Exploiting Artificial Swarms for the Virtual Measurement of Backlash in Industrial Robots

Exploiting Artificial Swarms for the Virtual Measurement of Backlash in Industrial Robots

Eliana Giovannitti
Comau S.p.A.
Torino, Italy
0000-0001-5180-6651

Sayyidshahab Nabavi
Islamic Azad university of Urmia
Urmia, Iran
0000-0003-4772-9651

Giovanni Squillero
Politecnico di Torino
Torino, Italy
0000-0001-5784-6435

Alberto Tonda
UMR 518 MIA, INRAE
Paris, France
0000-0001-5895-4809

***Abstract*—The *backlash* is a lost motion in a mechanism created by gaps between its parts. It causes vibrations that increase over time and negatively affect accuracy and performance. The quickest and most precise way to measure the backlash is to use specific sensors, that have to be added to the standard equipment of the robot. However, this solution is little used in practice because raises the manufacturing costs. An alternative solution can be to exploit a *virtual sensor*, i.e., the information about phenomena that are not directly measured is reconstructed by signals from sensors used for other measurements.**

This work evaluates the use of bio-inspired swarm algorithms as the processing core of a virtual sensor for the backlash of a robotic joint. Swarm-based approaches, with their relatively modest occupation of memory and low computational load, could be ideal candidates to solve the problem. In this paper, we exploit four state-of-the-art swarm-based optimization algorithms: the *Dragonfly Algorithm*, the *Ant Lion Optimizer*, the *Grasshopper Optimization Algorithm*, and the *Grey Wolf Optimizer*. The four candidate algorithms are compared on 20 different datasets covering a range of backlash values that reflect an industrial case scenario. Numerical results indicate that, unfortunately, none of the algorithms considered provides satisfactory solutions for the problem analyzed. Therefore, even if promising, these algorithms cannot represent the final choice for the problem of interest.

***Index Terms*—backlash, robotic manipulator, real-world, industry application, swarms-based approaches**

I. Introduction

Industrial robots are used by manufacturing companies to perform repetitive, difficult or even dangerous tasks [1]. Robotics manipulators are made of a sequence of rigid *links*, connected by movable *joints*. These joints are actuated by electrical motors and include a set of gears that are designed to deliver power from the motor to the link as efficiently as possible. In engineering, *Backlash* is the name given to the lost motion in a mechanism caused by gaps between the parts [2]; in industrial manipulators, that is, in the context of gears and gear trains, the backlash is the amount of clearance between mated gear teeth (Figure 2). Backlash is dependent on wear and increases over time [3]. It causes vibrations to arise and eventually impair the robot positioning accuracy degrading the performance. The immediate effect is a reduction in the quality of the pieces produced by the robot, and in the long term the breakage of the joint can occur.

Authors are listed in alphabetical order.

978-1-7281-8393-0/21/$31.00 ©2021 IEEE

Fig. 1: A robotic manipulator (*Comau SMART NJ 650*).

To prevent this scenario industries target maintenance interventions with great care. On the one hand, they try to avoid unnecessary components substitutions and limit the number of stops; on the other hand, they don't want to come up with the breakdown of a joint that may cause as a stop in the whole production line bringing significant economic losses. Therefore, to guarantee the quality of the production and to predict failures, an assessment of the value of the backlash is fundamental. A direct measure of the backlash can be obtained using two coupled encoders, one on the motor side and one on the load side, or using an accelerometer. Unfortunately, these sensors are not part of the standard equipment of industrial robots, and adding them to the robot means higher costs of the final product. Moreover, it can be difficult to find free space inside the robotic joint to place the extra sensors. A solution for these problems can be the use of virtual sensors. A *virtual sensor* is a piece of code which serves as physical sensor. It uses the information available from other onboard measurements, and infer an estimate of the quantity of interest that can not be directly measured. The measure is achieved relying on mathematical models and relationships between different variables.

In [4], authors try to estimate the value of the actual measurement of the joint backlash by using measures from the onboard motor encoder. The measure of the motor position is carefully analyzed looking for information about the backlash

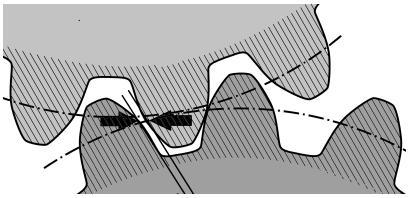

Fig. 2: Backlash in mating gears.

status. In particular, the analog signal is scanned to find the match with a peculiar disturbance pattern related to the backlash presence, see Figure3. This pattern is obtained and characterized through simulation, and is taken as the signature of the backlash in the encoder signal. A generic formula (1) is developed to model it, and an estimate of the underlying backlash is obtained by fitting the formula against the measured data. The measure of the backlash is simply given by the measure of the amplitude of the oscillation, but the fit of the whole formula is required to discern the specific disturbance caused by backlash from any other disturbance source. In more

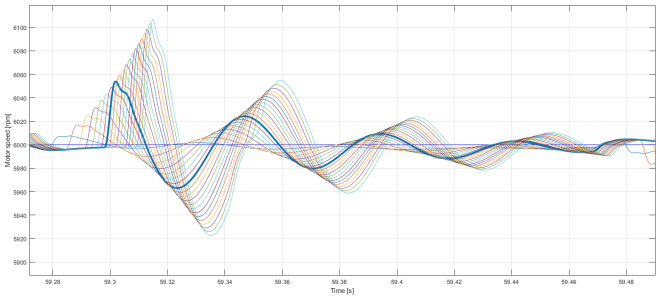

Fig. 3: Backlash disturbance on the reference signal (blue) and how its appearance changes at increasing backlash gap.

details, the formula of the expected disturbance on the encoder signal is

$$d_{\mathrm{b}}(t) = \begin{cases} A\, e^{-(t-t_1)\tau} \sin \omega(t) & t_1 < t < t_2 \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

where A and τ are the parameter related to the amplitude decay; t_1 is the starting time of the oscillation, and t_2 is the ending time, that is, $d_{\mathrm{b}}(t) = 0$ if $t \notin [t_1, t_2]$. In [4] the optimization phase is carried out by an Evolutionary Algorithm that provides accurate and repeatable results. Unfortunately, the algorithm has proved to be time consuming, and demanding in terms of memory occupation and computational load. From this emerged the idea of this paper: try to use easier and lighter to implement optimization algorithms.

In this work, we analyze the use of different swarm-based meta-heuristics for the core of the virtual sensor, that is the fitting of Equation (1) with the real data, and evaluate their performance in terms of precision and accuracy.

The paper is organized as follows: Section II describes the four algorithms analyzed and the bio-inspired behavior models at the basis of their functioning; Section III provides an overview of the backlash estimation problem and how it has been formalized in terms of an optimization problem; Section IV is about the experiments performed and the results discussion. Finally, the last section contains the conclusions on the work carried out.

II. Swarm-Based Approaches

This paper is focused on four state-of-the-art nature-inspired algorithms, all emulating typical behaviours of the animal world. They are: the Dragonfly Algorithm, the AntLion Optimizer, the Grasshopper Optimization Algorithm, and the Grey Wolf Optimizer. The algorithms were all developed by the same author, Seyedali Mirjalili, and are freely downloadable from the Web. The logic behind them is described as follows.

A. Ant Lion Optimizer

Ant Lion Optimizer (ALO) was developed in 2015, [5]. The idea was inspired by the hunting mechanism of the antlion (a species of insect in the family Myrmeleontidae). The hunting strategy of this animal is based on the use of traps it creates to capture its victims: the ants. The antlion digs a cone-shaped pit in the sand, then conceals under the trap and waits for the victim. As soon as the insect finds the prey in the trap, it starts throwing sand towards the edge of the trap making the victim slump down to the bottom of the trap. After having its meal, it prepares the trap for the next time.

The ALO algorithm takes place in the following 4 main steps: the definition of a random position and a random movement for the ants, the establishing of traps, the enmeshment of ants in the trap, the capture of the victims. Ants and antlions are associated to the possible solutions, at every iteration their fitness is evaluated. Ant move randomly, while antlions are more stationary, hidden in their traps, in defined points of the search space. The position of an antlion represents the best solution found so far in the area around it. This area correspond to the size of the antlion trap, and is proportional to the fitness of the antlion: the greater the fitness, the greater the area of the trap. Each ant is associated to a corresponding antlion, and can only move inside the area defined by the trap. The ant's task is to explore this area looking for a solution with a fitting higher than the fitting of the antlion. If this happens, the position of the antlion is updated with the position of the ant and the radius of the trap is increased accordingly. This phase correspond to the capture of the prey. Finally, the antlion with the best fit is considered as an *elite*, and when optimization ends, the position of this antlion represents the solution of the problem.

In ALO, the random walk followed by ants is modeled by

$$x(t) = [0, S(2r(t_1) - 1), ..., S(2r(T) - 1)] \qquad (2)$$

where $S(\cdot)$ is the cumulative sum, t_i counts the iterations, T indicates the maximum number of iterations of the algorithm, and $r(t)$ is a stochastic function that returns 0 or 1 value with

equal probability. The position of the ants during optimization is stored in the following matrix:

$$\mathcal{M}_A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1d} \\ A_{21} & A_{22} & \dots & A_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nd} \end{bmatrix} \tag{3}$$

where A_{ij} is the position in the j-th dimension of the i-th ant. The number of the ants is indicated by n, while d is the problem dimension. Given $f(\cdot)$ as the fitness function, the fitness values for each ant during optimization are stored in the matrix

$$\mathcal{F}_A = \begin{bmatrix} f([A_{11}, A_{12}, ..., A_{1d}]) \\ f([A_{21}, A_{22}, ..., A_{2d}]) \\ \vdots \\ f([A_{n1}, A_{n2}, ..., A_{nd}]) \end{bmatrix} \tag{4}$$

These two matrices are updated at each step of the optimization. Similar matrices, $\mathcal{M}_{AL}$ and $\mathcal{F}_{AL}$, are defined for antlions. To set the position for the antlions a roulette wheel selection is applied. Antlions are selected based on their fitness values, a higher chance is given to antlions with higher fitness value. To model the falling of the prey in the cone shaped trap, the radius of random walk used in each step is decreased following these equations

$$\begin{aligned} C^t &= \frac{C^t}{I} \\ d^t &= \frac{d^t}{I} \end{aligned} \tag{5}$$

where, C^t and d^t indicates the minimum and the maximum, respectively, of all variables at iteration t, I is the decreasing ratio calculated as $I = 10^w \frac{t}{T}$. In this last equation, T is the maximum number of iteration and w is a constant defined based on the current iteration. The value of w gradually increases as iterations proceed, controlling the accuracy level of the exploitation stage.

To model the effect of antlion's ambush on the random movement of the ants the following equations are used

$$\begin{aligned} C_i^t &= Antlion_j^t + C^t \\ d_i^t &= Antlion_j^t + d^t \end{aligned} \tag{6}$$

in which $Antlion_j^t$ is the position of antlion j at iteration t. Moreover, the random movement of an ant around the corresponding antlion is also affected by the position of the elite; so the position of the i-th ant at the t-th iteration, Ant_i^t, is expressed as

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \tag{7}$$

where R_A^t and R_E^t are the random walks around the antlion and the elite considered at the same iteration.

B. Grasshopper Optimization Algorithm

The Grasshopper Optimization Algorithm (GOA) [6] is a swarm-based algorithm that imitates social behavior of grasshoppers seeking for food.

In their lifetime, these insects pass through two main phases: nymph and adulthood. While nymph, they have no wings so they move slowly and eat all the vegetation on the floor. When adult, they develop wings so they can move fast and fly covering long distances to find a new place with food. The swarming behavior characterizes both nymph and adulthood. The two modality grasshoppers use to search for food can be easily compared to the exploitation and exploration phases in an optimization algorithm.

In GOA, grasshoppers represent the possible solutions. The insects move in the search space, and the fittest one at each step of the optimization is elected as the food position, i.e. the best solution so far. The food position is passed to the other individuals of the swarm requiring them to move toward it. While other grasshoppers move, they can find a fitter position than the previous one. This last position then becomes the position for the food in the new iteration. While the optimization progresses, the algorithm continuously finds better solutions and converges to the optimum.

In GOA the position, X, of i-th grasshopper is given by the formula

$$X_i = S_i + G_i + A_i \tag{8}$$

where S_i, G_i, A_i represent the three forces that act on the insect behavior. The social interaction, the gravitational attraction, and the wind flow, respectively.

The social interaction term, which has the highest impact on the movement, is expressed as

$$S_i = \sum_{j=1, j\neq i}^{N} s(d_{ij})\widehat{d_{ij}}. \tag{9}$$

In the formula, d_{ij} is the distance between grasshopper i and grasshopper j. It is computed as $d_{ij} = |x_j – x_i|$. While $\widehat{d_{ij}} = \frac{x_j - x_i}{d_{ij}}$ is the unit vector from the i-th grasshopper to $j-th$ one. The function $s(.)$ represents the power of social interaction, i.e, attraction or repulsion, among grasshoppers and is expressed as

$$s(r) = fe^{-\frac{r}{l}} - e^{-r} \tag{10}$$

where f represents attraction intensity and l is the the scale of attractive length. Thanks to this formula, the space between two grasshoppers is divided into a repulsion zone (the inner of the hypersphere centered in the i-th grasshopper), an attraction zone (the outer of the hypersphere), and a comfort zone (the surface of the hypersphere) where there is neither attraction nor repulsion. Changing the value of l and f changes the social behavior of grasshoppers.

The gravitational term in Equation (8) is given by

$$G_i = -g\widehat{e}_g \tag{11}$$

where g is the gravitational constant, and $\widehat{e}_g$ is the unity vector in the direction of the earth center.

The last component in the position formula (8) is the wind drift, A_i, computed as

$$A_i = u\widehat{e}_w \tag{12}$$

where u is a constant drift, and $\widehat{e}_w$ is a normalized vector in wind direction. From all the above, the equation that defines the position of the i-th grasshopper can be rewritten as

$$X_i = \sum_{j=1, j\neq i}^{N} s(|X_j - X_i|)\frac{x_j - x_i}{d_{ij}} - g\widehat{e}_g + u\widehat{e}_w \tag{13}$$

where N is the total number of grasshoppers.

However, this formula is a good model for the behavior of the elements of the swarm but needs a further modification to be used for an optimization algorithm. Indeed, the formula quickly leads, and then holds, the grasshoppers in the equilibrium position, without making them converge in a single point. So, it is modified as follow

$$X_i^d = c\left(\sum_{j=1, j\neq i}^{N} c\frac{ub_d - lb_d}{2} s(|x_j^d - x_i^d|)\frac{x_j - x_i}{d_{ij}}\right) + \widehat{T}_d \tag{14}$$

where c is a coefficient which decreases as the iterations proceed shrinking the repulsion region between grasshoppers, ub_d and lb_d are the upper and lower bound in the $d-$th dimension, $\widehat{T}_d$ is the best value found so far in dimension d, and $s(.)$ is approximately the same function used in (8). The G_i contribution is not considered, and it is presumed that the wind (i.e., the A_i term) is always in the direction of $\widehat{T}_d$. The c factor appears twice in the formula because is also used to act on the exploration and the exploitation behaviour. While decreasing, c reduces the movements of grasshoppers around the target, promoting the exploitation at the expense of exploitation. It is updated with the following equation

$$c = c_{max} - l\frac{c_{max} - c_{min}}{L} \tag{15}$$

where c_{max} is the maximum value, c_{min} is the minimum value, l indicates the current iteration, and L is the maximum number of iterations.

C. Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) is inspired by the social and hunting behavior of grey wolves in nature, [7].

Grey wolves live in packs and respect a strict social hierarchy which divides them into 4 main categories: alpha, beta, delta, and omega. The dominant wolf is the alpha, it leads the pack. The beta and delta wolves help the alpha in making decisions about pack activities or accomplice main tasks like hunting or watching the boundaries of the territory. All other wolves in the group are omega individuals. The hunting strategy of grey wolves can be summarized in three phases: searching for prey, encircling prey, and attack prey.

The GWO logic mimic these behaviours. Each wolf is associated to a possible solution. At every iteration the solutions are evaluated and fittest one is taken as alpha wolf, while the second and third best solutions became beta and delta wolves. These three solutions are used to define an estimate of the position of the prey, and this information is then used to update the position of wolves for the next iteration. Wolves have to converge to the prey while diverging form each other. The exploration (diverging) or the exploitation (converging) behavior of wolves can be mitigated or enhanced acting on specific coefficients in the position formula.

The generic position of a wolf around a prey can be expressed by using two equations:

$$\vec{D} = |\vec{C}.\vec{X}_p(t) - \vec{X}(t)| \tag{16}$$

$$\vec{X}(t+1) = |\vec{X}_p(t) - \vec{A}.\vec{D}|. \tag{17}$$

In these equations, t is the current iteration, $\vec{X}_p$ indicates the position vector of the prey, while $\vec{X}$ is the position vector of a grey wolf. $\vec{A}$ and $\vec{C}$ are coefficient vectors given by the formulas

$$\vec{A} = 2.a.\vec{r}_1 \tag{18}$$

$$\vec{C} = 2.\vec{r}_2 \tag{19}$$

where $\vec{r}_1$, and $\vec{r}_2$ are random vectors in [0,1], and a is a coefficient that is linearly reduced from 2 to 0 while the optimization proceeds. Since the position of omega wolves is updated with respect to the alpha, beta, and delta ones, the following equations are used

$$\vec{D}_\alpha = |\vec{C}_1.\vec{X}_\alpha - \vec{X}| \tag{20}$$

$$\vec{D}_\beta = |\vec{C}_2.\vec{X}_\beta - \vec{X}| \tag{21}$$

$$\vec{D}_\delta = |\vec{C}_3.\vec{X}_\delta - \vec{X}|. \tag{22}$$

$\vec{X}_\alpha$, $\vec{X}_\beta$, $\vec{X}_\delta$ indicates the position of alpha, beta and delta wolves respectively, and C_1, C_2, C_3 are random vectors. To estimate the approximate distance between the current solution and alfa, beta and delta wolves, equations (20), (17), and (22) are applied as follow:

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.(\vec{D}_\alpha) \tag{23}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2.(\vec{D}_\beta) \tag{24}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3.(\vec{D}_\delta) \tag{25}$$

Then, the position of the current solution can be updated for the next iteration, and becomes

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}. \tag{26}$$

It can be observed that two vectors, $\vec{A}$ and $\vec{C}$, are defined. These two vectors are random and are used to tune the strength of the exploration and exploitation phases in the GWO algorithm.

D. Dragonfly Algorithm

The last algorithm evaluated in the present work is the Dragonfly Algorithm (DA), [8]. The DA is an optimization method inspired by the swarming behavior of dragonflies. They exhibit two different attitudes while flying: a static feeding swarming, and a dynamic migratory swarming. These two habits loosely mimic the characteristics of the two phases of an optimization process: exploration and exploitation.

The base principles used in this algorithm are the same swarm principles identified by Reynolds in [9] and [10]. Elements in a swarm follow three fundamental rules while moving:

- Separation, which means avoid collisions between individuals;
- Alignment, which indicate individuals should have the same velocity;
- Cohesion, which refers to the tendency of individuals to converge to the center of the mass of the swarm.

Two more principles can be added to the previous three:

- Attraction towards food, which is the main aim of the swarm;
- Distraction outwards enemy.

In the following, a model for each principle is provided. The separation term is given by

$$S_i = -\sum_{j=1}^{N} X_i - X_j \tag{27}$$

where the position of the current individual is indicated by X_i, N is the number of agents in X_i's neighborhood, and X_j is the position of the j-th neighboring individual. A radius r is used to define a region of visibility for the considered dragonfly, all dragonflies inside this region are taken in account in the calculation of S_i. The same visibility rule is used for the other behavioural terms.

The alignment is computed as

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \tag{28}$$

where V_j represents the velocity of the j-th neighboring individual. While the cohesion is given by

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X_i. \tag{29}$$

Attraction towards the food source and distraction outwards the enemy are computed as

$$F_i = X^+ - X_i \tag{30}$$

$$E_i = X^- + X_i \tag{31}$$

where X^+ indicates the position of the food source, while X^- is the position of the enemy.

Agents move within the search space and their position is updated at every iteration with the rules:

$$X_{t+1} = X_t + \Delta X_{t+1} \tag{32}$$

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_t \tag{33}$$

The vector ΔX is the velocity vector of the dragonfly and contains information about the direction and speed of the movement. The constant w is an inertia weight, and t is the iteration counter. The coefficients s, a, c, f, e are used to weight the different behaviours. In the first steps of the optimization process the dragonflies fly in a dynamic swarm. The fly is highly coordinated, dragonflies speeds are aligned and cohesion and separation are high. This is the best organization for exploring the search space. While the optimization proceeds, the swarm behavior changes and gradually becomes a static swarming. In this last stage the cohesion increases while alignment and separation decrease, so that the dragonflies can converge towards the prey exploiting the search space. The transition between the two flight modes occurs by acting on the radius r and on the coefficients s, a, c, f, e. Their values are modified with the increasing number of iterations. A last note should be added about food and enemy position in DA. They are updated at the beginning of every iteration, when the fitting of all dragonfly is evaluated. The position of the fittest one is taken as the food location, while the position of the worst one is considered as the place where the enemy is.

III. Methodology

The simple logic and the easy equations of the presented algorithms have made them of interest for the industrial problem addressed. Since the idea is to implement a virtual sensor, that is a software sensor, the occupation of memory and the computational effort are taken carefully into consideration. These aspects are even more important when dealing with industrial devices where software resources are often tailored to the primary needs of the system. In order to use swarm algorithms to address the backlash problem this last must be translated into an optimization problem. Details on this are presented below.

The backlash phenomena arises inside the joint, but its effects propagate both to the link and to the motor that are connected at the input and output ends of the joint, see Figure4.

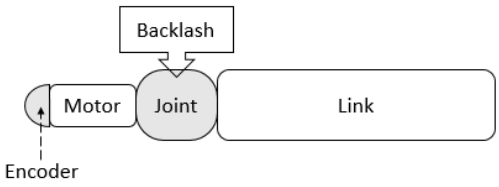

Fig. 4: Simplified model of a robotic joint.

In standard industrial robots there are no sensors in the joint, nor on the link. While an encoder is always present on the motor. The encoder is used in position and speed control loops of the motor itself. By detecting the consequences on the motor of the backlash in the joint, is possible to derive information

on the backlash presence in the joint itself. However, the detection is not so simple because these effects can reach the motor strongly attenuated and be added or confused with other sources of disturbance.

The main idea of this virtual sensor is to identify a model as the signature for the backlash effect on the motor and look for this pattern inside the signal from the encoder. When the model is detected, its parameters are measured end an estimate of the backlash level in the joint is derived. The greater the amplitude of the disturbance, the greater is the backlash.

The matching between the model and the signal, and the measurement of the model parameters are performed through the optimization algorithm. All the information about the cost function, the parameters, the search space and the constraints of the problem are described in what follows.

A. The objective function

For the problem of interest, Root-mean-square error (RMSE) between data and model is the objective function to minimize. Since data show a repetitive pattern, made of positive and negative sequential oscillations, the disturbance signature (1) has been slightly modified to replicate this behavior too. The resulting formula is

$$h(t, A, t_0, \tau, \omega, v_t, t_1, t_2, T, T_w) = v_t + \sum_{i=1}^{12} f(t, A, t_0 + i \cdot T_w, \tau, \omega, t_1, t_2, T) \quad (34)$$

where v_t is the commanded motor speed, and the function f is the sequence of a positive and a negative, T shifted, disturbance oscillation

$$f(t) = d_b(t - t_0) - d_b(t - t_0 + T). \quad (35)$$

The t_0 parameter is the disturbance starting time while T_w is the time period for the $f(t)$ function.

So, it turns out that the objective function is

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{N} \Big(v(t) - h(t, A, t_0, \tau, \omega, v_t, t_1, t_2, T, T_w) \Big)^2}{N}}$$

where N is the number of samples in the dataset.

B. The parameters

The parameters to be identified are

$$X = [A, t_0, \tau, \omega, v_t, t_1, t_2, T, T_w] \quad (36)$$

The vector dimension is 9 but, since T and T_w have fixed and known in advance values, the problem dimension becomes 7. Considering that all the parameters have a physical meaning and that some information about the system under test is a-priori available, it is possible to use it to define upper and lower bounds for the parameters. In particular, boundaries for amplitude related parameters of the model, A and c, are proportional to the amplitude of the signal to be analyzed. We know in advance they have increasing amplitude because the disturbance superimposed on it is growing.

TABLE I: Parameters Variability Range

Symbol	Min_value	Max_valuee	Units
A	$-\frac{1}{3} * (\frac{\max v(t) - \min v(t)}{2})$	$\frac{1}{3} * (\frac{\max v(t) - \min v(t)}{2})$	rpm
t_0	$\min t$	$\max t$	s
τ	5	30	-
ω	2π	$2\pi \cdot 40$	rad/s
v_t	$\min v(t)$	$\max v(t)$	rpm
t_1	0	0.204	s
t_2	0	$\frac{0.204}{2}$	s
T	0.204	0.204	s
T_w	0.408	0.408	s

NOTE: t is the time vector of the measured signal v(t)

Furthermore, these parameters must have a definite relationship to one another to ensure that the aspect of the function respects the expected one. In particular the constraints are defined for t_0, t_1, and t_2 and are: $0 < t_0 < (max(t) - t_1)$, $t_2 < t_1 < T$ and $0 < t_2 < pi/(2\omega)$.

At the end of every iteration, when the position of the agents is updated for the next iteration and after the check to verify if the new values fall within the defined domains, the constraint check is also performed. If the value of a parameter is outside the interval defined by the constraints, then the value is saturated to the extreme of the allowable range.

To perform a comparison, the algorithms are run under the same conditions: same number of agents and iterations, same constraints and same input data. All algorithms used are applied in their original *Matlab* implementation. No changes are made to the code other than minimal customization to fit the problem.

C. The datasets

Each algorithm is run on the same test set. The test set comprises 20 signals, all affected by a different level of backlash.

Signals are obtained with the help of simulation. The Matlab/Simulink environment is used to model the robotic joint and record the motor encoder signal when the system is affected by an increasing value backlash (see Figure 3). Dataset1 contains the signal that corresponds to the lowest level of backlash, while Dataset20 contains the data corresponding to the highest backlash. Figure 5 shows the datasets and the corresponding values for the backlash (in blue) and for the disturbance amplitude (in red). The datasets correspond to the 20 backlash values that span the interval $[\delta_{min}; \delta_{max}] = [0.0001; 0.004] radians$ with the constant step $\Delta\delta = 0.002 radians$. Since it represents the minimum distance between two possible measures, the step was also taken as the accuracy value required for the virtual sensor.

D. The optimization

The number of agents and the number of iterations used for the four algorithms is the same. The best setting is identified in 300 agents and 200 iterations. The value is chosen taking the DA algorithm as a reference and following a trial and error strategy. Since the aim of our study is to compare

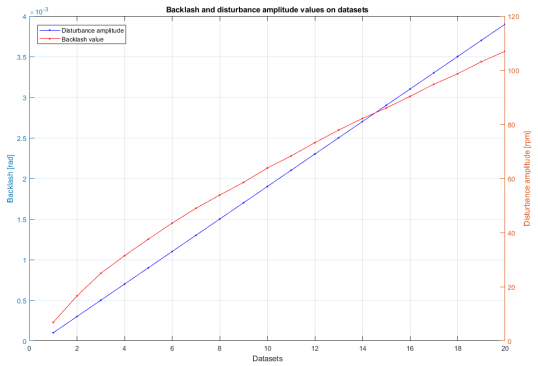

Fig. 5: Backlash and expected disturbance amplitude on the considered datasets.

the performances of the different optimization methods, it is decided to proceed under the same conditions for all the algorithms and use $[300, 200]$ as the common setting.

The workstation used for carrying out the result is equipped with a 2.70 GHz processor, 16 GB of RAM, and the system used is Microsoft7. To run the proposed algorithm the Matlab software is used.

IV. Experiments

To evaluate the performance of the different algorithms with respect to the problem of interest, an extensive test campaign is conducted. The four algorithms are run on the backlash measure problem described in Section I, and the identification is performed on the datasets described in Section III. Since preliminary tests have shown that in the first two datasets the backlash disturbance is too small and cannot be properly identified, these dataset are ignored in the analysis. The remaining 18 datasets have served as reference for the assessment. Thirty independent experiments are performed for each set while all conditions are kept as constants. The same setting is used for all tests and for all the algorithms. Furthermore, to provide each algorithm with equal opportunities, the starting solution is always randomly chosen.

A collection of all results is shown in Figure 6. Graphs in the upper part summarize the values of the backlash disturbance amplitude provided by the different optimization methods. Dots stacked in each column correspond to the 30 outcomes of experiments on a same dataset. The red line in the graph connects the expected values for the disturbance amplitude. While the red dotted lines represent the maximum allowable error for the measure, i.e. the accuracy on δ reported on the value of A. Since the relationship between A and δ is not linear, the accuracy on A is taken as the minimum distance between two consecutive values of A plotted in Figure 5. Looking at the figures, it can be noted that the distribution of results is always widening while moving towards larger values of the backlash, i.e., going left to right in the plots. The reason behind this is not the increased difficulty of the problem, but is the way the problem search space is defined. In fact, amplitude related parameters of the model, A and c, have boundaries that grow while moving through the datasets on the right leading to an increase in their search space, see Section III-B. It is also possible to note that it is more likely that the swarm optimization algorithms tend to underestimate, rather than overestimate, the solution. It can be attributed to the fact that the fitting result is sometimes wrong. When fitting fails, only the second and smaller half of the decreasing oscillation is fitted, while the first and largest part is lost. In such cases an underestimated solution is provided.

A different perspective on the same data is given by the box and whisker diagrams, in Figure 7. They present in quantitative terms the dispersion of the data around the expected value. Diagrams give clear information about GOA, ALO and DA performance, but unfortunately are not significant for GWO. This is because the particular shape of the dispersion of GWO results, with the majority of them lying very close to the red expected value and the wide empty band separating the remaining data (see GWO graph in Figure 6), makes the box plot misleading. So, even if box charts show ALO as the best in terms of accuracy, a more in-depth analysis returns GWO as the best one.

Since comparing the outcome of stochastic algorithms is not trivial, as just examining the mean performance is not enough, the results are also evaluated through a statistical test procedure. The series of 30 results obtained from experiments on the single dataset is treated as a set of samples from a continuous distribution. The two-sample Kolmogorov-Smirnov test [11] is used to check the hypothesis that two different series of results come from the same distribution. In particular, the test is run with $\alpha = 0.05$ as the threshold for significance of distinctiveness. The results are shown in Table II. The reported value is the mean of the fitting error on the 30 repeated experiments. Italic is used for results that are likely form the same distribution. Corresponding values are considered as not significant. For the majority of cases, however, it is possible to accept the hypothesis that the data collected come from different distributions. In these cases the best result is indicated in bold. Again, GWO appears as the best of the four algorithms.

TABLE II: Kolmogorov-Smirnov test results

Datasets	ALO	GOA	DA	GWO
3	2.777	4.781	3.227	**2.637**
4	3.331	6.169	4.558	**3.278**
5	4.560	7.214	6.039	**3.327**
6	5.752	9.511	7.422	**3.630**
7	6.728	*8.946*	*9.277*	**4.250**
8	6.622	11.628	9.363	**5.400**
9	7.580	*12.163*	*11.553*	**5.817**
10	8.816	13.065	11.412	**7.310**
11	9.456	*14.159*	*13.399*	**7.957**
12	8.647	16.212	12.846	**8.268**
13	10.218	16.571	14.409	**8.844**
14	10.729	18.311	14.321	**9.289**
15	10.957	17.419	16.261	**10.411**
16	**10.427**	19.766	16.709	10.509
17	11.301	20.982	16.921	**10.521**
18	11.932	20.055	18.920	**11.017**
19	13.628	20.69	17.799	**12.576**
20	13.660	22.126	19.150	**11.214**

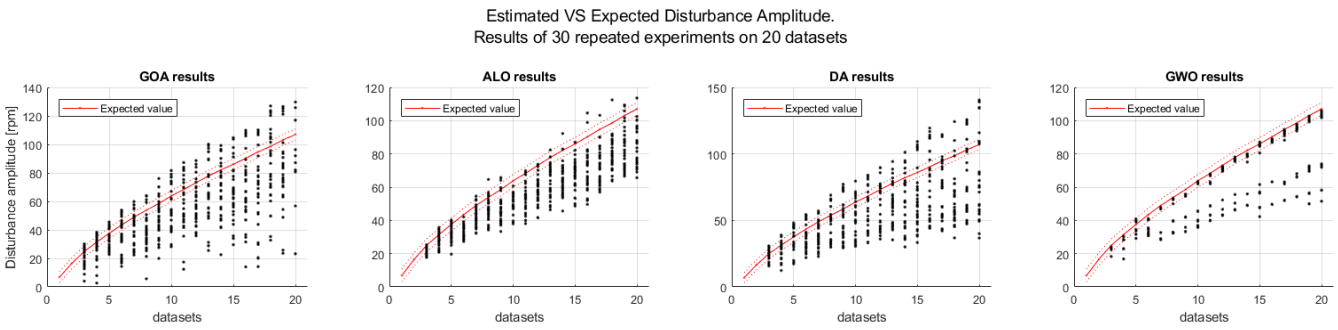

Fig. 6: Collection of all the results of the experiments.

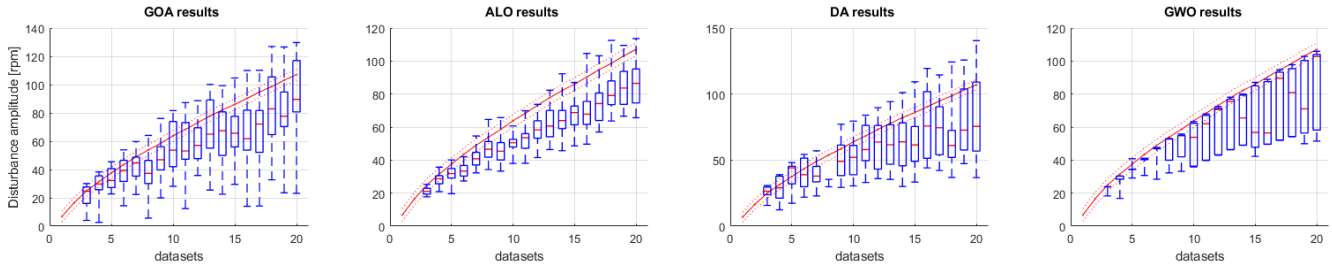

Fig. 7: Box and whisker diagram.

For all the above, it can be stated that the GWO algorithm has the best performances with respect to the backlash problem. While, DA is the worst. Nevertheless, none of the algorithms analyzed presents performances that can meet the requirements of precision and repeatability of our sensor. Figure 6 clearly shows the solutions lying outside the allowable band delimited by the desired accuracy. Therefore, at the present stage of development, these algorithms cannot be used for the implementation of the virtual sensor for the backlash.

V. Conclusions

The paper presented an application of four bio-inspired swarm algorithms to a real industrial problem. A extensive test campaign was conducted and a thorough comparison of results and performances of the algorithms was performed. Key aspects as accuracy, precision, and ease of implementation were taken into account, focusing on the possible use of the swarm algorithms for the implementation of a virtual sensor. The problem of interest, i.e., the matching between a disturbance signature and a signal, has proved difficult to solve, and the considered algorithms was found to be unsuitable to find its solution. The presence of many local minima has been observed, and none of the algorithms showed sufficient exploration capabilities not to get stuck in one of the many local solutions. It has been verified that, even showing some advantages in terms of memory occupation and ease of implementation, the bio-inspired swarm algorithms can not be used for the backlash sensor implementation. They were not able to guarantee the level of reliability and accuracy required by the sensor.

References

[1] M. Edwards, "Robots in industry: An overview," *Applied ergonomics*, vol. 15, no. 1, pp. 45–53, 1984.

[2] J. L. Stein and C.-H. Wang, "Estimation of gear backlash: Theory and simulation," 1998.

[3] Q. Yang, T. Liu, X. Wu, and Y. Deng, "Gear backlash detection and evaluation based on current characteristic extraction and selection," *IEEE Access*, vol. 8, pp. 107 161–107 176, 2020.

[4] E. Giovannitti, G. Squillero, and A. Tonda, "Virtual measurement of the backlash gap in industrial manipulators," in *Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing*. Springer, 2019, pp. 189–200.

[5] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997815000113

[6] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah, "Grasshopper optimization algorithm for multi-objective optimization problems," *Applied Intelligence*, vol. 48, no. 4, pp. 805–820, 2018. [Online]. Available: https://doi.org/10.1007/s10489-017-1019-8

[7] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997813001853

[8] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective,discrete, and multi-objective problems," *Neural Computing and Application*, vol. 27, pp. 1053—-1073, 2016.

[9] C. Raynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphic- ACM SIGGRAPH '87 Conference Proceedings*, vol. 21, no. 4, pp. 25–34, 1987.

[10] C. Raynolds, "Steering behaviour for autonomous characters," *http://www.red3d.com/cwr/steer/, first version*, 1999.

[11] N. V. Smirnov, "On the estimation of the discrepancy between empirical curves of distribution for two independent samples," *Bull. Math. Univ. Moscou*, vol. 2, no. 2, pp. 3–14, 1939.