessary to develop a WPS, consequently reducing costs and emissions. Notably, the proposed approach achieves better results with respect to other state-of-art metaheuristic data-driven optimization methods such as genetic algorithm. In particular, the setpoint-based optimization problem is solved in 8 min and with a final mean percentage absolute error (MPAE) of 2.48% with respect to the 42 min and the final 3.42% of the genetic algorithm. The second optimization problem is also solved in less time, 30 s with respect to 6 min of GA, with a higher final reward of 5.8 from the proposed SPO algorithm with respect to the 3.6 obtained from GA.

Keywords Process optimization · Arc welding · Reinforcement learning · Neural networks

1 Introduction

The employment of artificial intelligence has become increasingly widespread in the development of industrial applications, as evidenced by recent research in the field [2, 13],Mattera, Polden, et al., 2023; [19]. Among the various AI technologies available, reinforcement learning techniques are being increasingly applied across a wide range of scientific disciplines [15, 22], but there are still few applications in manufacturing and especially in welding [16, 17]. Reinforcement learning [24], as a sub-field of machine learning, involves the use of a trial-and- error approach to address decision-making problems. Using different learning rules, an agent interacts with an environment with the aim to maximize a reward function using optimization algorithms, such as gradient-based techniques. Although this data-driven technique is mostly employed to solve control problems [20, 23], nowadays researchers proposed the usage of this technique to solve dynamic multi-objective optimization problems [30].

Among all industrial processes, arc welding is one of the most important and, since the bead geometry is a crucial parameter related to process quality, several authors proposed methods to optimize the welding parameters to guarantee the final quality of junctions [5]. As reported in

Recommended for publication by Commission I—Additive Manufacturing, Surfacing and Thermal Cutting

✉ Giulio Mattera
giulio.mattera@unina.it

[1] Department of Chemical, Materials and Industrial Manufacturing Engineering, University of Naples Federico II, 80125 Naples, Italy

[2] Fraunhofer Joint Laboratory of Excellence On Advanced Production Technology (FhJ_LEAPT UniNaples), Piazzale Tecchio 80, 80125 Naples, Italy

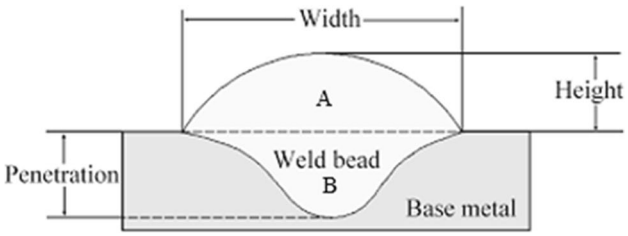

Fig. 1 The bead geometry can be synthesized by four factors, namely the bead width (w), bead height or reinforce (h), penetration depth, and dilution ratio (d)

Fig. 1, the weld geometry can be defined by 4 parameters, namely the bead width, the bead height, the penetration depth, and the dilution factor evaluated, as in Eq. 1, which depend by the area A of the added material and the area B melted by in the base material.

$$d = \frac{B}{A + B} \tag{1}$$

Many process parameters, including welding speed, wire feed speed, nozzle-to-workpiece distance, welding voltage, gas composition and flow rate, and torch angle, influence the bead height, bead width, and penetration depth in different and sometimes opposing ways [8, 28]. To optimize welding parameters, researchers have employed various methods, including genetic algorithms (GA), particle swarm optimization (SPO) and numerical optimization techniques, to overcome the necessity to employ a time-consuming trial and error process, with weld input parameters chosen by the skill of the engineer or machine operator, which lead in increase costs and material waste [7].

Generally, process parameter optimization of arc welding enabled by advanced computing technique can be divided in two macro-area, as reported in Fig. 2, namely data-driven techniques and physical-based techniques. Physical-based techniques consist into usage of finite element (FE) analysis with the aim to conduct virtual experiments, as proposed in [3], but also if physical-based models may be very accurate and give the possibility to estimate variables difficult to measure, such as the heat altered zone geometry or residual stresses and distortions, they require significant time in evaluation and several resources to develop an accurate model. For that reason, data-driven techniques are more used for industrial environments, especially to solve easier tasks. These techniques consist in two different steps, first a model of the system has to be developed using experimental data, and then an optimization technique can be employed to explore the process parameters space and evaluate performances through a fit or reward function.

Giridharan et al. [10] developed a linear model to relate process parameters with the penetration depth, bead width, and bead area for a pulsed GTAW process and used numerical optimization to minimize a non-linear function subjected to constraints with the aim to reduce energy of the process and maintaining the weld geometry parameters in an acceptable range. Dhas et al. [6] proposed a linear model for a SAW process and compared the performances of particle swarm optimization (PSO) and genetic algorithm (GA) in minimization of bead width under constraints on process parameters obtaining similar results. With the aim to improve performances in modeling, Katherasan et al. [12] used a feedforward neural network to model a FCAW process and employed a PSO algorithm to minimize the bead area (A) and maximize the penetration depth. Since the welding process is stochastic, Lee et al. [14] proposed the usage of a Gaussian process regression to model a MAG welding process and used the sequential quadratic programming to maximize the penetration depth and minimize the bead convexity. Finally, Valle Tomaz et al. [26] proposed a neural network to model a GTAW process and employed a GA with the aim to minimize the bead area and maximize both penetration depth and dilution.

Although data-driven models are easier to obtain and data-driven optimization techniques such as GA and PSO can be employed to solve non-linear constrained problems with low computational effort, the main limit of data-driven techniques which employees that data-driven models are related to the limit of their applicability to experimental ranges. In fact, once a model is generated from collected data, the optimization technique can be used only in defined experimental conditions. However, given the suitability of

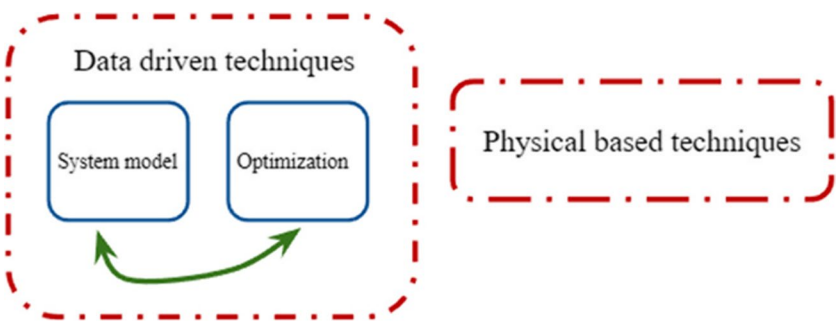

Fig. 2 Arc welding parameters optimization can be divided in data-driven or physical based techniques

data-driven models for industrial environments, this paper proposes a neural network to model the relationship between welding speed, wire feed speed, nozzle-to-workpiece distance, welding voltage, gas flow rate, torch angle, and the bead geometry (width, height, and depth of penetration) for a MIG welding process of mild steel. Furthermore, an innovative reinforcement learning-based framework is proposed to solve the high dimensional non-linear constrained optimization problem in a data-driven manner and the results are compared with other state-of-the-art methods such as GA.

2 Data driven optimization techniques

Data-driven optimization techniques can be employed to find optimal solutions to problems, making decisions based on real world data rather than using theoretical models, which most of the time are based on some assumptions. These techniques can be classified into two main categories: numerical [4] and metaheuristic [29] optimization techniques.

Numerical optimization techniques are focused on finding the optimal solution to a given problem through a systematic exploration of the solution space. These techniques use mathematical derivatives and gradients of the objective function and constraints, making them well-suited for smooth, continuous and well-behaved optimization problems. Typical example are gradient based methods and sequential quadratic programming (SQP) or gradient-based methods (e.g., quasi-Newton methods). Metaheuristic optimization techniques, in other hand, are higher-level strategies that operate at a more abstract level than the specific problem they are trying to solve, making them well-suited for discrete combinatorial optimization problems. They are designed to explore the solution space efficiently, often without relying on gradient information. Metaheuristics are stochastic in nature and use randomness to escape local optima and to explore a wide range of solutions. Typical example is the genetic algorithm and particle swarm optimization.

Numerical optimization methods are typically efficient for continuous problems with well-defined mathematical formulations, but they may struggle with highly non-linear, non-smooth, or combinatorial optimization problems. For these reasons, for these kinds of problems, metaheuristic optimization techniques may be employed, since they excel in solving complex, high-dimensional, non-smooth, and combinatorial optimization problems. They are particularly useful when the problem lacks a clear mathematical formulation or when derivatives are difficult or expensive to compute.

In a welding optimization problem, due to the stochasticity of the process and the usual usage of constrained multi-objective non-linear functions to optimize, metaheuristic optimization techniques are more suitable, also if once the action space is continuous, it can be difficult to explore in an efficient way all the possible actions. To solve that problem, in this paper, we are proposing the usage of a new data-driven optimization technique based on reinforcement learning. Similar to metaheuristic techniques, reinforcement learning can be employed to solve non-linear, constrained and with a non-clear mathematical formulation problem, using a trial-and-error approach based on actions exploration, but at the same time is well suited for problems with both continuous and discrete actions.

2.1 Genetic algorithm

The genetic algorithm (GA) is a powerful metaheuristic decision-making algorithm inspired by Darwin's evolutionary theory and genetics. The algorithm starts by randomly initializing a population of N potential solutions, each represented as a chromosome with genes representing different actions. The population's initialization in the first generation is important since it is the first way for GA to explore the action space. Each chromosome in the population is associated with a score derived from a fitness function, which evaluates the quality of the solution. During the selection process, individuals with better fitness scores, indicating higher objective function values or lower cost function values, have a higher chance of survival and being chosen for the next generation. This selection bias allows the algorithm to concentrate on promising regions of the solution space. After M samples survive to form the new generation, crossover and mutation operators come into play to explore the action space and introduce diversity. Crossover combines pairs of parent individuals to create new offspring with a mix of their genes, simulating genetic recombination. Mutation introduces small random changes in the offspring, promoting exploration of the solution space.

Through repeated iterations of selection, crossover, and mutation, the GA evolves the population, gradually improving the quality of solutions. The algorithm, reported in Fig. 3, aims to find optimal or near-optimal solutions for the given problem by leveraging the principles of natural selection and genetics.

As described above, this algorithm is well suited for discrete action spaces, since the continuous action space problems need for a high population initialization to be solved and the operations, like the crossover, require specific techniques to handle the continuous nature of the variables (genes).

2.2 Reinforcement learning

Optimization problems aim to find the best configuration from a set of potential solutions. These problems can be categorized as either continuous or discrete, depending on whether the values to be combined are infinite or discrete in nature. In recent times, machine learning techniques,

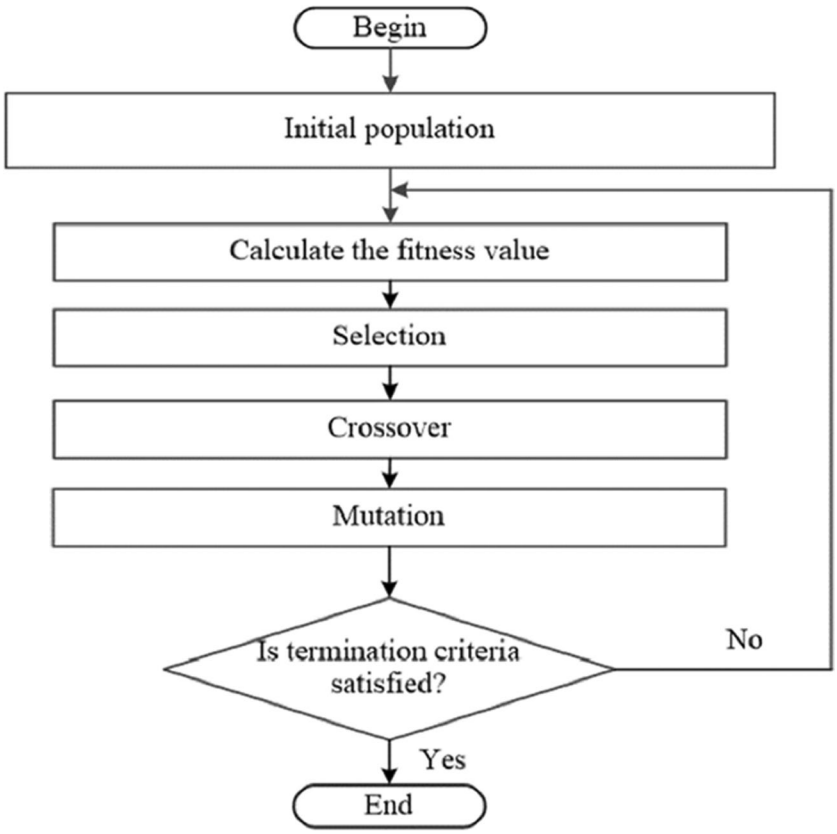

Fig. 3 Genetic algorithm scheme

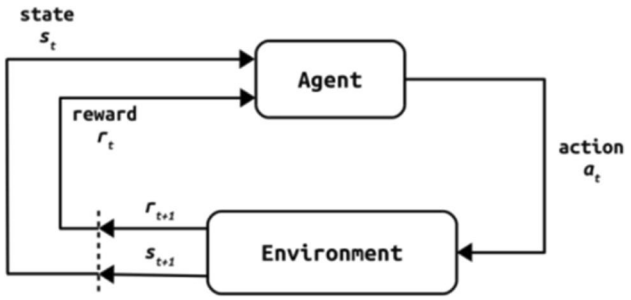

Fig. 4 Reinforcement learning general scheme: an agent receives observations from the environment and act on the environment

particularly reinforcement learning (RL) algorithms, have gained popularity for tackling optimization problems [18]. RL algorithms, in Fig. 4, work by allowing an agent to determine the best sequence of actions that maximize a reward function given a specific state. The agent learns an optimal action policy, denoted as $a = \pi(s)$, through exploration of the environment and receiving feedback in the form of rewards. Model-free RL methods are preferred over model-based ones because they do not require knowledge of the environment's transition functions. Instead, they solely rely on the agent's experiences gained by interacting with the environment, often utilizing Monte Carlo methods.

The main objective of a model-free agent using Monte Carlo methods is to maximize the discounted reward G obtained during N interactions with the system, as represented by Eq. 2, in which R is the reward obtained during the T interaction with the environment and γ is the discounted factor between 0 and 1, which allow to give less importance to the last interactions with the system.

$$G = \sum_{i=0}^{T} \gamma^i R_i(s, a) \tag{2}$$

For k-discrete problems, we evaluate k-discounted rewards associated at each action and the optimal policy π^* is straightforward, since can be evaluated as reported in Eq. 3, in which the action to take is the one associated with higher discounted reward G.

$$\pi^* = argamx[G] \tag{3}$$

Discrete methods become computational unfeasible to solve with the presented methodology when the state and action space tends to increase, so for continuous problems, a differentiable function $f_\theta(s)$ with parameters θ can be used to approximate the optimal policy. Using gradient based methods, such as policy gradient [25], the function parameters can be update with the aim to maximize the final discount

reward. In this work, we proposed the usage of stochastic policy optimization algorithm.

2.3 Stochastic policy optimization

Policy gradient algorithms [25] are the most popular class of continuous action reinforcement learning algorithms, since it is possible use a continuous function with parameters θ to approximate the optimal control policy π_θ, namely the function that correlate the input states with the optimal actions. The learning phase consist into evaluation of the gradient associated to the policy, as in Eq. 4, and update the weights of the policy using a gradient ascent algorithm, where J in the function to maximize or $J = E[R(s,a)]$, in which E is the expected value of the obtained rewards using the action a in the state s. In this work, we proposed a variant of the REINFORCE algorithm [27] in which instead to use the Q value to evaluate the policy gradient we use the reward R.

$$\nabla_\theta J(\pi) = \int \nabla_\theta log\pi_\theta(s,a)R(s,a) \tag{4}$$

If the policy is stochastic, the algorithm is named stochastic policy optimization (SPO), and a general schema is reported in Fig. 5. If we suppose that the stochastic policy is given by a Gaussian distribution, the policy function aims to find the mean vector and covariance matrix of the optimal policy. Once an input is given to the policy, an information related to the optimization problem such as a reference point or an initial state, a random action is sampled from the stochastic policy. A reward can be obtained as result of the application of that action on the environment, as in the case of the fit score of the GA, and a gradient can be computed accordingly with Eq. 4, and finally, the policy parameters can be updated. To approximate the policy, any function can be used, so utilizing neural networks give the possibility to approximate the optimal policy using a complex non-linear function; for that reason, this method is largely used in SOA applications.

As presented, this methodology is not free from drawbacks. First, RL algorithms suffer from samples inefficiency, since they require many interactions with the environment to learn an effective policy. The learning process can be time-consuming, especially in complex environments, and for that reason usually, the agent is trained in simulation. In contrast, GA can converge to solutions more efficiently, particularly in problems with a well-defined fitness landscape. Furthermore, as gradient-based optimization, RL can get stuck in local optima, especially in problems with sparse rewards or non-differentiable environments. In comparison the GA has similar problem but mostly related to population initialization. Finally, especially for static optimization problem, we have to handle with policy representation. In fact, a RL algorithm aims to map the best state-action relationship, so practically, we need for an input with which impulse the optimal policy to get an output. This means that it really can take advantages from dynamic optimization problem, in which the state of the last interaction can be used to drive the next one decision, but in static problem can be difficult synthetize the policy. However, in this work, we are proposing a way to represent a policy in this scenario.

3 Parameter optimization

This study is developed on experimental data obtained from a two-level design of experiment (DOE) of a GMAW process using a 1.2-mm ER70S-6 wire and a structural mild steel (plates) having composition—C—0.10, Mn—0.9, Si—0.04, S—0.032, P—0.032 and dimensions of 150×75×8 mm as base material under argon protection. During the experimental campaign, the input parameters were changed according to Table 1 for a total of 64 experiments [8]. The process parameters to optimize are the wire feed speed (WFS), the welding voltage (V), the welding speed (WS), the

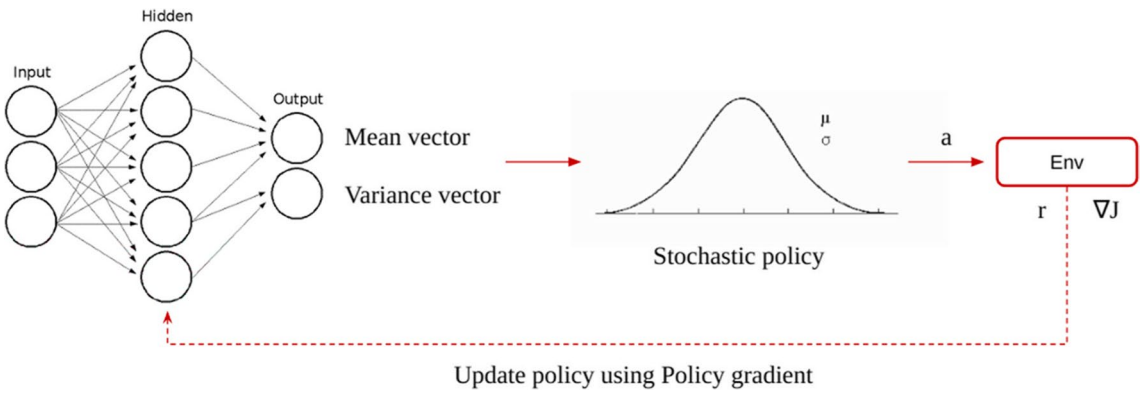

Fig. 5 Stochastic policy optimization algorithm. The action is sampled from a stochastic policy, e.g., a Gaussian distribution

Table 1 Experimental campaign (DOE) [8]

Parameter	*Level 1*	*Level 2*	*Unit*
Welding speed	25	45	cm/min
Welding voltage	26	30	V
Wire feed speed	6	7	m/min
Gas flow rate	14	18	l/min
Contact to workpiece distance	15	20	mm
Torch angle	70	100	degree

contact-to-workpiece-distance (CTWD), the travel angle (TA), and the gas flow rate (GFR), for a total of 6 process parameters. The aim of the optimization is to minimize the bead area reaching the desired level of penetration depth. To reach the scope, two different data-driven optimization algorithms are used, namely the genetic algorithm and the stochastic policy optimization. A model of the system is obtained via regression analysis, and the optimization problem is treated as a continuous action space problem, since the generalization capability of the regression analysis gives the possibility to explore the effect of different parameters combination.

3.1 Regression analysis for metal inert gas static modeling

Based on collected data, a static model of the welding system was developed using linear regression analysis and a shallow neural network with the aim to correlate the input process parameters with the average steady state bead geometry. The most prevalent type of regression is the linear regression, where the connection between the predictor variables x and the response variable y is presumed to be linear, as shown in Eq. 5.

$$y = Wx \tag{5}$$

As regression constitutes a supervised learning technique, the coefficients W, which multiply the input features x, can be approximated with the aid of the desired result's information. For this purpose, estimators like ordinary least squares (OLS), Lp estimator [9], or gradient descent optimization [1] may be employed. The OLS estimator is commonly used, particularly when information about the error distribution is not available, as it adheres to the Gauss-Markov theorem, making it the best linear unbiased estimator (BLUE), and the weights can be evaluated using Eq. 6.

$$W = \left(XX^T\right)^{-1}Xy \tag{6}$$

Although it is possible use a linear regression to develop a static model for a GMAW system, it has not a huge performance in finding a complex relationship between data. For this reason, as presented also in the introduction, several authors used neural networks as data-driven modeling techniques, reaching good performances. A neural network is a computational graph inspired by biology in which the core element is the artificial neuron, a function f of the inputs $x = (x_1 \ldots x_n)$ weighted by a vector of connection weights $w = (w_{j,1} \ldots w_{j,n})$ and passed through a non-linear function Φ, namely the activation function. The mathematical expression is reported in Eq. 7, while in Fig. 6 is shown the how work the neuron graph.

$$y_j = \Phi(w_{jn}x_n) \tag{7}$$

The activation function Φ and the initialization algorithm of the weights at the beginning of training are crucial ingredients for neural networks [21], and several functions and initialization algorithms [11] might be used. The goal of the learning process is to find the weights w that minimize a cost, namely the loss function.

Before developing a model, an exploratory analysis was performed using the Spearman correlation index, and the findings were presented in Table 2. The results reveal significant correlations between welding voltage and various bead geometry aspects. Specifically, welding voltage shows a positive correlation with width and penetration depth, while having a negative correlation with bead height. Similarly, the wire feed speed demonstrates positive correlations with all bead geometry states, as an increase in this speed leads to

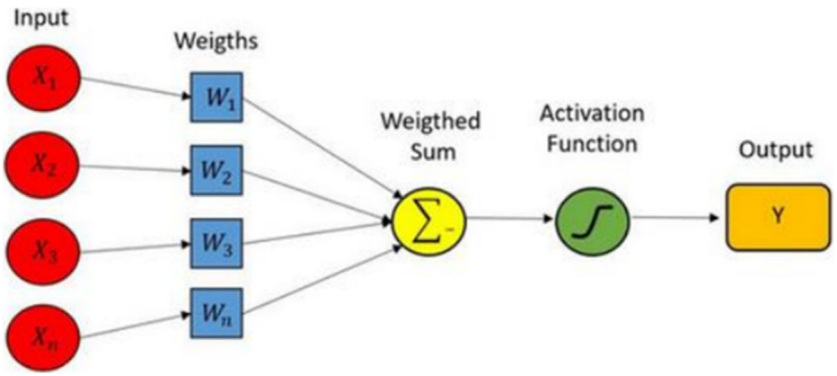

Fig. 6 A neuron is a computational agent that gives in output a linear combination of the inputs passed in a non-linear activation function

Table 2 Results of correlation analysis using Spearman index (SI)

Variable name		SI
Penetration depth (p)	Wire feed speed	0.42
	Welding speed	0.27
	Welding voltage	−0.32
	Contact to workpiece distance	−0.015
	Gas flow rate	−0.002
	Torch angle	−0.25
	Bead area	0.51
Bead width (w)	Wire feed speed	0.55
	Welding speed	0.22
	Welding voltage	−0.44
	Contact to workpiece distance	−0.28
	Gas flow rate	−0.08
	Torch angle	−0.033
Bead height (h)	Wire feed speed	−0.53
	Welding speed	0.39
	Welding voltage	−0.76
	Contact to workpiece distance	−0.29
	Gas flow rate	0.06
	Torch angle	−0.12

higher deposition rates and arc currents. In contrast, welding speed shows negative correlations with all bead geometry states, as it results in lower heat input. Furthermore, the nozzle-to-workpiece distance also exhibits negative correlations with all bead geometry states, with increased torch height leading to higher arc resistance, reduced arc current, and lower heat input. While the gas flow rate does not show strong correlations with bead geometry, the torch angle appears to have some degree of correlation with penetration depth and bead height. Moreover, the data indicates that penetration depth is significantly influenced by the bead area, which is defined as the product of bead height and width, in particular a positive correlation was found.

As shown by the correlation analysis conducted, some parameters, like wire feed speed, have different correlations with respect to different weld geometry variables. Furthermore, also if same correlation can be found between different variables, the entity of that correlation can be different, as for the welding voltage. This confirm that finding the set of parameters that allow to have the desired bead geometry, related to the final quality of the joint, is not an easy task and the usage of optimization techniques is required. As comparison analysis, in Table 3 are reported the mean squared error (MSE) and the mean absolute error (MAE) obtained employing for modeling a linear regression with an OLS estimator used for weights identification and a neural network. As expected, better results are obtained with a neural network, with a MSE and MAE, respectively, of 1.26 and 0.68, when the network is a simple single layer perceptron (SLP) with 1 hidden layer with 16 neurons and it is trained using a root mean squared propagation optimization algorithm with a learning rate of 0.0001 and a discount factor of 0.9 for 1000 epochs.

Table 3 Metrics to compare the performance in prediction of a neural network and a linear regression using OLS estimator

Model	*MAE*	*MSE*
OLS linear regression	1	2.54
Shallow neural network	0.68	1.26

3.2 Optimizers setup

In this work, two different optimization algorithms are proposed, namely the genetic algorithm and a new method based on reinforcement learning named stochastic policy optimization (SPO). The aim of the optimization of this work is to find the optimal process parameters to use to reach different penetration depth references, namely 1.6, 1.7, 1.8, 1.9, 2, 2.1, and 2.2 mm, which at the same time minimize the final bead area. To solve the policy representation problem of the SPO algorithm, the proposed framework is shown in Fig. 7. The penetration depth reference is the input to neural network, and the agent, interacting with the environment, updates each time his knowledge about how to solve the problem, reach the reference, and minimize the bead area, using the policy gradient theorem.

4 Results

For the GA optimization framework, 7 different optimization problems are solved using an initial population of 50 chromosomes randomly initialized. Each chromosome contains 6 genes, namely the six process parameters to use, that are randomly initialized sampling from an multivariate uniform distribution in which the minimum and maximum values are − 1 and + 1, which during the selection phase, in which a fit value is computed interacting with the GMAW model, are scaled accordingly with minimum and maximum values of the proposed 2 level DOE. This encoding technique allows to use a single point crossover method, as in the standard GA algorithm. The probability of mutation and crossover is set to 0.5, and a Gaussian white noise with 0.2 standard deviation (mutation gain) is used during the mutation to explore the continuous action space. The algorithm is repeated for 10 generation.

For the SPO optimization framework, to try a best comparison of the results, a neural network with 1 hidden layer 50 neurons and ReLu activation function is used. The action space, as in the case of GA, needed to be bounded with

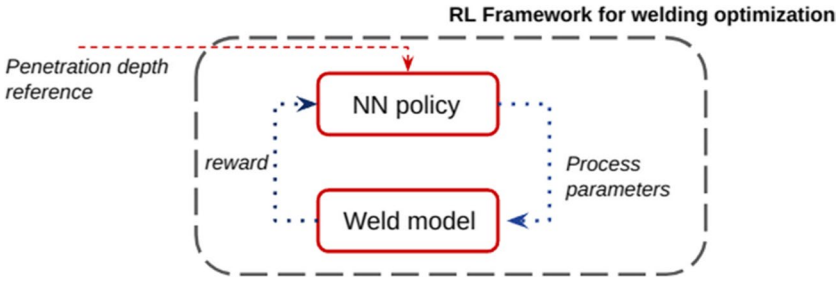

Fig. 7 The proposed RL framework for welding optimization

respect to a lower and an upper bound of the DOE, so a hyperbolic tangent is used as the activation function of the output layer, which refers to the mean vector of the multivariate Gaussian distribution (MVGD) policy. Furthermore, a scale layer is added to convert the output of tanh function from the interval [− 1, 1] into the action space interval to evaluate the reward function. The covariance matrix of the MVGD Σ is fixed to be a diagonal matrix as reported in Eq. 8, so no correlation noise is used to approximate the policy. During the training phase, the initial standard deviation utilize is 0.2, and it is linear reduced until reach the value of 0.03 at the end of the training phase, encouraging the agent to use the mean value as action at the end of training.

$$\Sigma = \sigma_{\text{ini}} \cdot I \tag{8}$$

The input layer consists of 1 unit, namely the penetration depth reference. In this way, the goal of the agent is to find the optimal combinations of process parameters to use knowing the reference point. The reward function used is the same for both algorithms and is computed using a setpoint − based approach. In particular, a modified version of the Spielberg method is used and reported in Eq. 9.

$$R(s,a) = \begin{cases} c - \beta(w \cdot h) & if \ \sum_i p_r^i - p^i < \epsilon \\ -(|\gamma \sum_i p_r^i - p^i|) - 4\beta(w \cdot h) & otherwise \end{cases} \tag{9}$$

When the model output is closer to the setpoint by the error tolerance defined by ε, the agent receives the highest possible reward "c" with a fixed penalty which depends by the bead area value scaled by β. If the goal it is not reached, the agent receives the negative reward whose magnitude is the deviation from the setpoint scaled by a factor γ, and it is proportional to bead area. For the learning phase, adaptive momentum optimizer was used for the network policy in the SPO algorithm. Constructing the reward function in this manner, the goal of the agent is to find the best set of process parameters that allow obtaining the desired level of penetration and that at the same time reduced the bead area for different input of penetration depth reference. The learning setup for both algorithms is show in Table 4.

To solve the same optimization problem, the genetic algorithm need of 42 min using an AMD Ryzen 7 4800H CPU, while the SPO based on neural network learning, can be benefit of easy GPU parallelization offered by NVIDIA and TensorFlow which allow to solve the same task in 8 min with a NVIDIA GTX 1650 Ti 4GB. The general results, in terms of predicted final penetration depth and bead, are synthesized in Table 5. The results have shown that SPO algorithm outperforms the GA in all references except for the reference of 2.2 mm in terms of final reward. Furthermore, the final penetration error obtained is lower using the SPO algorithm, thanks to better performance in exploration in solving continuous action space problems. In particular, the mean percentage absolute error (MPAE) is reduced from 3.42 to 2.48%. The importance of exploration can be observed also looking to the final variance of the solution offered from SPO, which led to obtain better results. Finally, the agent learning process is summarized in Fig. 8. At the beginning, due to exploration, the agent takes random actions to explore the results in combining different possibilities. This random exploration is repeated 100 times to complete an episode, and then, the

Table 4 Hyperparameters used to train the SPO agent and the 7 parallel GAs

Model	*Hyperparameters*	*Value*
Genetic algorithm	Initial population	50
	Crossover prob	0.5
	Mutation prob	0.5
	Mutation gain	0.2
	Num. of generations	10
Stochastic policy gradient	Learning rate	0.001
	Interactions	200
	Initial exploration	0.2
Reward function hyperparameters	Max score (c)	10
	Error scale (γ)	5
	Bead area scale (β)	0.05
	Error threshold (ϵ)	0.1

Table 5 Comparison between obtained results

Model	*Parameters*	*Parameter values [V, WFS, WS, CTWD, GFR, TA]*	*Final p*	*Final area*	*Reward*	*Error %*
Genetic algorithm	1.6	26, 6, 450, 20, 14, 80	1.53	23	8.85	4.40
	1.7	29.1, 7, 346, 18, 16, 90	1.79	29.6	8.52	5.23
	1.8	28.8, 6.9, 340, 18, 15, 90	1.83	31	8.45	1.66
	1.9	30, 7, 350, 18, 15, 90	1.85	29.3	8.54	2.63
	2	29, 7, 450, 17, 16, 77	1.95	24.8	8.76	2.5
	2.1	29.5, 7, 400, 17, 17, 80	2.08	26	8.70	0.95
	2.2	29.5, 7, 385, 18, 15, 75	2.27	26.5	8.67	3.18
Stochastic policy optimization	1.6	26, 6, 450, 15, 16, 100	1.56	20.2	8.99	2.5
	1.7	26, 6.1, 450, 18, 16, 90	1.66	22.7	8.87	2.35
	1.8	26.3, 6.8, 447, 19, 15, 90	1.84	24	8.80	2.2
	1.9	27.5, 6.5, 427, 17, 15, 77	1.95	24.4	8.78	2.63
	2	28.5, 6.3, 423, 17, 16, 77	1.96	24.2	8.78	2
	2.1	28.6, 6.5, 400, 18, 16, 77	2.07	24.3	8.78	1.43
	2.2	26.1, 6.1, 352, 15, 16, 100	2.16	28.5	8.58	1.8

agent knowledge is updated with respect to the behavior of current policy. During the learning procedure, the error, that is the most important factor at the beginning, is driven under the defined threshold, and then the agent aims to reduce the bead area without exceeds the error bounds.

4.1 Optimization problems without setpoints

Although the obtained results look promising, since they give the possibility to outperform the state-of-the-optimization methods in less amount of time, in some optimization problems there are not references to follow, as proposed by the several example in literature. So, an additional example of optimization is proposed in this section in which the aim of the agent is to maximize the penetration depth and reduce the bead area. In this case, the proposed optimization framework can be represented as in Fig. 9, in which the input of the agent is the states of the environment, and at each iteration, the aim is to maximize the obtained penetration depth and minimize the bead area respect to last interaction.

In this case, the proposed reward function, used in the GA and for the training of SPO, is reported in Eq. 10. Once a reward function is developed, especially for gradient based optimization algorithms, it is important that it has a gradient that the agent can use. A continuous reward inspired by Gaussian function is proposed for that scope.

$$\mathrm{R(s,a)} = \mathrm{e}^{-\left(\frac{BA}{\beta}\right)^2} + -\eta \cdot p^2 \tag{10}$$

Once the bead area is in the interval described to $[-\beta, +\beta]$, the Gaussian part of the reward function give a high result, while a higher value is returned by a high value of penetration depth scaled by a factor η. The reward constructed in this manner is derivable, and selecting the hyperparameters, it is a possible constrain in the optimization problem. The results of GA and SPO with the hyperparameter used are reported in Table 6 and in Fig. 10.

Also in this second optimization task, the SPO algorithm outperforms the GA with a final reward of 5.8 with respect to 3.6 of GA. The computational time of SPO is 30 s with respect of 6 min of GA, and the final obtained penetration depth is 6% higher in the case of SPO.

5 Conclusions

In this study, a novel reinforcement learning method is proposed for optimizing a gas metal arc welding process, and the results are compared with other state-of-the-art data-driven optimizations as genetic algorithm. The first task proposed in this work is to find an optimal policy that selects the appropriate combination of welding parameters, including welding speed (WS), wire feed speed (WFS), contact-to-workpiece distance (CTWD), welding voltage (V), gas flow rate (GFR), and torch angle (TA), to achieve the desired penetration depth (p) and at the same time minimizing the bead area, so reducing material costs. The simulation results demonstrate that SPO agent can achieve the target with a mean percentage absolute error (MPAE) error of 2.48% with respect to the 3.42% of the genetic algorithm, indicating the high potential of the proposed approach especially thanks the reduced optimization time that is reduced from 42 to 8 min.

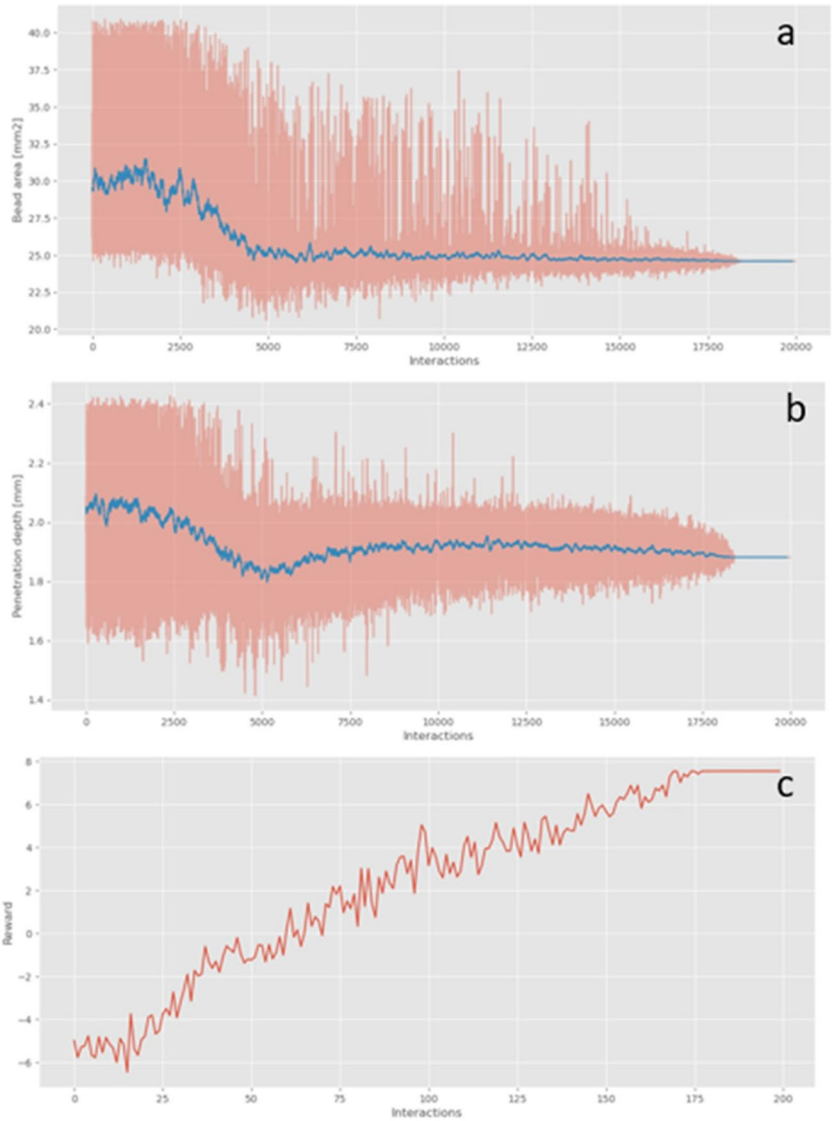

Fig. 8 The bead area (**a**), penetration depth (**b**), and reward function (**c**) trend during the optimization for the case $r = 1.9$ mm

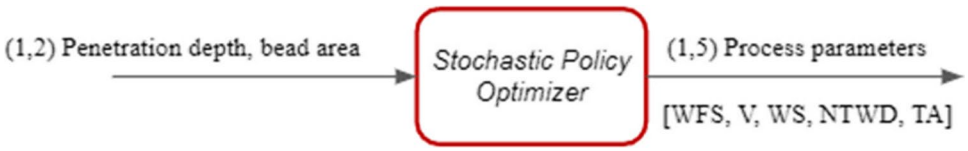

Fig. 9 Modified framework for static optimization without setpoints

To show the performance in solving task not based on setpoint, a new reward function is proposed with the aim to maximize the penetration depth and reduce the final bead area.

After 30 s of optimization, the SPO algorithm outperform the GA with a final reward of 5.8 with respect to 3.6 of GA that required 6 min to solve the same optimization problem. This work confirms the superior performance achieved by our innovative approach, which harnesses the advantages of both gradient-based numerical optimization and the simplicity in problem formulation, including goal and constraint definition and characteristic of metaheuristic methods like genetic algorithms.

Table 6 Comparison between obtained results on optimization problem without setpoint

Model	*Parameters*	*Values*	*Final p*	*Final area*	*Reward*
Genetic algorithm	Wire feed speed	6.1	2.11	24.1	3.6
	Welding speed	443			
	Welding voltage	28.8			
	Contact to workpiece distance	19			
	Gas flow rate	14			
	Torch angle	70			
Stochastic policy optimization	Wire feed speed	6.5	2.25	26.7	5.8
	Welding speed	28			
	Welding voltage	358			
	Contact to workpiece distance	18			
	Gas flow rate	16			
	Torch angle	85			

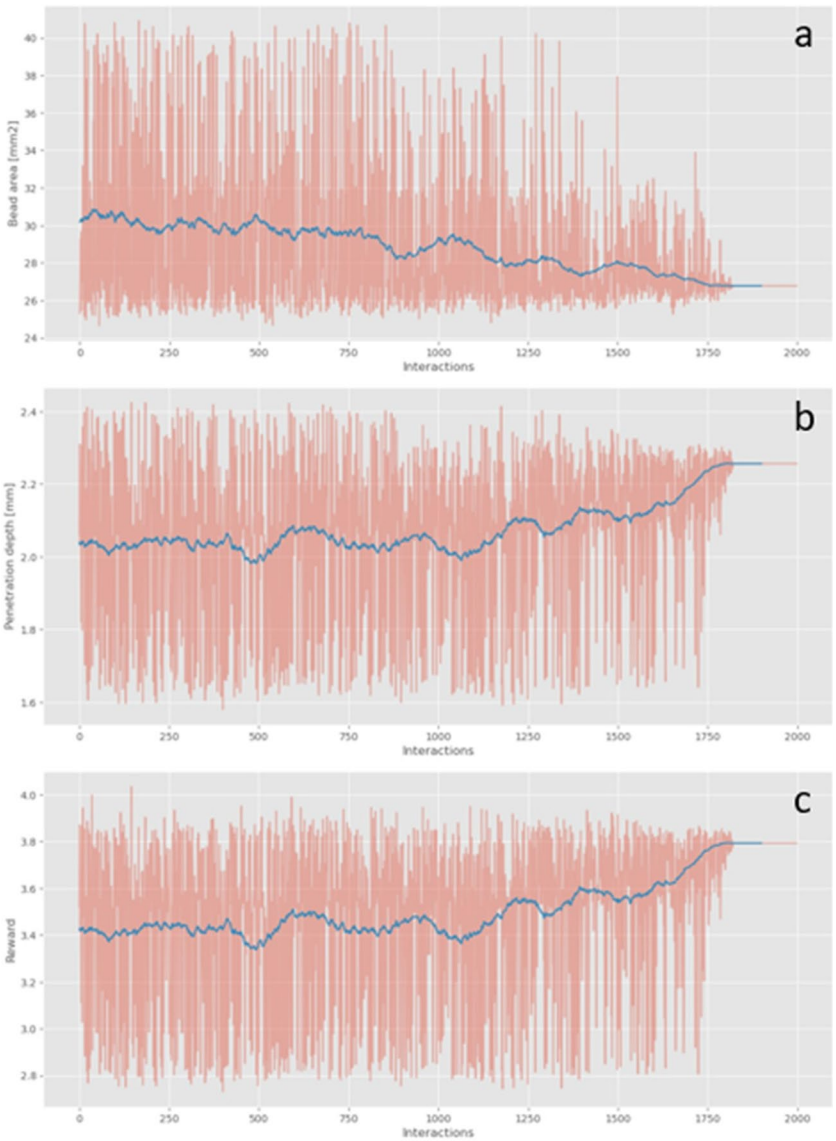

Fig. 10 The bead area (**a**), penetration depth (**b**), and reward function (**c**) trend during the optimization without setpoints

Author contribution Data curation: G.M. Formal analysis: G.M. and A.C. Investigation: G.M. Methodology: G.M. Software: G.M. Supervision: L.N. All authors have read and agreed to the published version of the manuscript.

Funding Open access funding provided by Università degli Studi di Napoli Federico II within the CRUI-CARE Agreement.

Data availability Data are available under request to corresponding author.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Amari S (1993) Backpropagation and stochastic gradient descent method. Neurocomputing 5(4–5):185–196
2. Caggiano A, Mattera G, Nele L (2023) Smart tool wear monitoring of CFRP/CFRP stack drilling using autoencoders and memory-based neural networks. Appl Sci (Switzerland) 13(5):3307. https://doi.org/10.3390/app13053307
3. Chen FF, Xiang J, Thomas DG, Murphy AB (2020) Model-based parameter optimization for arc welding process simulation. Appl Math Model 81:386–400. https://doi.org/10.1016/j.apm.2019.12.014
4. Conway BA (2012) A survey of methods available for the numerical optimization of continuous dynamic systems. J Optim Theory Appl 152(2):271–306. https://doi.org/10.1007/s10957-011-9918-z
5. Das D, Jaypuria S, Pratihar DK, Roy GG (2021) Weld optimisation. Sci Technol Weld Joining 26(3):181–195. https://doi.org/10.1080/13621718.2021.1872856
6. Dhas JER, Kumanan S (2011) Optimization of parameters of submerged arc weld using non conventional techniques. Appl Soft Comput 11(8):5198–5204. https://doi.org/10.1016/j.asoc.2011.05.041
7. Gadakh VS, Shinde VB, Khemnar NS (2013) Optimization of welding process parameters using MOORA method. Int J Adv Manuf Technol 69(9–12):2031–2039. https://doi.org/10.1007/s00170-013-5188-2
8. Ganjigatti JP, Pratihar DK, RoyChoudhury A (2008) Modeling of the MIG welding process using statistical approaches. Int J Adv Manuf Technol 35(11–12):1166–1190. https://doi.org/10.1007/s00170-006-0798-6
9. Giacalone M, Panarello D, Mattera R (2018) Multicollinearity in regression: an efficiency comparison between Lp-norm and least squares estimators. Qual Quant 52(4):1831–1859. https://doi.org/10.1007/s11135-017-0571-y
10. Giridharan PK, Murugan N (2009) Optimization of pulsed GTA welding process parameters for the welding of AISI 304L stainless steel sheets. Int J Adv Manuf Technol 40(5–6):478–489. https://doi.org/10.1007/s00170-008-1373-0
11. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proc Mach Learn Res 9:249–256
12. Katherasan D, Elias JV, Sathiya P, Haq AN (2014) Simulation and parameter optimization of flux cored arc welding using artificial neural network and particle swarm optimization algorithm. J Intell Manuf 25(1):67–76. https://doi.org/10.1007/s10845-012-0675-0
13. Krishnaveni S, Kunchala BR, Gamini S, Ch Anilkumar T (2023) Machine learning-based bead modeling of wire arc additive manufacturing (WAAM) using an industrial robot. Materal Today: Proceedings. https://doi.org/10.1016/j.matpr.2023.04.534
14. Lee DY, Leifsson L, Kim J-Y, Lee SH (2020) Optimisation of hybrid tandem metal active gas welding using Gaussian process regression. Sci Technol Weld Joining 25(3):208–217. https://doi.org/10.1080/13621718.2019.1666222
15. Mattera G, Mattera R (2023) Shrinkage estimation with reinforcement learning of large variance matrices for portfolio selection. Intell Syst Appl. https://doi.org/10.1016/j.iswa.2023.200181
16. Mattera G, Nele L, Paolella D (2023) Monitoring and control the wire arc additive manufacturing process using artificial intelligence techniques: a review. J Intell Manuf. https://doi.org/10.1007/s10845-023-02085-5
17. Mattera G, Polden J, Caggiano A, Commins P, Nele L, Pan Z (2023) Anomaly detection of wire arc additively manufactured parts via surface tension transfer through unsupervised machine learning techniques. 17th CIRP Conference on Intell Comput Manuf Eng
18. Mazyavkina N, Sviridov S, Ivanov S, Burnaev E (2021) Reinforcement learning for combinatorial optimization: a survey. Comput Oper Res 134:105400. https://doi.org/10.1016/j.cor.2021.105400
19. Nele L, Mattera G, Vozza M (2022) Deep neural networks for defects detection in gas metal arc welding. Appl Sci (Switzerland) 12(7):3615. https://doi.org/10.3390/app12073615
20. Nian R, Liu J, Huang B (2020) A review On reinforcement learning: introduction and applications in industrial process control. Comput Chem Eng 139:106886. https://doi.org/10.1016/j.compchemeng.2020.106886
21. Pedamonti D (2018) Comparison of non-linear activation functions for deep neural networks on MNIST classification task. http://arxiv.org/abs/1804.02763
22. Polydoros AS, Nalpantidis L (2017) Survey of model-based reinforcement learning: applications on robotics. J Intell Rob Syst 86(2):153–173
23. Spielberg SPK, Gopaluni RB, Loewen PD (2017) Deep reinforcement learning approaches for process control. 2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP), 201–206. https://doi.org/10.1109/ADCONIP.2017.7983780
24. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press. http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html
25. Sutton RS, Mcallester D, Singh S, Mansour Y (1999) Policy gradient methods for reinforcement learning with function approximation. Adv Neural Info Process Syst 12
26. Tomaz do IV, Colaço FHG, Sarfraz S, Pimenov DYu, Gupta MK, Pintaude G (2021) Investigations on quality characteristics in gas tungsten arc welding process using artificial neural network integrated with genetic algorithm. Int J Adv Manuf Technol 113(11–12):3569–3583. https://doi.org/10.1007/s00170-021-06846-5
27. Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8(3):229–256
28. Xiong J, Zhang G, Hu J, Wu L (2014) Bead geometry prediction for robotic GMAW-based rapid manufacturing through a neural network and a second-order regression analysis. J Intell Manuf 25(1):157–163
29. Yang XS (2011) Metaheuristic optimization: algorithm analysis and open problems. In: Pardalos PM, Rebennack S (eds) Experimental Algorithms. SEA 2011. Lecture Notes in Computer Science, vol 6630. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-20662-7_2

30. Zou F, Yen GG, Tang L, Wang C (2021) A reinforcement learning approach for dynamic multi-objective optimization. Inf Sci 546:815–834. https://doi.org/10.1016/j.ins.2020.08.101

The manuscript has not been submitted to another journal and will not be published elsewhere within 1 year.