Neurocomputing 647 (2025) 130551

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Slime Mould Metaheuristic for optimization and robot path planning

Enol García [a],*, José R. Villar [a], Javier Sedano [b], Camelia Chira [c], Enrique de la Cal [a], Luciano Sánchez [a]

[a] Department of Computer Science, University of Oviedo, Campus de Viesques s/n, Gijøn, 33204, Asturias, Spain
[b] Instituto Tecnológico de Castilla y León, Polígono Ind. Villalonquéjar, C/Lopez Bravo, 70, Burgos, 09001, Castilla y León, Spain
[c] Department of Computer Science, Babes-Boliay University, Faculty of Mathematics and Computer Science, Kogalniceanu 1, Cluj-Napoca, RO-400084, Romania

ARTICLE INFO

Keywords:
Evolutionary algorithms
Metaheuristics
Robot path planning

ABSTRACT

Function optimization represents a remarkable challenge in industry and society, aiming to find reasonable solutions – even if they are suboptimal – for everyday problems. Metaheuristics drive the optimization search towards the goals using a specific algorithm inspired by different concepts: from industrial processes to the behaviour of living beings in nature, from mathematical ideas to physics notions.

This research proposes a new metaheuristic inspired by the Slime Mould and its foraging behaviours. On the one hand, an exploitation stage mimics the greedy amoeba's conduct when food is plenty. On the other hand, an exploration stage copies the fruity aggregation of the cells and the subsequent spore dissemination. This study compares the most cited metaheuristics and the Slime Mould Optimization in two different experimentation stages: on the one hand, the optimization of standard benchmarking functions; on the other hand, solving the robot path planning problem. Moreover, a hybridization of the SMO and the WOA is presented, which keeps the SMO's convergence speed and the WOA's good performance in finding the best solutions.

1. Introduction

Metaheuristics have emerged as a promising solution to solving mathematical optimization problems, searching for a minima or a maxima of one or more objective functions subject to predefined constraints. The applicability of metaheuristics to a widespread of different knowledge areas and domains has been successfully reported in the literature, either in medicine [1–3], engineering [4–6] or economics [7–9].

Optimization problems need feasible solutions in a reasonable time; reaching these objectives becomes a challenge with the number of functions to optimize and the number of constraints to consider. Metaheuristics represent a compromise in the search process by finding suboptimal feasible solutions in finite and limited time. Metaheuristics are typically inspired by the behaviour of existing aspects of the world that have proven to work well, such as the evolution of species, foraging behaviours of animal species, or physical and mathematical concepts. This inspiration and adaptation of existing concepts to the optimization field makes metaheuristics very versatile algorithms capable of tackling any problem with a minimal amount of changes.

This research proposes a new metaheuristic inspired by the slime mould, a fungus that has demonstrated extraordinary capabilities in food search in nature [10–12]. On the one hand, they behave as independent cells when there is plenty of food in the environment, draining their neighbourhood. On the other hand, when food is scarce, the cells agglomerate in a body that moves through the local area until it reaches a suitable location; at this point, the body morphs into a fruity body that expells spores for settling in remote and more promising areas. These behaviours allow the mould to find food in complex laboratory environments – such as mazes –, and are the motivation of this research. The Slime Mould Optimization algorithm (SMO) swaps from exploitation to exploration and vice versa when the fitness improvement stagnates, mimicking the mould's changes behaviour with the food availability. Exploitation introduces a local search in a neighbourhood that shrinks with fitness improvement, while exploration allows individuals to explore broader areas in the domain. We evaluate the performance of this metaheuristic in two domains – a challenge for function optimization and the problem of single robot path planning in three different scenarios –, comparing the SMO against the most recently cited metaheuristics.

The structure of this study is as follows. The next section presents the related works on metaheuristics, considering the different strategies and classifications of metaheuristics according to several authors. Then,

* Corresponding author.
E-mail addresses: garciaenol@uniovi.es (E. García), villarjose@uniovi.es (J.R. Villar), javier.sedano@itcl.es (J. Sedano), camelia.chira@ubbcluj.ro (C. Chira), delacal@uniovi.es (E.d.l. Cal), luciano@uniovi.es (L. Sánchez).

https://doi.org/10.1016/j.neucom.2025.130551
Received 3 June 2024; Received in revised form 17 December 2024; Accepted 17 May 2025
Available online 3 June 2025

Section 3 describes the new metaheuristic, emphasizing its natural inspiration and mathematical approach. Section 4 gives details on the experimentation design, with two main comparisons: one in benchmark functions, and one in robot path planning. In these comparisons, the Slime Mould Optimiser is put alongside the most cited metaheuristics in the literature. Results and the discussion on them are developed in Section 5. The study ends by drawing the corresponding conclusions.

2. Related work

Metaheuristics may be classified following diverse criteria, leading to different taxonomies. Examples of these taxonomies are the one presented by Fister et al. [13] or the one suggested by Gendreau and Potvin [14], although there are more taxonomies in the literature [15].

Fister et al. proposed a simple two-level taxonomy that classifies the metaheuristics based on their inspiration. The first level labels the metaheuristics as *nature-inspired* and *non-nature-inspired*. The motivation source is the criteria used in assigning a second-level term to nature-inspired metaheuristics, labelling each one as based on swarm intelligence, based on biology, based on physics/chemistry, or other nature-inspired algorithms.

Besides, Gendreau and Potvin propose grouping the metaheuristics into two categories according to the solutions' update method [14]. Hence, there are trajectory-based metaheuristics or population-based methods. Trajectory-based metaheuristics are local-search oriented, substituting the current solutions with the newly generated ones. On the other hand, population-based metaheuristics are focused on exploration, producing solutions based on the current population, and substituting part of the current population with the newly created individuals.

The efficacy of metaheuristics lies in the symbiosis of two search paradigms according [16]: exploration and exploitation. Exploration navigates the vicinity of optimal solutions, while exploitation ventures into uncharted territories. Despite metaheuristics' efficacy in problem-solving, their universality extends only to a confined set of problems, as postulated by the no-free lunch theorems [17].

This fact has catalyzed the proliferation of diverse metaheuristics tailored to address a gamut of problem typologies; the remainder of this section describes the metaheuristics with a significant number of cites in the literature – in the order resulting from the Fister et al.'s taxonomy –, followed by a discussion of the pros and cons that these methods achieved.

Particle Swarm Optimization PSO [18] is the most recognizable nature-inspired metaheuristic based on swarm behaviour. This metaheuristic works on a population of solutions scattered around the world, where each solution has a velocity of motion and, in an iterative process, updates its position using its velocity of motion. The swarm intelligence comes into play in updating that velocity since, after each movement, the velocities of all individuals drift following the best solution present in the population at the current time and the best global solution found in any iteration.

Gray Wolf Optimization (GWO), Whale Optimization Algorithm (WOA), and Moth Flame Optimization (MFO) are examples of metaheuristics inspired by biology, as well as the whole family of Genetic Algorithms (GA). While GA belongs to the category of population-based algorithms [14], PSO, GWO, WOA and MFO are trajectory-based methods. GWO mimics the social behaviour of a pack of wolves, remarking on the role of each individual in the population [19]. Therefore, wolves are labelled as Alpha, Beta, Delta, or Omega according to their strength and influence in the pack. Similarly, GWO labels each solution within the population with a social role – Alpha, Beta, and Delta – according to their fitness value. The mating probabilities vary according to these roles, diverting the evolution of the population towards the best candidates.

WOA imitates the foraging method of whales in the ocean [20], independently of the position of other whales in the population: whales search for food autonomously. When far from a prey, the whales swim randomly, trying to find its trace; when they are close to a prey, they encircle it with a bubble net: spiral movements that trap the prey inside, finally eating it. The WOA metaheuristic works similarly by randomly exploring the domain; when a fitness improvement is detected, WOA performs a local search through the spiral movements of the affected individuals.

Besides, MFO copies the erratic movements towards light sources of the moths [21], proposing two subpopulations that evolve in parallel: one population are moths that move towards their assigned flame — one of the best individuals found so far. The second population collects the best individuals – flames – found so far by any moth, updating this population at each iteration. Assigning a flame to each moth defines the moth's movements.

Two other examples of biology-based metaheuristics that mimic animal behaviour are Fire Hawk Optimizer (FHO) [22] and Artificial Jellyfish Search Algorithm (AJSA) [23]. FHO is a metaheuristic that emerged after observing the behaviour of some birds in Australia during a bushfire. While most animals tend to flee or hide from a fire, the fire hawks try to spread the fire to shoo away the possible prey, so they become easily hunted. In the FHO, prey and hawks are search agents; hawks are the better-fitted individuals in the population. Pray agents are linked to their closer hawk, randomly moving towards the hawk or flying away. Thus, the FHO is a trajectory-based optimization algorithm.

AJSA is also a trajectory-based algorithm inspired by the swarm foraging process of jellyfish. Jellyfish follow the currents in search of food – passive movements –; changing to active movements when they detect a promising area. These two behaviours swap from one to another according to predefined control criteria. The main mathematical foundation of this metaheuristic is that it considers the trend of the values in the search space to determine the vector of their movement.

GA mimics the species' reproduction and mutation behaviours throughout their evolution. At each iteration, a population of individuals reproduces and mutates to generate the siblings. Before the end of the iteration, a selection process mimics natural selection in which the stronger individuals – in this case, the best solutions – have a higher probability of survival than the weakest individuals. Differential Evolution (DE) [24] represents an example of a metaheuristic belonging to the GA family.

Additionally, methods such as Simulated Annealing (SA) or Chemical Reaction Optimization (CRO) belong to the nature-inspired algorithms following physics or chemical rules; both methods are considered trajectory-based. SA [25] is a local search-based metaheuristic that mimics the cooling process of metals after undergoing an annealing process: starting from a randomly generated solution at an initial temperature, the SA's iterative process updates the solution by finding an alternative in its neighbourhood; the temperature variation law delimits the width of this neighbourhood. The SA solves the exploration/exploitation dilemma with the temperature: the higher the temperature, the higher the chances of exploration – bigger neighbourhoods –; and vice versa, the lower the temperature, the higher the chances of exploitation — smaller neighbourhoods. On the other hand, CRO [26] simulates the interactions of molecules in a reaction to achieve a stable low-energy state. For this purpose, it proposes a population of molecules characterized by kinetic and potential energies, which drive the population update.

Similarly, Energy Valley Optimizer (EVO) [27] and Atomic Orbital Search (AOS) [28] are recent physics-based metaheuristics. Both metaheuristics focus on the concept of stability of atoms but do so at a different level. EVO considers the emission of radiative atoms due to the differences between protons and neutrons. Mathematically, the metaheuristic poses a scenario in which the atoms are the search agents that move through the solutions space, looking for the best one. This metaheuristic defines the updating of positions based on the different types of radiations emitted by radioactive particles; thus, it can be considered a trajectory-based method.

AOS is related to EVO but at a lower level of abstraction. Instead of positing the scenario in which we have many unbalanced atoms, it posits the structure of a single atom scythed by the imbalance between electrons and protons. When that happens, the electrons, which are moving in different shells at different distances from the nucleus, change levels to move closer or away from it; in this way, the atom would naturally tend to an equilibrium state. Mathematically, AOS models a scenario with the best global solution found as the atom's nucleus, and the electrons are the search agents. They update their position iteratively, deciding at each iteration whether to move towards the nucleus or away from it.

Metaheuristics such as Harmony Search (HS) or Sine Cosine Algorithm (SCA) are arguably inspired more by human abstract concepts than by nature. In the last category of other naturally inspired metaheuristics, the author includes those inspirations that have a smaller number of published metaheuristics, such as Harmony Search (HS) [29,30], which is inspired by musical concepts of harmony, or Sine Cosine Algorithm (SCA) [31,32], which is inspired by a mathematical concept. HS is a population-based algorithm that finds optimal points in the domain using either a local search from one of the individuals in the population or a randomly generated individual; each behaviour has its corresponding and complementary probability. Furthermore, SCA is a trajectory-based method that creates a random initial population, changing the individuals' positions according to several mathematical functions using the $\sin$ or the $\cos$ function according to the distance to the best individual so far.

Chaos Game Optimization (CSO) [33] is inspired by the mathematical theory of chaos, more specifically, in the chaos games. This game creates fractals starting from a random point of an initial polygon, generating shapes using chaotic shapes, such as the Sierpinski triangle. Hence, from the random initial positions of the individuals, the method iterates by generating triangles for each point, evaluating the vertexes, and substituting the current point with the best fitness vertex.

Humanism and social aspects also drive nature-inspired metaheuristics, such as Squid Game Optimizer (SGO) [34] and Adolescent Identity Search Algorithm (AISA) [35]. SGO is a metaheuristic based on a Korean children's competitive team-based strategy game that is played on a squid-shaped field drawn on the ground. The team inside the squid must prevent an attacking team from entering the squid's head. Attacking team members must not touch any defendant while trying to sneak into the goal area. SGO uses players as search agents, updating their position in each iteration depending on their role: an attacker moves toward its closer defendant, while defendants choose between moving toward an attacker or moving toward the attackers' centroid.

Besides, adolescents' human social behaviour inspires AISA, focusing on how adolescents decide to follow others on social media networks. AISA proposes three different patterns: following the best individual, the worst, or another individual that fulfils some condition. Using these patterns, AISA updates the position of the search agents in the population iteration after iteration.

Finally, examples of non-nature-inspired metaheuristic are the Greedy Randomized Adaptive Search Procedure (GRASP) and Tabu Search (TS). The construction-based metaheuristics start from an empty solution that iteratively incorporates new elements until building up a final solution. An example of this type of metaheuristics is the Greedy Randomized Adaptive Search Procedure (GRASP) [36]. In each iteration, GRASP selects the component from a list of ranked elements according to a greedy function, adding the component to the partial solution. The selection can be made according to the rank or randomly among the subset of the best candidates. TS is similar to SA in that it performs a local search by analysing the vicinity of the current individual [37]. An individual in the vicinity with better fitness than the current one substitutes this latter when it has not been considered before. The size of the list of considered individuals is continuously monitored, limiting the total memory size.

Metaheuristics are algorithms that have proven very efficient in solving optimization problems. However, comparative analyses of metaheuristics in the literature have shown that there is much room for improvement in the performance of these metaheuristics [38–42]. WOA, for example, is a metaheuristic that has demonstrated a great capacity to reach reasonable solutions to any problem, but it is one of the metaheuristics that requires a longer execution time. Others, such as PSO, GWO, DE, or MFO, considerably reduce the cost of execution to reach reasonable solutions, but in exchange, they pay for it with their capacity to reach the optimal solutions to any problem. Metaheuristics such as GWO or PSO are good in unimodal problems, which are those in which there is only a local minimum in the optimization, which means that the whole search space tends to converge to that point. Other metaheuristics, such as DE or MFO, enhance the exploration of the whole search space, making them suitable for multimodal problems with multiple local minima. In these problems with multiple local minima, metaheuristics such as GWO tend to get stuck in a local minimum and do not reach the global minimum.

This observation, applicable in the comparisons found in the scientific literature, raises the need to develop a metaheuristic that combines all these advantages of the different metaheuristics. It is desirable to propose a metaheuristic with a good exploration and exploitation capacity, as is the case of WOA, avoiding the problem of GWO, MFO, or DE, which sacrifice one of the behaviours in exchange for a lower computational cost. At the same time, it is also an objective to create more efficient metaheuristics so that the quality of both behaviours is not sacrificed in the computational cost, as in the case of WOA. If all the advantages of these metaheuristics could be combined in a new metaheuristic, we would like to obtain a metaheuristic with a very low computational cost as DE but with a great optimization capacity in all kinds of problems, as in the case of WOA.

3. A slime mould inspired optimizer

This section aims to describe the SMO, a new bio-inspired metaheuristic for optimization. Firstly, the following subsection deals with the natural basis that inspired this study, discussing how some concepts found in nature address the exploitation–exploration dilemma. Afterward, Section 3.2 describes the mathematical model of the proposal.

3.1. Inspiration from nature

The mucilaginous fungus, also known as Slime Mould, is a fascinating single-celled organism with surprising foraging behaviours. Surprisingly, they become multicellular beings when food is scarce, using several different behaviours to increase the probability of finding bacteria they feed from; exploration and explosion are the two most remarkable of these behaviours [43,44].

When the food is plentiful, the Slime Mould does not move and exploits the environment to the end of existence; at this stage, the mould behaves as a unicellular entity. However, when food becomes scarce, the mould becomes a multicellular body, exploiting its neighbourhood. In this case, the cells move in different directions of the nearby area, rapidly expanding towards a new food source.

The Slime Mould applies a different strategy as food becomes increasingly scarce: it branches. The mould forms fruiting bodies and explodes, spreading cells over a broader area in the search for new food sources. With this exploration behaviour, the organism generates a branching network that extends in multiple directions while scouting for new food sources, detecting evidence of potential food as specific chemical compounds. The fungus moves in the most promising directions; hence, continuous branching and thorough network exploration maximize the fungus's chances of finding food in a highly challenging environment.

Both behaviours – exploration and exploitation – represent adaptive strategies that allow the mucilaginous fungus to survive in challenging environments, finding food under adverse conditions. Branching ensures a thorough exploration of the environment in search of new food sources, while neighbourhood search takes advantage of the available food in the Slime Mould location. These behaviours teach us about the adaptive capacity of organisms and how they can use different strategies to find resources and ensure their survival in changing conditions.

In this research, we design a new optimization metaheuristic by incorporating these mucilaginous fungus behaviours. The exploitation behaviour conduits the trajectory-based search process; however, the exploration behaviour assists the optimizer when the improvement in the population's fitness remains almost constant. These operations drive the optimizer to vary from local to global optimum and vice versa, so the fitness keeps improving. In this sense, the metaheuristic is a hybridization of trajectory-based and population-based metaheuristics, taking the best from each type: finding good local candidates and browsing the domain thoroughly.

3.2. *Mathematical model and algorithm*

3.2.1. *General behaviour*

As explained before, food search drives the general behaviour of the mucilaginous fungus, making a move whenever food becomes scarce — either as a shift to the neighbourhood or as a long jump to explore remote areas in the domain. A Slime Mould colony compacts again anytime it finds a new food source, exploiting and consuming the available food in the area and breeding. Besides, the Slime Mould is an unicellular being that behaves as a multicellular body anytime food availability reduces. With all these ideas in mind, we designed the proposed metaheuristic, combining these two exploration and exploitation strategies — see Algorithm 1 for a description of the SMO.

Algorithm 1 SMO basic loop, $O(n)$

```
Require:
  population size ($pop\_size$),
  number of iterations ($max\_iter$),
  number of iterations without fitness improvements ($wait\_iterations$)
  maximum burst radio ($max\_r$)

  Strategy ← *Exploitation*
  Create the initial population with $pop\_size$ individuals
  for $iter \in \{1, \dots, max\_iter\}$ do
      if no fitness improvement during $wait\_iterations$ then
          Switch Strategy between *Exploitation* and *Exploration*
      end if
      Selection of an individual using fitness weighting
      if Strategy is *Exploitation* then
          $new\_inds$ ← call Algorithm 2 for exploitation
      else                                        ▷ Strategy is Exploration
          $new\_inds$ ← call Algorithm 3 for exploration
      end if
      Population update
  end for
```

We denote the fitness function value improvement – increasing it when maximizing, decreasing it otherwise – as the measure of food availability. Moreover, a Slime Mould colony comprises a predetermined number of individuals (pop_size), a metaheuristic's hyperparameter.

An individual represents a hypothesis in the optimization problem. We assume that an individual represents the hyperparameters of a model; without loss of generality, an individual includes a vector of rational numbers whose length is N. Additionally, the collective of individuals decides their strategy, each individual acting coherently. Furthermore, to mimic the agglomerative capacity of becoming a body, each individual x receives a *cardinality* (C_x); this cardinality represents the number of individuals that agglomerate in a single position, moving and acting as a single entity. The sum of the cardinalities for all the individuals is always equal to the population size. The cardinality becomes relevant in the exploitation – as will be explained in Section 3.2.2 –, while in the exploration strategy, it only increases the pressure selection among the different individuals in the population — see Section 3.2.3 for further details.

The strategy switches from exploitation to exploration – and vice versa – when the optimiser runs more than $wait\_iterations$ iterations without fitness improvement. Having no fitness improvement means that percentual variations in the fitness keep under $min\_fitness$, a method hyper-parameter experimentally set to 5%. Afterwards, the selection of the individual that moves takes place, choosing one individual in the population with a probability that is proportional to its fitness value — the better the fitness, the higher the probability of being selected.

Individuals collectively adopt the most feasible strategy to improve their fitness function f. Controlled bursts into the neighbourhood eventually enable the individual to exploit its vicinity. During exploration, a random radius value (R) limits the jump lengths, defining the excursion of an individual through the domain.

The population's update receives a different number of generated individuals according to the chosen search strategy . During exploration – see Section 3.2.3 –, the number of generated individuals varies from 1 to $n\_neighbours$. The exploitation strategy generates a predefined $n\_neighbours$ number of neighbours, uniformly distributing C_x among them. Hence, the selected individual x with cardinality C_x is replaced with several individuals, although the total cardinality remains constant. The next two sections describe each of the strategies: exploitation and exploration, respectively.

The basic loop of the metaheuristic (Alg. 1) shows that the algorithm starts with the selected exploitation behaviour. It will perform that behaviour until it detects that the fitness has stagnated. Once a stagnation in fitness is detected, it will switch to the exploration behaviour until it detects stagnation again. Whenever better fitness values are no longer obtained, the behaviour will be changed to try to find better solutions with the other behaviour. A fitness improvement check is made with each iteration to detect stagnation of the fitness function. If many iterations are observed without improvement it will be considered stagnation. The number of iterations is defined by the parameter $wait\_iterations$. In terms of complexity, the algorithm runs linearly with the number of iterations times the number of individuals to generate in each step.

3.2.2. *Exploitation by bursting into the neighbourhood*

The exploitation behaviour thoroughly explores the neighbourhood of an individual x within the population by simulating mushroom bursts that generate new solutions in a hypersphere centred in x and a random radius r; this hypersphere radius r varies with the fitness function improvements and defines the vicinity. Eq. (1) shows the constraints for the current value of r according to the fitness value changes of the individual, where $max\_r \in [0.0, 1.0] \subset \Re$ is a hyperparameter and f_t refers to the fitness function of the individual at iteration t. It is worth remarking that the radius is a percentual value on the corresponding span for each dimension of the individual's vector; thus, a radius of 0.1 allows random wanderings in the vicinity of $0.1 \times (\max x_i - \min x_i)/2$, $\forall i : 1, \dots, N$.

$$0 < r \leq \min(max\_r, e^{2(\frac{|f_t - f_{t-1}|}{|f_{best} - f_0|} - 1)}) \tag{1}$$

Let U and V be two vectors of the same dimension as X; U holds randomly values uniformly sampled in $[0.0, 1.0] \subset \Re$, while V stores random binary values v_i, with $v_i \in \{0, 1\} \subset \aleph$, $\forall i : 1, \dots, N$.

A burst generates $n\_neighbours$ new individuals, each one following Eq. (2). In this equation, X is the individual to burst into its r-radius vicinity, producing X' (see Fig. 1). Let U and V be two vectors of the

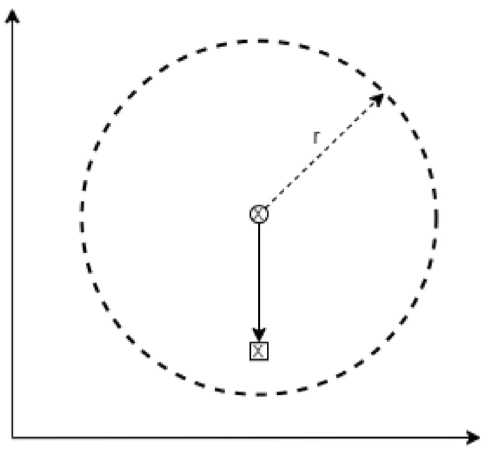


Fig. 1. Exploitation shift due to a burst in an r-radius vicinity. Assuming the same span for all the dimensions, in a 2-dimensional space the vicinity is a circle. If $X = [3.5, 3.0]$ and $r = 2.5$, and U and V happen to be $U = [0.3, 0.1]$ and $V = [0, 1]$, then $X' = [3.5, 1.0]$.

same dimension as X; U stores the shift to introduce in each dimension, storing random values uniformly sampled in $[0.0, 1.0] \subset \Re$. Besides, V marks which dimensions the burst introduces changes in the X — those dimensions for which V_i equals 1. To generate V, we first generate a zero-vector v of length N; the element at position i in v is set to 1 with probability $P_i(1) = \frac{1}{k!}$, where k is the number of ones already drawn in v. A final random rearrangement of the elements in v produces V, guaranteeing that every position has the same probability of storing a 1.

$$X' = X + V \times (U \times 2 \times r - r) \tag{2}$$

Algorithm 2 formalizes the exploitation operation, showing how Eq. (2) drives the individual generation. It is worth mentioning that this algorithm evaluates each single generated individual, uniformly reassigning the cardinality of individual X (C_X). Therefore, this algorithm is $O(n\_neighbours)$, with $n\_neighbours$ being the hyperparameter holding the number of individuals to generate in the vicinity during an exploitation step. As can be seen, the algorithm always generates n individuals regardless of the cardinality, which can lead to 3 different situations depending on the cardinality value. If $n\_neighbours > C_X$, then there will be individuals – the best ones –, with a higher cardinality. On the contrary, when $n\_neighbours < C_X$, there will be generated individuals that will be discarded because their cardinality will be 0 — these will be the worst ones. In case $n\_neighbours = C_X$, all the generated individuals will have cardinality 1.

Algorithm 2 Neighbourhood exploitation

```
Input
    X          Current individual
    n          $n\_neighbours$, number of individuals in the vicinity
    $max\_r$      maximum vicinity radius
Output
    new_sols   $n\_neighbours$ solutions in the vicinity of X
Update r using Eq. (1)
$new\_sols \leftarrow \{X\}$
for $i \in 1$ to $n$ do
    $sol \leftarrow$ generate a solution following Eq. (2)
    Evaluate $sol$
    Add $sol$ to $new\_sols$
end for
Sort $new\_sols$ according to the fitness
Discard the worst solution in $new\_sols$
Distribute $C_X$ uniformly among the $n$ individuals in $new\_sols$
```

3.2.3. Exploration of remotes areas

When food is scarce, the Slime Mould colony reunifies in a single body, migrating to different locations. This migration is achieved either by relocating the body along the environment or by creating a fruity form with spores that travel through the environment in the hope of reaching a promising allocation. In this study, Algorithm 3 adopts this second approach, allowing the spores to perform successive jumps among the domain in the hope of favourable regions in terms of fitness, but with the capacity to jump back in case of failure.

Algorithm 3 Exploring the domain

```
Input
    X        Current individual
    mR       $max\_R$, maximum expedition radius
    p_end    Probability of ending the scouting at this jump
    p_back   Probability of going backwards one jump in the sequence
Output
    X        A new and remote solution
$new\_sols \leftarrow \{X\}$
$X_{init} \leftarrow X$
$R \leftarrow random(mR)$
$path \leftarrow \{X\}$
do
    $X' \leftarrow$ generate a solution following Eq. (3)
    Add $X'$ to $new\_sols$
    $parent(X') \leftarrow X$
    Add $X'$ to $path$
    $X \leftarrow X'$
    while $random < p\_back$ and $X$ have parent do
        Drop $X$ from $path$
        $X \leftarrow parent(X)$
    end while
while $random < p\_end$
Evaluate all solutions in $new\_sols$
$X \leftarrow best(new\_sols)$
```

As mentioned before, a random radius value (R) defines the excursion of an individual through the domain; in this case, the max_R hyper-parameter controls the maximum length of an individual's jump ($0 < R \leq max\_R \leq 1.0$). As before, R is a percentual value that multiplies the corresponding span of each dimension of the individual's vector. This study proposes max_r and max_R to be equal to a value that has been experimentally tuned to 40% of the size of the search space, reducing the total number of metaheuristic's hyper-parameters.

A jump generates a new individual X' following Eq. (3). As for exploitation, X is the current individual, and the radius R defines its vicinity. Let U and V be two vectors of the same dimension as X; U holds randomly values uniformly sampled in $[0.0, 1.0] \subset \Re$, while V is a zeros vector that includes a single one at a random position: this position represents the jumping dimension.

$$X' = X + V \times (U \times 2 \times r - r) \tag{3}$$

An individual keeps jumping with probability p_end trying to find a better location in terms of fitness. The individual can also roll back from a jump with probability p_back. Interestingly, these jumps are blind operations as long as there is no fitness evaluation for these intermediate individuals; only the final one evaluates its performance. This design decision is a direct implementation from nature, as the fungus explores the remote domain even though intermediate positions become disappointing.

4. Experimentation set up

This research proposes a complete experimentation setup to demonstrate the effectiveness of the SMO. On the one hand, a comparison

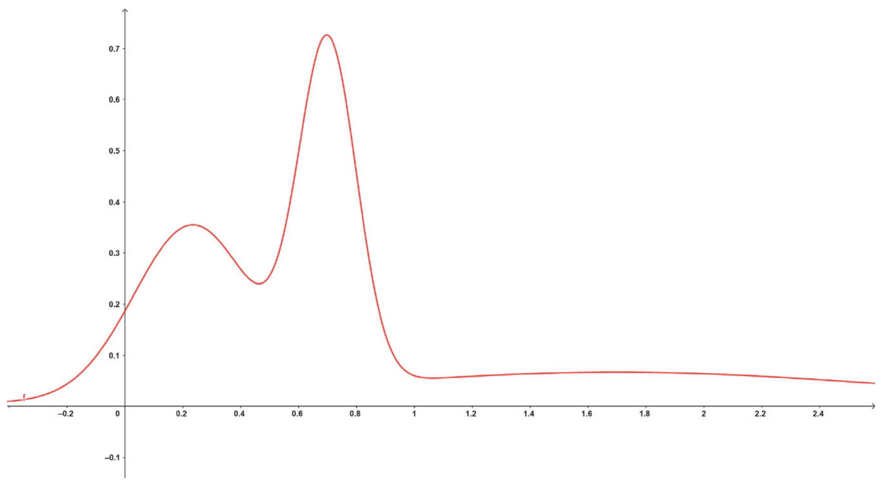


Fig. 2. Obtained hyperparameter distribution after running Iterated racing [46].

evaluates the most cited metaheuristics in recent years and the SMO on different well-known optimization problems, discussing how the hybridization of SMO and a trajectory-based metaheuristic enhances this latter's performance. On the other hand, a robot path planning problem examines how to use SMO in a real scenario. The following subsections give details on each of these experimentation stages. In the two experiments, the performance of the SMO is compared against the most recently cited metaheuristics: GWO [19], WOA [20], MFO [21], PSO [18], and DE [24]. This methodology for the evolution of metaheuristics was taken from a previous study published in [45]. The objective was to employ a neutral testing environment in which all metaheuristics compete under the same conditions.

4.1. A metaheuristics comparison on a standard set of problems

Fair comparisons are always required when proposing new solutions or methods; however, developing such an experiment is, by no means, an easy task. Two main issues must be addressed: the hyperparameter configuration of the methods and the benchmarking functions.

Concerning optimization methods, fair comparisons imply using the best hyperparameter set for each technique to compare while fixing the same computational resource limits for all of them. Indeed, manually guaranteeing the same opportunities in finding the best hyperparameter set for all the methods might become a human resource-consuming task that might introduce bias in the results. Instead, this study employs Iterate racing through the *irace package* [46], avoiding the biassing problem while automating the comparison as much as possible.

Iterated racing is an automatic configuration method that samples hyperparameter subsets according to a particular distribution, then selects the best subsets using racing, updating the sampling distributions accordingly. The process repeats until it meets a termination criterion; eventually, the method learns the hyperparameter distribution, producing a feasible hyperparameter suboptimal subset. Fig. 2 shows a possible final distribution of a hyperparameter, suggesting that 0.45 and 0.75 may become promising values when facing a new problem. Irace produces these peaks and their value as its outcome, allowing the user to reduce the search domain to their neighbourhood of these peaks.

In the hyperparameter selection process, the tool presented in [46] has been configured to search for hyperparameters of each of the metaheuristics included in the comparison. In this configuration, it has been established that the value the tool will use as a reference when searching for the best set of parameters will be the sum of the fitness functions of the 23 problems included in the comparison. This way, a configuration will be obtained that generally works well for all the problems. The specific details of this procedure can be found in the previous work that sets out how to compare [45].

This research adheres the experimentation criteria and design proposed in [47–49], which is very common in the literature. These studies select a set of functions serving not only as a testbed for the irace hyperparameter tuning but also as a benchmark for performance comparison; Table 1 describes the set of functions included in the benchmark for this research. In this way, the best hyper-parameter set is found for a metaheuristic in the benchmark first; then, the metaheuristic's performance is obtained for each function in the benchmark using this best hyper-parameter set. This experimentation design grants that all the metaheuristics make use of a hyper-parameter suboptimal set that works reasonably well for the majority of the problems.

Therefore, the comparison experimentation takes two stages: the hyperparameter irace tuning and the performance evaluation with the best hyperparameter subset. This latter part consists of 10 runs of each optimization method, with a total number of model evaluations set to 300,000. This number of model evaluations might seem huge; however, it is commonly used in the literature [50–53], ensuring enough computational resources, and allowing us to assess the speed of convergence of the different methods.

Two sets of metrics measure the quality assessment of each metaheuristic. On the one hand, the best, worst and average fitness in each evaluation; on the other hand, the iteration and total execution time. Collecting these metrics will produce two graph types – the quality of the solutions reached and the cost in execution time – that will depict the most relevant aspects of the performance of each metaheuristic.

The first graph type plots the evolution of the mean fitness value versus the number of evaluations, showing the performance of the metaheuristic with the number of individual appraisings before converging. In producing this plot, we register the mean fitness value obtained on each of the ten runs of a metaheuristic for each corresponding number of evaluations of individuals. Due to the high fluctuation of this fitness value, we smooth the plot using overlapping windows of 1000 values with a shift of 1 value.

The second graph type is a boxplot that draws a clear picture of the computational requirements of each metaheuristic. Registering the mean computational time to reach the total number of evaluations (300,000) for every single run of each metaheuristic produces ten values used to create the box plot for each function and metaheuristic. The execution time over the total number of evaluations is used instead of the converging time because there are benchmark functions for which some metaheuristics do not converge, difficulting the creation of the graph.

4.2. A practical comparison on robot path planning

This second experimentation stage aims to compare the metaheuristics on a real-world problem, obtaining a measure of their mathematical

Table 1
Standard set of functions used in the experimentation.

$F_1 = \sum_{i=1}^{n} x_i^2, \quad x_i \in [-100, 100] \quad \forall i : 1, \dots, n = 30.$
$F_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert,\ x_i \in [-10, 10]\ \forall i : 1, \dots, n,\ n = 30.$
$F_3 = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2,\ x_i \in [-100, 100]\ \forall i : 1, \dots, n,\ n = 30.$
$F_4 = \max_i \{x_i, 1 \leq i \leq n\}, \quad x_i \in [-100, 100] \quad \forall i : 1, \dots, n = 30.$
$F_5 = \sum_{i=1}^{n-1} [100(x_{x+1} - x_i^2)^2 + (x_i - 1)^2],\ x_i \in [-30, 30]\ \forall i : 1, \dots, n,\ n = 30.$
$F_6 = \sum_{i=1}^{n} (x_i + 0.5)^2,\ x_i \in [-100, 100]\ \forall i : 1, \dots, n,\ n = 30.$
$F_7 = \sum_{i=1}^{n} i x_i^4 + random[0, 1),\ x_i \in [-1.28, 1.28]\ \forall i : 1, \dots, n,\ n = 30.$
$F_8 = \sum_{i=1}^{n} (-x_i \sin \sqrt{\lvert x_i \rvert}), \quad x_i \in [-500, 500] \quad \forall i : 1, \dots, n = 30.$
$F_9 = \sum_{i=1}^{n} [x_i^2 - 10 \cos 2\pi x_i + 10],\ x_i \in [-5.12, 5.12]\ \forall i : 1, \dots, n,\ n = 30.$
$F_{10} = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=0}^{n} x_i^2}) - \exp(\frac{1}{n} \sum_{i=0}^{n} \cos 2\pi x_i) + 20 - e,\ x_i \in [-32, 32]\ \forall i : 1, \dots, n,\ n = 30.$
$F_{11} = \frac{1}{4000} \sum_{i=0}^{n} x_i^2 - \prod_{i=1}^{n} \frac{x_i}{\sqrt{i}} + 1, \quad x_i \in [-600, 600] \quad \forall i : 1, \dots, n = 30.$
$F_{12} = \frac{\pi}{n} \{10 \sin \pi y_1 + \sum_{i=1}^{n} [(y_i)^2[1 + 10 \sin \pi {y_{i+1}}^2]] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),\ x_i \in [-50, 50]\ \forall i : 1, \dots, n,\ n = 30.$
Where $y_i = 1 + \frac{x_i+1}{4}$, and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i \geq a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x \leq -a \end{cases}$
$F_{13} = 0.1\{\sin 3\pi {x_1}^2 + \sum_{i=1}^{n} [(x_i - 1)^2[1 + \sin 3\pi x_i + 1^2]]\}+,\ x_i \in [-50, 50]\ \forall i : 1, \dots, n,\ n = 30.$
Where u is defined in the same way like F_{12}
$F_{14} = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6})^{-1}, \quad x_i \in [-65, 65] \quad \forall i : 1, \dots, n = 2.$
$F_{15} = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2,\ x_i \in [-5, 5]\ \forall i : 1, \dots, n,\ n = 4.$
Where: $a = [.1957, .1947, .1735, .16, .0844, .0627, .0456, .0342, .0323, .0235, .0246]$
and $b = [4, 2, 1, .5, .25, \frac{1}{6}, .125, -1, \frac{1}{12}, \frac{1}{14}, .0625]$
$F_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3} x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4,\ x_i \in [-5, 5]\ \forall i : 1, \dots, n,\ n = 2.$
$F_{17} = (x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6)^2 + 10(1 - -\frac{1}{8\pi}) \cos x_1 + 10, \quad x_i \in [-5, 5] \quad \forall i : 1, \dots, n = 2.$
$F_{18} = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)] \cdot [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)],$ $x_i \in [-2, 2]\ \forall i : 1, \dots, n,\ n = 2.$
$F_{19} = \sum_{i=1}^{4} c_i \exp - \sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2,\ x_i \in [1, 3]\ \forall i : 1, \dots, n,\ n = 3.$
Where: $a = \begin{bmatrix} 10 & .05 & 3 & 17 \\ 3 & 10 & 3.5 & 8 \\ 17 & 17 & 1.7 & .05 \\ 3.5 & .1 & 10 & 10 \\ 1.7 & 8 & 17 & .1 \\ 8 & 14 & 8 & 14 \end{bmatrix}$, $c = \begin{bmatrix} 1 & 1.2 & 3 & 3.2 \end{bmatrix}$,
and $p = \begin{bmatrix} .1312 & .2329 & .2348 & .4047 \\ .1696 & .4135 & .1415 & .8828 \\ .5569 & .8307 & .3522 & .8732 \\ .0124 & .3736 & .2883 & .5743 \\ .8283 & .1004 & .3047 & .1091 \\ .5886 & .9991 & .6650 & .0381 \end{bmatrix}$
$F_{20} = \sum_{i=1}^{4} c_i \exp - \sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2,\ x_i \in [0, 1]\ \forall i : 1, \dots, n,\ n = 6.$
Where a, c and p, are the defined identical as F_{19}
$F_{21} = -\sum_{i=0}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}, \quad x_i \in [0, 10] \quad \forall i : 1, \dots, n = 4.$
Where: $a = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{bmatrix}$,
and $c = \begin{bmatrix} .1 & .2 & .2 & .4 & .4 & .6 & .3 & .7 & .5 & .5 \end{bmatrix}$
$F_{22} = -\sum_{i=0}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1},\ x_i \in [0, 10]\ \forall i : 1, \dots, n,\ n = 4.$
Where a, c and p, are the defined identical as F_{21}
$F_{23} = -\sum_{i=0}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1},\ x_i \in [0, 10]\ \forall i : 1, \dots, n,\ n = 4.$
Where a, c and p, are the defined identical as F_{21}

capacity and practical results. Robot path planning in an environment with random obstacles is the selected real-world challenge due to its complexity and applicability [54]. Interestingly, the selected hyper-parameters' values will be the same extracted values obtained with irace for the metaheuristics comparison on standard problems, as those would be the most promising values.

Given a robot whose position is always known, a space through which the robot can move freely, a set of known obstacles in the environment, and the destination coordinates to which the robot must arrive, the problem consists of calculating the best route from the robot's current position to the destination coordinates avoiding collisions with the obstacles. The route length and the number of collisions represent the quality metrics for comparing the results.

An ordered sequence of coordinates represents a solution or individual in an optimization problem; each consecutive pair of coordinates delimits a straight segment of the route. Although the sequence does not include the starting robot position, this position and the sequence's first coordinates pair determine the first route's segment. With this representation, registering the intersections of each segment with the existing obstacles defines the number of detected collisions of a route. Additionally, the sum of the Euclidean distances between consecutive

Table 2
Centre position (c) and radius (r) for each of the obstacles included in the 3 scenarios used in this experimentation [54].

	Scenario 1	Scenario 2	Scenario 3
1	c:(300,150) r:75	c:(250,50) r:150	c:(200,200) r:100
2	c:(250,600) r:100	c:(380,420) r:150	c:(380,500) r:150
3	c:(600,100) r:100	c:(550,100) r:150	c:(550,150) r:125
4	c:(500,750) r:100	c:(250,750) r:200	c:(250,750) r:150
5	c:(850,550) r:75	c:(700,500) r:100	c:(750,450) r:100
6	c:(450,300) r:75	c:(850,250) r:200	c:(800,250) r:150
7	c:(750,350) r:50	c:(450,700) r:150	c:(500,700) r:150
8	c:(700,800) r:75	c:(750,800) r:150	c:(800,800) r:100

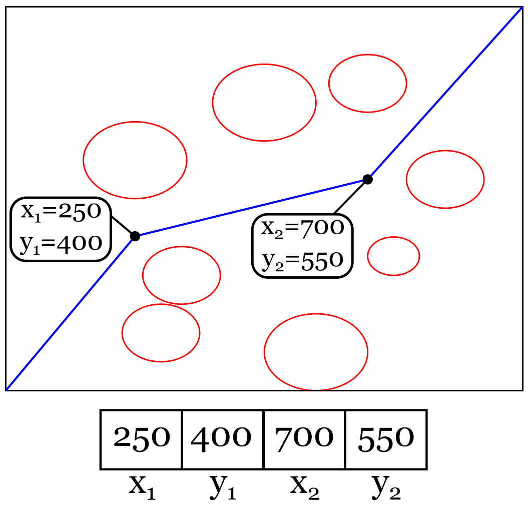


Fig. 3. An example of a solution using the representation of a path planning problem in this experimentation.

coordinate pairs of the route, including the robot's starting point, represents the path's length. Fig. 3 provides a graphical representation of this explanation, showing how the solution is constructed with a sequence of points represented by a number vector. Thanks to this representation, any metaheuristic can tackle the problem without the need to adapt it to the problem.

As stated in [54], the robot path planning challenge defines three 2D scenarios where obstacles are random rounded objects with a centre position and a radius: refer to Table 2 for a complete list of obstacles from [54], and to Fig. 4 for a graphical representation of the scenarios.

5. Results and discussion

This section splits into two main parts: the results and discussion on comparing the metaheuristics on standard optimization problems (Section 5.1), and the performance of the metaheuristics on the real-world problem of robot path planning (Section 5.2). As explained before, these results compares the performance of SMO against GWO [19], WOA [20], MFO [21], PSO [18], and DE [24].

5.1. Comparison of the metaheuristics on standard problems

As explained before, hyperparameter optimization represents the first step to accomplish; the irace package performs this stage on the complete set of standard functions. Table 3 shows the best hyperparameter set found for each metaheuristic in this comparison; this study uses these values throughout the experimentation.

Results for the metaheuristic comparison on standard functions are depicted in Fig. 5 with the evolution of the fitness, Fig. 6 with the box plots of the execution time of each metaheuristic, and Table 4 with the best mean fitness values among the 10 repetitions for each case.

Fig. 5 shows the smoothed evolution of the mean fitness value versus the number of individual evaluations, with a total number of fitness function calls of 300,000. Each subgraph shows the behaviour of every metaheuristic on a specific benchmark function; it does not represent the evolution of the fitness for one concrete run of the metaheuristics but the aggregated performance. Hence, we can study the general behaviour of each metaheuristic: Does a metaheuristic generally reach an optimum? Does it tend to converge? Is the metaheuristic fast or slow in converging?

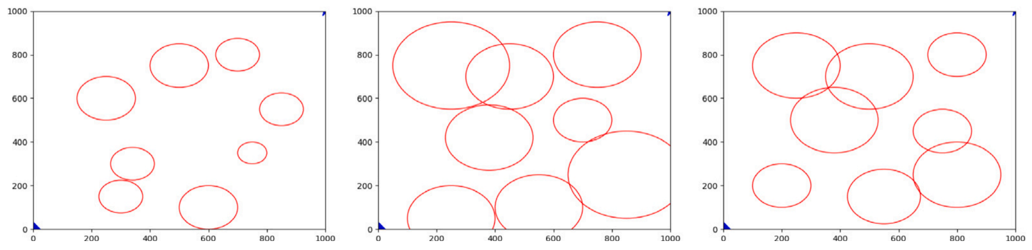
In general, the metaheuristic with the best behaviour varies with the nature of the problem, either according to the number of suboptimal points in the domain, the problem's depth and width around the optimal value, or the variability of the values in the domain. For instance, DE performs remarkably well on multimodal functions ($F8$, $F12$, $F14$), while GWO does the same on unimodal functions ($F1$ to $F7$). Interestingly, SMO behaves voracious at the beginning, finding interesting search areas. Although sometimes it converges to reasonable suboptimal values ($F7$ or $F8$), it seems able to optimize problems where the other metaheuristics fail, as for the $F19$ and $F20$ functions.

However, SMO pays for its voracity, penalizing the quality of its solutions in many cases as SMO gets stuck at a local minimum and fails to reach the optimal solution. Moreover, the switching between the two behaviours seems to fail in detecting the stagnation. These two ideas would drive future improvements to the SMO.

Table 4 shows the best mean fitness values found over the ten repetitions of the optimization process for each metaheuristic. In this aspect, WOA is the metaheuristic that reaches the best solution in a more significant number of problems, with 16 out of the 23 functions in the benchmark. SMO achieves the best solution for 8 of the 23 functions, mainly for its voracity and inability to get out of local minima. It is worth mentioning that DE reaches the best solution in 10 out of 23 problems.

Now, let us pay attention to the running times in the box plots shown in Fig. 6; each box plot represents the execution time required by each metaheuristic to finish a concrete optimization problem. As can be seen, the SMO is the faster method in converging due to its voracity, with DE closely following for the major part of the cases. WOA is not the slowest method but is far apart from the performance of SMO or DE. It seems that optimality is at odds with convergence speed.

Considering, on the one hand, the good performance of WOA in terms of fitness function but its problems in time consumption, and, on the other hand, that SMO is not so bad in terms of fitness but is by far the faster, one question that arises is if it is possible to accelerate WOA by hybridizing it with SMO. The hybridization was proposed after an analysis of some preliminary results and it is expected to achieve a method that can acquire the speed characteristics of the most voracious algorithm of the proposed ones, which is SMO, and the quality characteristics of the most accurate algorithm of the evaluated ones, which is WOA. We want to analyse if this hybridization achieves to build a method almost as fast as SMO and almost as accurate as WOA in reaching the best conclusion. A simple hybridization scheme is studied in this research, including the results in Table 4 and Figs. 5 and 6 with the label SMO+WOA.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 4. Graphical representation of the scenarios in the robot path planning problem.

Table 3
Best values for each hyperparameter of the metaheuristics after the irace process.

Parameter name	DE	GWO	PSO	MFO	WOA	SMO
Population size	210	180	180	190	190	200
Crossover prob.	0,78					
Mutation prob.	0,21					
c1			1,86			
c2			2,18			
b				1,68	1,72	
End explore prob.						0,80
Backwards prob.						0,50
R_max						0,40
n_neighbours						1000
wait_iterations						100

Table 4
Best mean fitness value among the 10 repetitions of the optimization problem for each metaheuristic. The results for the hybridized method SMO+WOA are also included. F stands for the function acronym.

F	DE	GWO	PSO	MFO	SMO	WOA	SMO+WOA
F1	8,9788E−58	1,7808E−175	6,6240E−24	1,0000E+03	1,3526E−02	**0,0000E+00**	**0,0000E+00**
F2	1,2385E−33	1,4025E−99	1,0000E+00	2,9000E+01	5,3245E−02	2,7070E−186	**0,0000E+00**
F3	4,2378E+00	**2,2005E−60**	1,2583E−01	1,6167E+04	6,8775E−01	1,7565E+02	7,9695E−02
F4	6,1912E−01	**2,8705E−43**	4,9054E−02	4,8742E+00	1,9498E−01	8,6626E+00	9,6815E−02
F5	2,8511E+01	2,5341E+01	4,8678E+01	1,8077E+04	2,8161E+01	**1,9263E+01**	2,5794E+01
F6	**0,0000E+00**	1,0012E−01	5,1912E−25	2,0001E+03	2,8563E−02	1,4594E−06	2,4908E−04
F7	2,1339E−07	1,6311E−297	1,3422E+00	2,6844E−01	8,3296E−08	**0,0000E+00**	**0,0000E+00**
F8	−1,1096E+04	−6,6669E+03	−7,0575E+03	−8,4930E+03	−1,0338E+04	**−1,2421E+04**	−1,0480E+04
F9	6,7204E+01	1,2037E+00	5,5980E+01	1,0932E+02	4,5338E−01	**0,0000E+00**	**0,0000E+00**
F10	4,3521E−15	7,5495E−15	2,6477E−13	1,5960E+01	4,2362E−02	3,2863E−15	**2,5757E−10**
F11	**0,0000E+00**	2,7756E−20	2,7756E−20	1,8000E+01	5,7510E−05	5,5511E−21	2,7756E−21
F12	**−1,0472E+00**	−1,0407E+00	**−1,0472E+00**	**−1,0472E+00**	−1,0471E+00	**−1,0472E+00**	−1,0469E+00
F13	**1,3498E−32**	1,5626E−01	9,6305E−25	1,1305E−01	2,9924E−03	7,2768E−06	4,5749E−04
F14	**9,9800E−01**	**9,9800E−01**	**9,9800E−01**	**9,9800E−01**	**9,9800E−01**	**9,9800E−01**	**9,9800E−01**
F15	1,9926E−02	**1,7811E−02**	2,0012E−02	**1,7811E−02**	1,9432E−02	**1,7811E−02**	**1,7811E−02**
F16	**0,0000E+00**	**0,0000E+00**	1,0192E−205	**0,0000E+00**	4,2395E−09	**0,0000E+00**	**0,0000E+00**
F17	**3,9789E−01**	**3,9789E−01**	**3,9789E−01**	**3,9789E−01**	**3,9789E−01**	**3,9789E−01**	**3,9789E−01**
F18	**3,0000E+00**	**3,0000E+00**	**3,0000E+00**	**3,0000E+00**	**3,0000E+00**	**3,0000E+00**	**3,0000E+00**
F19	**−3,0048E−01**	**−3,0048E−01**	**−3,0048E−01**	**−3,0048E−01**	**−3,0048E−01**	**−3,0048E−01**	**−3,0048E−01**
F20	**−3,4085E−05**	**−3,4085E−05**	**−3,4085E−05**	**−3,4085E−05**	**−3,4085E−05**	**−3,4085E−05**	**−3,4085E−05**
F21	−9,6019E+00	−9,0995E+00	−7,5890E+00	−7,5774E+00	**−1,0104E+01**	**−1,0104E+01**	**−1,0104E+01**
F22	−9,6019E+00	−8,5973E+00	−8,5896E+00	−8,0835E+00	**−1,0104E+01**	**−1,0104E+01**	**−1,0104E+01**
F23	−1,0028E+01	−8,5973E+00	−7,0829E+00	−7,3365E+00	**−1,0104E+01**	**−1,0104E+01**	**−1,0104E+01**

The hybridization always starts with SMO, swapping to WOA when registering three consecutive changes in the SMO behaviour – from exploitation to exploration and vice versa – without improvement. In other words, we start with SMO to converge to a reasonable solution quickly; once the stagnation is detected, we switch to WOA for a more intense search, starting at a more promising position. Transitioning from SMO to WOA follows a simple scheme: the SMO individuals with a cardinality higher or equal to 1 become the initial population in WOA.

Results are promising (see Table 4) as the hybridization reaches the optimum value in 15 out of 23 functions — one less than WOA, but seven functions more than SMO. As can be extracted from Fig. 5, the SMO assists WOA, so it starts from a promising point and reaches the optimum value (for instance, refer to the evolutions for $F2$, $F3$, $F4$). Even though the convergence of SMO+WOA occurs after more evaluations than WOA for some cases (i.e., for $F5$, $F9$, or $F13$), the time consumed in the optimization is significative smaller. Indeed, as shown in Fig. 6), the SMO+WOA becomes the second or third faster method – according to the problem –, closely following the performance of SMO. This fact suggests that SMO is helping WOA in finding an optimal solution by locating intermediate promising solutions, accelerating the per se slower WOA method.

5.2. Performance of the metaheuristics in robot path planning

This experimentation on robot path planning uses the same collection of hyperparameters found with irace for the previous Subsection; as long as these values represent the best compromise hyperparameters for each metaheuristic, the comparison in the performance remains fair.

Results are depicted in Table 5: the upper part shows the length of the best path found by the different metaheuristics that participated in

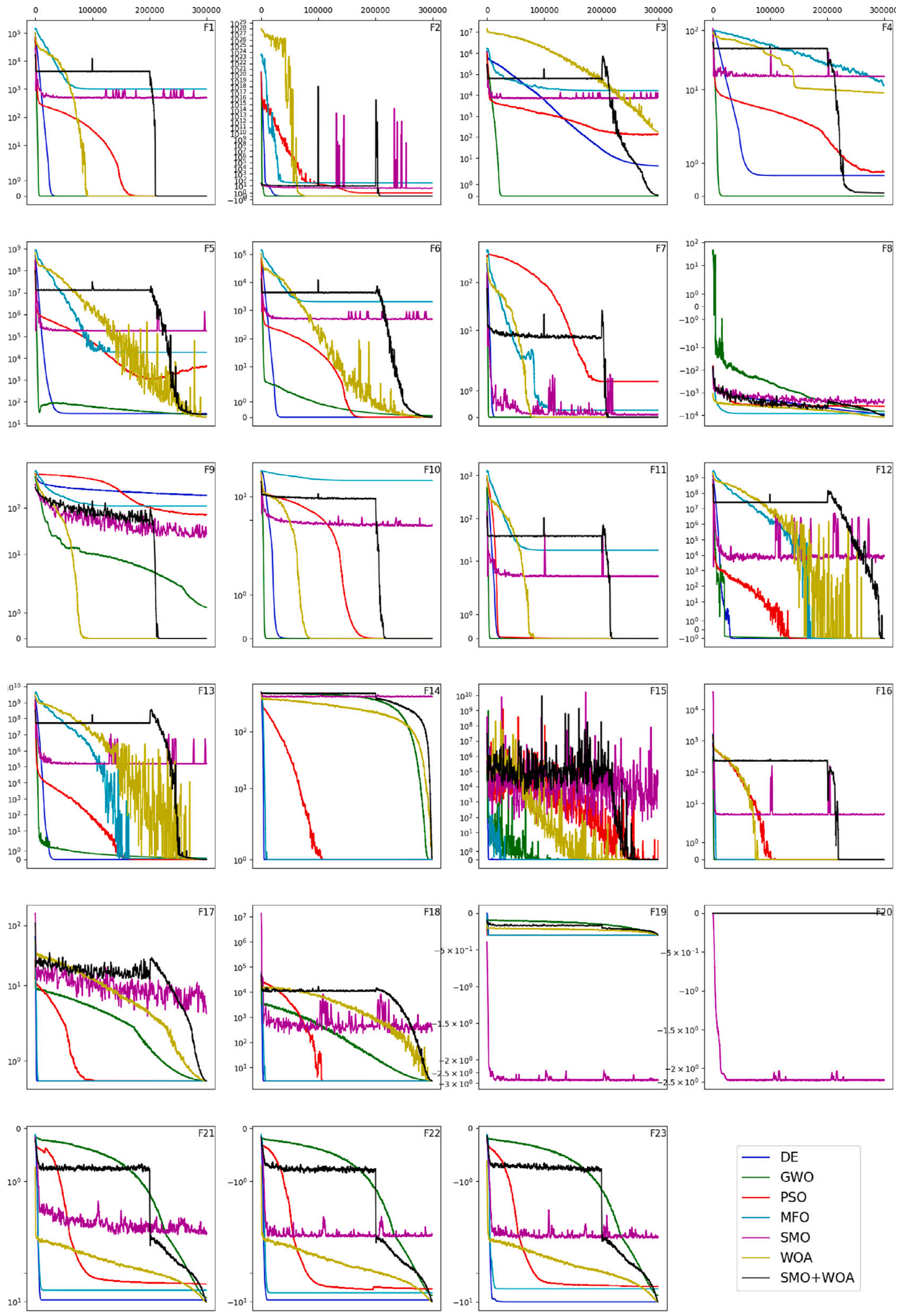


Fig. 5. The smoothed evolution of the mean fitness value versus the number of evaluations for each metaheuristic and problem, with a total number of evaluations of 300,000.

the comparison, while the lower part includes the execution time for each metaheuristic and scenario. Furthermore, Fig. 7 shows the mean and standard deviation of the path lengths versus the execution time for each case.

SMO obtained the best solution in the three scenarios; in all the cases, the variability in the results was either small or negligible. Although SMO did not stand out for reaching perfect fitness solutions in mathematical problems, SMO represents a competitive alternative to choose for robot path planning.

Besides, DE performed similarly in the three scenarios when considering the running time, with a small standard deviation value for all of them; however, the path lengths were far from optimum. On the

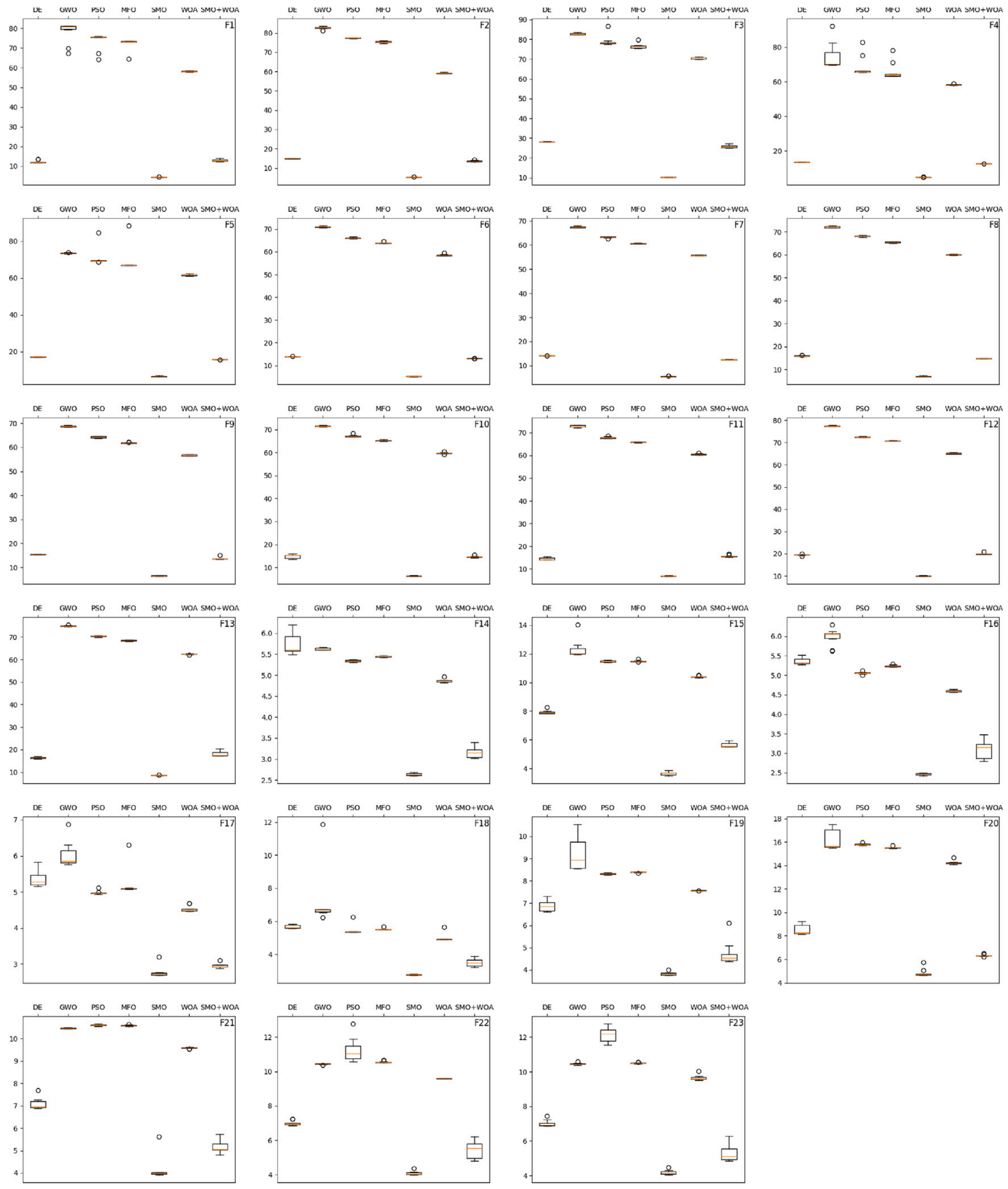


Fig. 6. Execution time of each metaheuristic for each problem in seconds.

other hand, SMO's running times were similar to those of the remaining metaheuristics.

6. Conclusion

This research proposed a new metaheuristic inspired by the slime mould and its foraging behaviours. On the one hand, an exploitation stage mimics the greedy amoeba's conduct when food is plenty. On the other hand, an exploration stage copies the fruity aggregation of the cells and the subsequent spore dissemination. The obtained metaheuristic shows a voracious manner that finds reasonable solutions in a reduced time.

The study compared the most cited metaheuristics and the SMO in two different experimentation stages: on the one hand, the optimization of standard benchmarking functions; on the other hand, solving the robot path planning problem. Moreover, a hybridization of the SMO and the WOA is also presented, which keeps the SMO's convergence speed and the WOA's good performance in finding the best solutions.

Besides, SMO found the best solutions in two of the three scenarios designed for robot path planning, and the second best in the worst case. Concerning the running time, the SMO performed similarly to the majority of the compared methods; however, DE is undoubtedly the faster method in this real-world problem. The SMO algorithm fails in two main things: getting stuck in local minima and switching between

Table 5
Path lengths (upper part) and running time (lower part) for each metaheuristic and scenario. MN and SD stand for the mean and standard deviation of the shortest path length from each metaheuristic among the ten runs.

Path length						
Method	Scenario 1		Scenario 2		Scenario 3	
	MN	SD	MN	SD	MN	SD
DE	1584.33	99.19	1822.34	0.60	1815.59	99.54
GWO	1536.60	194.84	1892.04	148.11	1786.45	23.36
PSO	1440.03	58.46	1925.78	407.86	2260.02	854.47
MFO	1648.74	200.77	1899.69	74.06	1793.54	31.85
WOA	1570.95	125.82	1888.54	23.81	1885.78	53.65
SMO	**1423.11**	1.22	**1597.95**	68.59	**1460.60**	4.22

Running time						
Method	Scenario 1		Scenario 2		Scenario 3	
	MN	SD	MN	SD	MN	SD
DE	**13.07**	0.58	**12.83**	1.39	**13.06**	0.88
GWO	18.30	1.90	18.21	2.72	17.54	2.20
PSO	17.88	0.45	19.73	0.83	20.59	3.39
MFO	16.04	1.29	15.86	0.78	16.20	0.66
WOA	14.31	0.70	16.48	1.40	15.29	1.39
SMO	17.13	1.49	19.39	0.80	15.24	1.55

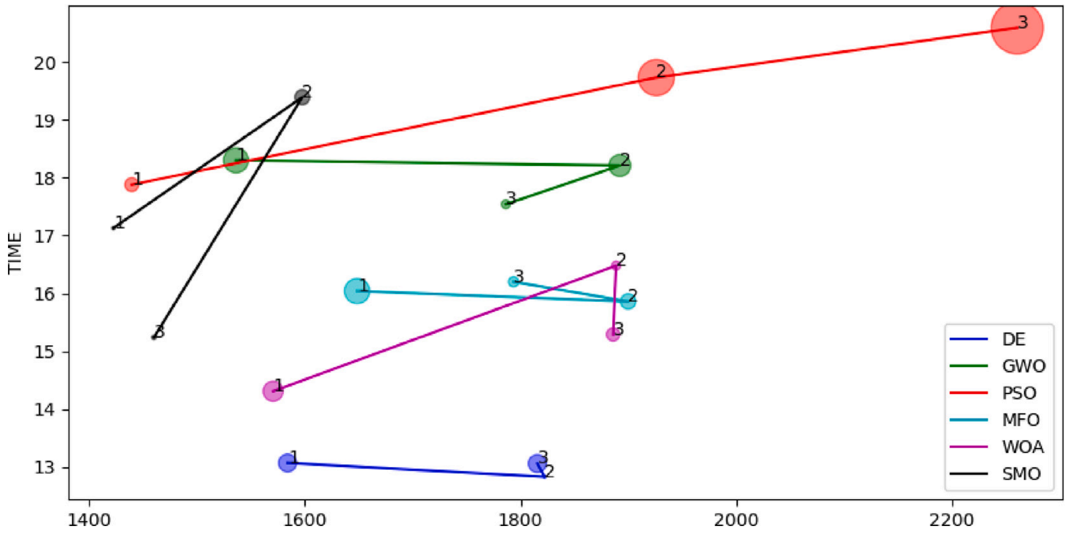


Fig. 7. Path length quality. Colours are specific for each metaheuristic. Each point has the corresponding scenario number as a label. The *x*-axis registers the path lengths, while the *y*-axis plots the execution time. The size of the points is the corresponding path length's standard deviation. Lines join the results from the same metaheuristic.

behaviours. While the former can be addressed using different exploitation or exploration schemes, perhaps randomly biasing one or another to avoid promising directions, the latter issue is more challenging to measure and decide on.

Interestingly, SMO can be easily adapted to different real world problems, such as logistics. The main issue would be to represent the problem as a vector, similar as it has been done for this experimentation. Introducing some types of constraints would require an in-depth analysis, though, but it certainly does not represent a limitation to the metaheuristic. This is part of future work, where we are applying SMO to a logistic problem in a factory.

Future research also includes the application of these metaheuristics to different industrial problems, such as resource assignment or robot scheduling. Moreover, extending the SMO to deal with multiobjective optimization problems and its hybridization capabilities shall concentrate the research as well. Also, future research will include a deeper analysis of the performance of this metaheuristic, especially in high dimensionality environments with a large number of constraints.

CRediT authorship contribution statement

Enol García: Writing – original draft, Software, Project administration, Investigation, Formal analysis. **José R. Villar:** Writing – original draft, Validation, Supervision, Project administration, Investigation, Funding acquisition, Conceptualization. **Javier Sedano:** Writing – review & editing, Validation, Funding acquisition. **Camelia Chira:** Writing – review & editing, Validation, Supervision. **Enrique de la Cal:** Writing – review & editing, Validation, Supervision. **Luciano Sánchez:** Writing – review & editing, Validation, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Jose R. Villar reports financial support was provided by Spanish Ministry of Economics and Industry. Jose R. Villar reports financial support was provided by Spanish Research Agency. Javier Sedano reports financial support was provided by Missions Science and Innovation project. Jose R. Villar reports financial support was provided by Principado de Asturias. Jose R. Villar reports financial support was provided by Institute of Industrial Technology of Asturias. Jose R. Villar reports financial support was provided by Fundación Universidad de Oviedo. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research has been funded by the Spanish Ministry of Economics and Industry – grant PID2020-112726RB-I00 –, the Spanish Research Agency – grant PID2023-146257OB-I00 –, and Missions Science and Innovation project MIG-20211008 (INMERBOT). Also, by Principado de Asturias, grant SV-PA-21-AYUD/2021/50994, and by the Council of Gijón through the University Institute of Industrial Technology of Asturias grants SV-21-GIJON-1-19, SV-22-GIJON-1-19, SV-22-GIJON-1-22, SV-23-GIJON-1-09, and SV-23-GIJON-1-17. Finally, this research has also been funded by Fundación Universidad de Oviedo grants FUO-23-008 and FUO-22-450. This research is also supported by the project "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 334906.

Data availability

No data was used for the research described in the article.

References

[1] M. Riaz, M. Bashir, I. Younas, Metaheuristics based COVID-19 detection using medical images: A review, Comput. Biol. Med. 144 (2022) 105344, http://dx.doi.org/10.1016/j.compbiomed.2022.105344, URL https://www.sciencedirect.com/science/article/pii/S0010482522001366.

[2] S. AlMuhaideb, M.E.B. Menai, A new hybrid metaheuristic for medical data classification, Int. J. Metaheuristics 3 (1) (2014) 59–80, http://dx.doi.org/10.1504/IJMHEUR.2014.058860.

[3] A.C.B. Monteiro, R.P. França, V.V. Estrela, N. Razmjooy, Y. Iano, P.D.M. Negrete, Metaheuristics applied to blood image analysis, in: N. Razmjooy, M. Ashourian, Z. Foroozandeh (Eds.), Metaheuristics and Optimization in Computer and Electrical Engineering, Springer International Publishing, Cham, 2021, pp. 117–135, http://dx.doi.org/10.1007/978-3-030-56689-0_6.

[4] D. Kumar, B.G.R. Gandhi, R.K. Bhattacharjya, Firefly algorithm and its applications in engineering optimization, in: F. Bennis, R.K. Bhattacharjya (Eds.), Nature-Inspired Methods for Metaheuristics Optimization: Algorithms and Applications in Science and Engineering, Springer International Publishing, Cham, 2020, pp. 93–103, http://dx.doi.org/10.1007/978-3-030-26458-1_6.

[5] N. Xiong, D. Molina, M.L. Ortiz, F. Herrera, A walk into metaheuristics for engineering optimization: Principles, methods and recent trends, Int. J. Comput. Intell. Syst. 8 (4) (2015) 606–636, http://dx.doi.org/10.1080/18756891.2015.1046324.

[6] S.K. Gupta, M. Ramteke, Applications of genetic algorithms in chemical engineering II: Case studies, in: J. Valadi, P. Siarry (Eds.), Applications of Metaheuristics in Process Engineering, Springer International Publishing, Cham, 2014, pp. 61–87, http://dx.doi.org/10.1007/978-3-319-06508-3_3.

[7] A. Soler-Dominguez, A.A. Juan, R. Kizys, A survey on financial applications of metaheuristics, ACM Comput. Surv. 50 (1) (2017) http://dx.doi.org/10.1145/3054133.

[8] M.A. Rahman, Quantile regression using metaheuristic algorithms, Int. J. Comput. Econ. Econ. 3 (3–4) (2013) 205–233, http://dx.doi.org/10.1504/IJCEE.2013.058498.

[9] M.-M. Memmah, F. Lescourret, X. Yao, C. Lavigne, Metaheuristics for agricultural land use optimization. A review, Agron. Sustain. Dev. 35 (3) (2015) 975–998, http://dx.doi.org/10.1007/s13593-015-0303-4.

[10] A. Adamatzky, Slime mold solves maze in one pass, assisted by gradient of chemo-attractants, IEEE Trans. NanoBioscience 11 (2) (2012) 131–134, http://dx.doi.org/10.1109/TNB.2011.2181978.

[11] C.R. Reid, T. Latty, A. Dussutour, M. Beekman, Slime mold uses an externalized spatial "memory" to navigate in complex environments, Proc. Natl. Acad. Sci. 109 (43) (2012) 17490–17494, http://dx.doi.org/10.1073/pnas.1215037109, URL https://www.pnas.org/doi/abs/10.1073/pnas.1215037109.

[12] J. Smith-Ferguson, C.R. Reid, T. Latty, M. Beekman, Hänsel, Gretel and the slime mould—how an external spatial memory aids navigation in complex environments, J. Phys. D: Appl. Phys. 50 (41) (2017) 414003, http://dx.doi.org/10.1088/1361-6463/aa87df.

[13] I. Fister, X.-S. Yang, I. Fister, J. Brest, D. Fister, A brief review of nature-inspired algorithms for optimization, 2013, http://dx.doi.org/10.48550/arXiv.1307.4186, arXiv:1307.4186.

[14] M. Gendreau, J.-Y. Potvin, Metaheuristics in combinatorial optimization, Ann. Oper. Res. 140 (1) (2005) 189–213, http://dx.doi.org/10.1007/s10479-005-3971-7.

[15] I.H. Osman, Focused issue on applied meta-heuristics, Comput. Ind. Eng. 44 (2) (2003) 205–207, http://dx.doi.org/10.1016/S0360-8352(02)00175-4.

[16] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308, http://dx.doi.org/10.1145/937503.937505.

[17] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, http://dx.doi.org/10.1109/4235.585893.

[18] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.

[19] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.

[20] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67, http://dx.doi.org/10.1016/j.advengsoft.2016.01.008.

[21] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249, http://dx.doi.org/10.1016/j.knosys.2015.07.006.

[22] M. Azizi, S. Talatahari, A.H. Gandomi, Fire Hawk optimizer: a novel metaheuristic algorithm, Artif. Intell. Rev. 56 (1) (2023) 287–363, http://dx.doi.org/10.1007/s10462-022-10173-w.

[23] J.-S. Chou, D.-N. Truong, A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean, Appl. Math. Comput. 389 (2021) 125535, http://dx.doi.org/10.1016/j.amc.2020.125535, URL https://www.sciencedirect.com/science/article/pii/S0096300320304914.

[24] R. Storn, K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.

[25] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680, http://dx.doi.org/10.1126/science.220.4598.671.

[26] A.Y.S. Lam, V.O.K. Li, Chemical-reaction-inspired metaheuristic for optimization, IEEE Trans. Evol. Comput. 14 (3) (2010) 381–399, http://dx.doi.org/10.1109/TEVC.2009.2033580.

[27] M. Azizi, U. Aickelin, H.A. Khorshidi, M. Baghalzadeh Shishehgarkhaneh, Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization, Sci. Rep. 13 (1) (2023) 226, http://dx.doi.org/10.1038/s41598-022-27344-y.

[28] M. Azizi, Atomic orbital search: A novel metaheuristic algorithm, Appl. Math. Model. 93 (2021) 657–683, http://dx.doi.org/10.1016/j.apm.2020.12.021, URL https://www.sciencedirect.com/science/article/pii/S0307904X20307198.

[29] Z.W. Geem, J.K. Kim, G.V. Loganathan, A new heuristic optimization algorithm: Harmony search, Simulation (2001) http://dx.doi.org/10.1177/003754970107600201.

[30] J.H. Kim, Harmony search algorithm: A unique music-inspired algorithm, Procedia Eng. 154 (2016) 1401–1405, http://dx.doi.org/10.1016/j.proeng.2016.07.510, 12th International Conference on Hydroinformatics (HIC 2016) - Smart Water for the Future.

[31] S. Mirjalili, SCA: A Sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133, http://dx.doi.org/10.1016/j.knosys.2015.12.022, URL https://www.sciencedirect.com/science/article/pii/S0950705115005043.

[32] J.C. Bansal, P. Bajpai, A. Rawat, A.K. Nagar, Sine cosine algorithm, in: Sine Cosine Algorithm for Optimization, Springer Nature Singapore, Singapore, 2023, pp. 15–33, http://dx.doi.org/10.1007/978-981-19-9722-8_2.

[33] S. Talatahari, M. Azizi, Chaos game optimization: a novel metaheuristic algorithm, Artif. Intell. Rev. 54 (2) (2021) 917–1004, http://dx.doi.org/10.1007/s10462-020-09867-w.

[34] M. Azizi, M. Baghalzadeh Shishehgarkhaneh, M. Basiri, R.C. Moehler, Squid game optimizer (SGO): a novel metaheuristic algorithm, Sci. Rep. 13 (1) (2023) 5373, http://dx.doi.org/10.1038/s41598-023-32465-z.

[35] E. Bogar, S. Beyhan, Adolescent identity search algorithm (AISA): A novel metaheuristic approach for solving optimization problems, Appl. Soft Comput. 95 (2020) 106503, http://dx.doi.org/10.1016/j.asoc.2020.106503, URL https://www.sciencedirect.com/science/article/pii/S1568494620304427.

[36] T.A. Feo, M.G. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Oper. Res. Lett. 8 (2) (1989) 67–71, http://dx.doi.org/10.1016/0167-6377(89)90002-3.

[37] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (5) (1986) 533–549, http://dx.doi.org/10.1016/0305-0548(86)90048-1, URL https://www.sciencedirect.com/science/article/pii/0305054886900481. Applications of Integer Programming.

[38] W. Wong, C.I. Ming, A review on metaheuristic algorithms: Recent trends, benchmarking and applications, in: 2019 7th International Conference on Smart Computing & Communications, ICSCC, 2019, pp. 1–5, http://dx.doi.org/10.1109/ICSCC.2019.8843624.

[39] A.W. Mohamed, K.M. Sallam, P. Agrawal, A.A. Hadi, A.K. Mohamed, Evaluating the performance of meta-heuristic algorithms on CEC 2021 benchmark problems, Neural Comput. Appl. 35 (2) (2023) 1493–1517, http://dx.doi.org/10.1007/s00521-022-07788-z.

[40] T. Sadhu, S. Chowdhury, S. Mondal, J. Roy, J. Chakrabarty, S.K. Lahiri, A comparative study of metaheuristics algorithms based on their performance of complex benchmark problems, Decis. Mak.: Appl. Manag. Eng. 6 (1) (2023) 341–364, http://dx.doi.org/10.31181/dmame0306102022r, URL https://dmame-journal.org/index.php/dmame/article/view/386.

[41] E. García González, J.R. Villar, J. Sedano, C. Chira, Benchmarking annalysis for biologica-based metaheuristics, Dyna 99 (3) (2024) 296–305, http://dx.doi.org/10.6036/11070.

[42] E. García, J.R. Villar, C. Chira, J. Sedano, A comparison of meta-heuristic based optimization methods using standard benchmarks, in: P. García Bringas, H. Pérez García, F.J. Martínez de Pisón, J.R. Villar Flecha, A. Troncoso Lora, E.A. de la Cal, A. Herrero, F. Martínez Álvarez, G. Psaila, H. Quintián, E. Corchado (Eds.), Hybrid Artificial Intelligent Systems, Springer International Publishing, 2022, pp. 494–504.

[43] J.T. Bonner, The Evolution of Evolution: Seen through the Eyes of a Slime Mold, BioScience 65 (12) (2015) 1184–1187, http://dx.doi.org/10.1093/biosci/biv154.

[44] V. Nanjundiah, Individual and collective behaviour in cellular slime mould development: contributions of John B<mamfonner (1920–2019), Int. J. Dev. Biol. 63 (2019) 333–342, http://dx.doi.org/10.1387/ijdb.190272vn.

[45] E. García González, J.R. Villar, J. Sedano, A comparison procedure for the evaluation of metaheuristics, in: Hybrid Artificial Intelligent Systems, Springer Nature Switzerland, Cham, 2025, pp. 153–164.

[46] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58, http://dx.doi.org/10.1016/j.orp.2016.09.002.

[47] Z. Ma, G. Wu, P.N. Suganthan, A. Song, Q. Luo, Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms, Swarm Evol. Comput. 77 (2023) 101248, http://dx.doi.org/10.1016/j.swevo.2023.101248.

[48] C.L. Camacho-Villalón, T. Stützle, M. Dorigo, Designing new metaheuristics: Manual versus automatic approaches, Intell. Comput. 2 (2023) 0048, http://dx.doi.org/10.34133/icomputing.0048.

[49] J. Silberholz, B. Golden, S. Gupta, X. Wang, Computational comparison of metaheuristics, in: Handbook of Metaheuristics, Springer International Publishing, Cham, 2019, pp. 581–604, http://dx.doi.org/10.1007/978-3-319-91086-4_18.

[50] M. Ghasemi, M. Zare, A. Zahedi, M.-A. Akbari, S. Mirjalili, L. Abualigah, Geyser inspired algorithm: A new geological-inspired meta-heuristic for real-parameter and constrained engineering optimization, J. Bionic Eng. 21 (1) (2024) 374–408, http://dx.doi.org/10.1007/s42235-023-00437-8.

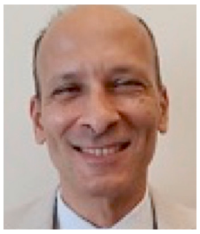
[51] M.A. Farag, M.A. El-Shorbagy, A.A. Mousa, I.M. El-Desoky, A new hybrid metaheuristic algorithm for multiobjective optimization problems, Int. J. Comput. Intell. Syst. 13 (1) (2020) 920–940, http://dx.doi.org/10.2991/ijcis.d.200618.001.

[52] J. Xue, B. Shen, Dung beetle optimizer: a new meta-heuristic algorithm for global optimization, J. Supercomput. 79 (7) (2023) 7305–7336, http://dx.doi.org/10.1007/s11227-022-04959-6.

[53] A.E. Ezugwu, O.J. Adeleke, A.A. Akinyelu, S. Viriri, A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems, Neural Comput. Appl. 32 (10) (2020) 6207–6251, http://dx.doi.org/10.1007/s00521-019-04132-w.

[54] C. Qu, W. Gai, J. Zhang, M. Zhong, A novel hybrid grey wolf optimizer algorithm for unmanned aerial vehicle (UAV) path planning, Knowl.-Based Syst. 194 (2020) 105530, http://dx.doi.org/10.1016/j.knosys.2020.105530.

Enol García González B.Sc. in Software Engineering by the University of Oviedo and M.Sc. in Artificial Intelligence by the Technical University of Madrid. Ph.D. in Computer Science by the University of Oviedo. My main research interests are metaheuristics and optimization algorithms, specially applied to real-world environments. Teaching at the University of Oviedo since 2021.



José R. Villar Industrial Engineer in Electronics and Control (Universidad de Oviedo). Ph. D. in Computer Science (University of León). Currently, he is an Full Professor with the Department of Computer Science at University of Oviedo. His research lines include (i) AI applied to Bio-medical Engineering, (ii) Metaheuristics and its applications to real world problems. Main figures in the last 5 years: i) JCR indexed: 22, ii) in international scientific conferences: 30, (iii) research projects with public funding: 3, (iv) I+D+I projects with private funding: 3. Dr. Villar is a member of the Advisory Committee for the Integrated Computer-Aided Engineering.

Javier Sedano Doctor Ingeniero Industrial, experto en el desarrollo de dispositivos electrónicos (hardware) y en sistemas de control industrial, además del diseño de modelos conexionistas para la identificación, clasificación y modelado de sistemas dinámicos en Big Data y Block Chain. Actualmente es el director general de ITCL Centro Tecnológico e investigador principal del grupo de Investigación de Electrónica Aplicada e Inteligencia Artificial. Ha dirigido y participado en gran número de proyectos de I+D, más de 150, para el desarrollo de prototipos en proyectos precompetitivos, competitivos y de investigación industrial. Ha participado con otros grupos de investigación y durante los últimos 25 años ha trabajado en diferentes proyectos y publicaciones relacionados con inteligencia artificial y sistemas de modelización. Colabora con diferentes grupos de investigación: Grupo de Inteligencia Computacional Aplicada y grupo de investigación de electromagnetismo y electrónica de la Universidad de Burgos, Grupo BISITE de la Universidad de Salamanca, Universidad de Oviedo y Universidad Carlos III. Además, colabora desde hace unos años en la organización de congresos científicos internacionales, comités de programa y de organización. Es miembro del IEEE Systems, Man&Cybernetics Society Spanish Chapter, miembro del IEEE (Institute of Electrical and Electronics Engineers) y revisor de varias revistas con índice de impacto. Cuenta con gran número de publicaciones científicas, más de 100, de tipo internacional, capítulos de libros y participaciones en congresos, asícomo registros de software y patentes industriales, en explotación.

Camelia Chira is professor in computer science at the Faculty of Mathematics and Computer Science, Babeş-Bolyai University (Romania) and senior researcher in Artificial Intelligence within the Research Institute on Artificial Intelligence, Virtual Reality and Robotics. Current main research interests include evolutionary algorithms, nature-inspired computing, complex networks, machine learning, computer vision and bioinformatics. Her research work has generated important scientific contributions in the field of AI and its applications to complex search and optimization problems. She participated in several international research projects with results disseminated in more than 100 publications.

Enrique A. de la Cal Marín received the Computer Science M.Sc. in 1995 (University of Oviedo). Ph.D. degree in Computer Science from the University of Oviedo in 2003 (University of Oviedo). Currently, he is senior lecturer with the Department of Computer Science in the knowledge area of Computer Science and Artificial Intelligence, at University of Oviedo.

He has been involved in 8 national publicly funded projects as — collaborator researcher, 6 regional publicly funded projects. (5 as collaborator researcher and one as main researcher), 6 private contracts with relevant industrial companies (4 as responsible researcher) and 1 international private funded grant.

Also, he is co-author of 25+ papers on impact factor journals and more than 60 contributions to international conferences. He has organized several national and international conferences and special sessions. Besides, he has been the guest editor in several special issues for JCR indexed journals. Finally, he is reviewer for several indexed journals (Neurocomputing-Elsevier, Applied Intelligence — Springer, Journal Logic of IGPL-Oxford and Age and Aging).

His research interests primarily focused in two key areas: The development of novel Artificial Intelligence techniques, with a strong emphasis on Intelligent Analysis of Time Series, Anomaly Detection, and Evolutionary Computation, and applied AI, particularly in Biomedical engineering and Soft Computing Industrial Problem Modelling.

Luciano Sánchez is a professor at the University of Oviedo. Since 2022, he has been the Director of the TotalEnergies Chair of Data Analytics and Artificial Intelligence at the University of Oviedo. He coordinates the research group "Metrology and Models" and is a founding partner of the group's spin-off, Idalia Intelligent Data Analysis S.L. He was a visiting professor at the University of California in Berkeley (1995) and at General Electric (GE Global Research, Schenectady, New York, 1996). His research includes the theoretical study of algorithms for mathematical modelling of systems and intelligent data analysis, as well as their application to industrial modelling, signal processing, and dimensional metrology, with a special interest in low-quality data. He received the Outstanding Paper Award at the FUZZ-IEEE 2013 congress (Hyderabad, India) and the 1st Engineering Innovation Preis from Rolls-Royce (Germany) in 2013.