





A Collection of Robotics Problems for Benchmarking Evolutionary Computation Methods

Jakub Kúdela^(✉) , Martin Juříček , and Roman Parák 

Institute of Automation and Computer Science, Brno University of Technology
Technická 2, 602 00 Brno, Czech Republic
{Jakub.Kudela,200543,Roman.Parak}@vutbr.cz

Abstract. The utilization of benchmarking techniques has a crucial role in the development of novel optimization algorithms, and also in performing comparisons between already existing methods. This is especially true in the field of evolutionary computation, where the theoretical performance of the method is difficult to analyze. For these benchmarking purposes, artificial (or synthetic) functions are currently the most widely used ones. In this paper, we present a collection of real-world robotics problems that can be used for benchmarking evolutionary computation methods. The proposed benchmark problems are a combination of inverse kinematics and path planning in robotics that can be parameterized. We conducted an extensive numerical investigation that encompassed solving 200 benchmark problems by seven selected metaheuristic algorithms. The results of this investigation showed that the proposed benchmark problems are quite difficult (multimodal and non-separable) and that they can be successfully used for differentiating and ranking various metaheuristics.

Keywords: Evolutionary computation · Metaheuristics · Benchmarking · Robotics

1 Introduction

The field of evolutionary computation (EC) produced over its long history several crucial metaheuristic optimization algorithms, that took inspiration from natural processes. These algorithms found their use in numerous complex applications, where the utilization of conventional methods was inadequate or overly computationally demanding [23]. Over the last decade, there has been an explosion of “novel” methods that draw on natural principles [4]. Many of these novel methods have been found to hide their lack of novelty behind a metaphor-rich jargon [3], or flawed experimental analysis [21, 27].

As nature-inspired metaheuristics are usually hard to analyze analytically, their utility is conventionally analyzed through benchmarking [13]. There have been many different benchmark functions and sets proposed by various authors

[9,22], but the most popular and widely used benchmark set have been developed for special sessions (competitions) on black-box optimization in two EC conferences: the IEEE Congress on Evolutionary Computation (CEC), and the Genetic and Evolutionary Conference (GECCO), where the Black-Box Optimization Benchmarking (BBOB) workshop was held. The BBOB functions constitute a part of the COCO platform for comparing optimization algorithms [12], while the benchmark functions from the different CEC competitions can be found on GitHub of one of the authors. It was shown that the characteristics of the functions used in the CEC and BBOB benchmark sets are very different [10]. However, the use of these benchmark sets is not without critique, as some authors criticized the artificial nature of these benchmark sets [31], and recommended testing optimization algorithms on real-world problems instead [39].

A possible way of comparing the characteristics of the different benchmark sets is by using the Exploratory Landscape Analysis (ELA) [25]. In this approach, the benchmark functions are represented by a collection of landscape features (numerical measures) that describe the different aspects of the functions. The ELA can subsequently be used for designing representative benchmark suites [5], or for feature-based algorithm selection [37].

One of the areas that utilized EC methods to a large extent is robotics [29,40]. The locomotion of snake-like robots using genetic algorithms (GA) was investigated in [14]. The inverse kinematics (IK) problem in robot path tracking [15,30] was approached using particle swarm optimization (PSO) variants [8] or slime mould algorithm [41]. EC methods were used in tracking control of redundant mobile manipulator [20], robot part sequencing and allocation problem with collision avoidance [6], or in the control tuning of omnidirectional mobile robots [33]. Another robotics application where EC methods are widely utilized is trajectory or path planning [1,24]. Using GA and PSO, the authors of [28] studied energy-efficient robot configuration and motion planning. Another applications of EC methods [32] investigated an autonomous unmanned aerial vehicle path-planning for predisaster assessment or path planning and tracking of a quadcopter for payload hold-release missions [2].

In this paper, we present a collection of parametrizable problems in robotics that combine inverse kinematics and path planning problems, and show that they can be successfully used in benchmarking EC methods. The rest of the paper is structured as follows. Section 2 describes the 6-DOF (Degrees of Freedom) collaborative robotic arm that is used as the base framework. In Sect. 3 are defined the benchmark problems and their parametrization. In Sect. 4 we briefly describe the seven EC methods that were selected for running numerical tests on the proposed benchmarks. Section 5 gives a detailed account of the results of the numerical test. In Sect. 6, we provide ELA of the problems and a comparison to problems from the BBOB and CEC 2014 sets. Finally, conclusions and future research directions are discussed in Sect. 7.

2 Forward Kinematics of a Robotic Arm

As the framework for the benchmark problems, we chose the 6-DOF collaborative robotic arm UR3 CB-Series¹, shown in Fig. 1. The solution of forward kinematics (FK) requires the knowledge of the so-called Denavit-Hartenberg (D-H) table, shown in Table 1. The table has become a standard used in robotics, having been first published in [7]. The commonly used convention of the D-H table is defined by four parameters. These parameters describe how the reference frame of each link is attached to the robot. Each inertial reference frame of each link is then assigned an additional robot reference frame. The parameters are defined for each joint $i \in [1, n]$, which represents the table:

- α_i : angle about common normal from z_i to z_{i+1}
- θ_i : angle about previous z axis from x_i to x_{i+1}
- a_i : length of the common normal or radius about previous z axis for revolute joint
- d_i : offset along previous z axis to the common normal

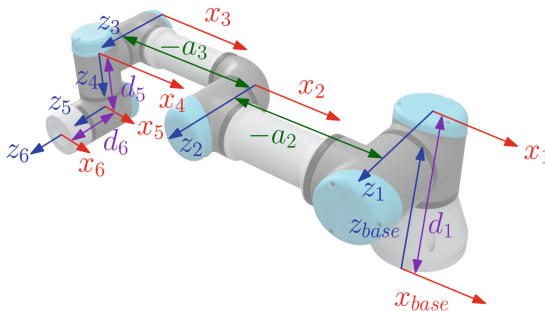


Fig. 1. UR3 D-H parameters

The solution for the calculation of the FK itself using the D-H parameters can then be approached by multiplying the individual D-H matrices. For the solution, so-called standard (distal) matrices or modified (proximal) matrices can be used. The basic difference between the standard D-H parameters and the modified D-H parameters is the locations of the coordinates system attached to the links. The modification consists of both the calculation of the D-H matrix itself and the change of the D-H table. To switch from a standard D-H table to a modified D-H table, it is necessary to perform the shift operation $\alpha_i = \alpha_{i-1}$ and $a_i = a_{i-1}$ where $i \in [1, n]$, $\alpha_1 = \alpha_n$, $a_1 = a_n$. The calculation of the forward kinematics is then a matrix in which the vector \mathbf{p} represents the translational part and \mathbf{R} represents the rotation matrix, which can then be converted into

¹ <https://www.universal-robots.com/cb3/>.

quaternions or euler angles. The modified D-H matrix can then have its advantage when calculating the Jacobian or determining the rotation and translation of the individual joints of the robot. Homogeneous transformation matrix \mathbf{T}_i is represented as the product of four basic transformations [34]. The calculation of the resulting matrix \mathbf{T}_n of the n -axis robot can generally be described mathematically as follows:

$$\mathbf{T}_n = \prod_{i=1}^n \mathbf{T}_i \quad (1)$$

$$\begin{aligned} \mathbf{T}_i &= \mathbf{Rot}_{z_i \theta_i} \mathbf{Trans}_{z_i d_i} \mathbf{Trans}_{x_i a_i} \mathbf{Rot}_{x_i \alpha_i} = \\ &= \left[\begin{array}{ccc|c} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & r_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & r_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ \hline 0 & 0 & 0 & 1 \end{array} \right] = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{p} \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \\ \mathbf{T}_i &= \mathbf{Rot}_{x_i \alpha_{i-1}} \mathbf{Trans}_{x_i a_{i-1}} \mathbf{Rot}_{z_i \theta_i} \mathbf{Trans}_{z_i d_i} = \\ &= \left[\begin{array}{ccc|c} \cos \theta_i & -\sin \theta_i & 0 & a_{i-1} \\ \sin \theta_i \cos \alpha_{i-1} & \cos \theta_i \cos \alpha_{i-1} & -\sin \alpha_{i-1} & -d_i \sin \alpha_{i-1} \\ \sin \theta_i \sin \alpha_{i-1} & \cos \theta_i \sin \alpha_{i-1} & \cos \alpha_{i-1} & -d_i \cos \alpha_{i-1} \\ \hline 0 & 0 & 0 & 1 \end{array} \right] = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{p} \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \end{aligned}$$

Table 1. Universal Robots UR3 CB-Series D-H table

Joint i	θ_i [rad]	α_i [rad]	a_i [m]	d_i [m]
1	0	$\pi/2$	0	0.1519
2	0	0	-0.24365	0
3	0	0	-0.21325	0
4	0	$\pi/2$	0	0.11235
5	0	$-\pi/2$	0	0.08535
6	0	0	0	0.0819

3 Definition of the Benchmark Problems

The α_i , a_i , and d_i values are fixed (as they depend on the particular robot design) and the robot is controlled by changing the angles θ_i . The [x, y, z]-coordinate position of the last link of the robot can be found as the translation part \mathbf{p} in the matrix \mathbf{T}_n . We will denote this relationship simply as

$$[x, y, z]^T = FK(\theta), \quad (2)$$

where FK is the forward kinematics solution, and $\theta = [\theta_1, \dots, \theta_6]^T, \theta_i \in [-2\pi, 2\pi], i = 1, \dots, 6$. In the proposed benchmark problems, we will be interested in the trajectories of the robot's last link (end-effector), which corresponds to the way θ changes in time τ , and can be expressed as

$$[x(\tau), y(\tau), z(\tau)]^T = FK(\theta(\tau)). \quad (3)$$

The first quality of the trajectory we will use is its length L , which (starting in $\tau = 0$ and ending in $\tau = 1$) can be expressed as

$$L = \int_0^1 \sqrt{x'(\tau)^2 + y'(\tau)^2 + z'(\tau)^2} d\tau. \quad (4)$$

The second quality of the trajectory will be its closeness to a predefined point $[x_p, y_p, z_p]$, which can be written as

$$\min_{\tau \in [0,1]} \|[x(\tau) - x_p, y(\tau) - y_p, z(\tau) - z_p]\|_2, \quad (5)$$

and, in the case of multiple predefined points $[x_p^j, y_p^j, z_p^j], j = 1, \dots, P$, the closeness (C) to the farthest one

$$C = \max_{j=1,\dots,P} \min_{\tau \in [0,1]} \|[x(\tau) - x_p^j, y(\tau) - y_p^j, z(\tau) - z_p^j]\|_2. \quad (6)$$

As continuous control would pose too complex of a problem, we will restrict our attention to a situation where the angles θ change linearly from one setting θ^a to the next θ^b , i.e.

$$\theta(\tau) = \theta^a + \tau(\theta^b - \theta^a). \quad (7)$$

In the case where we want to have multiple points of change $\theta^0, \dots, \theta^M$ one of the possibilities is to model it as M time intervals of length 1, i.e.:

$$\hat{\theta}(\tau) = \theta^\iota + (\tau - \iota)(\theta^\iota - \theta^{\iota+1}), \quad \text{for } \tau \in [\iota, \iota + 1], \iota = 0, \dots, M - 1. \quad (8)$$

Even with the restriction on linear change in θ , the expressions (4) and (6) would be hard to compute analytically, which is why we resort to a discretization of τ into $M \cdot N$ evenly spaced values $[\tau_1 = 0, \dots, \tau_{M \cdot N} = M]$ and compute:

$$[x(\tau_i), y(\tau_i), z(\tau_i)] = FK(\hat{\theta}(\tau_i)), \quad i = 1, \dots, M \cdot N \quad (9)$$

$$\hat{L} = \sum_{i=1}^{M \cdot N - 1} \|[x(\tau_{i+1}) - x(\tau_i), y(\tau_{i+1}) - y(\tau_i), z(\tau_{i+1}) - z(\tau_i)]\|_2 \quad (10)$$

$$\hat{C} = \max_{j=1,\dots,P} \min_{\tau_i, i=1,\dots,M \cdot N} \|[x(\tau_i) - x_p^j, y(\tau_i) - y_p^j, z(\tau_i) - z_p^j]\|_2. \quad (11)$$

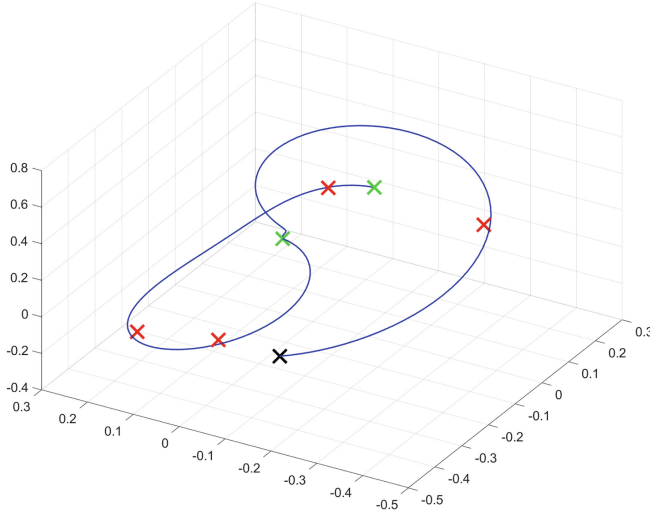


Fig. 2. Trajectory (blue line) of one solution starting of the black cross with $M = 2$ points of change (green crosses) and $P = 4$ predefined points (red crosses). Using $N = 100$, $\gamma = 100$ the objective value $f(\theta^1, \theta^2) = 4.8641$. (Color figure online)

For a given starting position θ^0 , the objective function for all the considered benchmark problems has the form:

$$f(\theta^1, \dots, \theta^M) = \gamma \cdot \hat{L} + \hat{C}, \quad (12)$$

where the parameter $\gamma \geq 0$ lets us control the degree to which we prefer trajectories with shorter length (higher γ), or higher precision in reaching the predefined points (lower γ). The resulting optimization problem is a “simple” box-constrained one:

$$\begin{aligned} & \text{minimize } f(\theta^1, \dots, \theta^M) \\ & \text{subject to } \theta^\iota \in [-2\pi, 2\pi]^6, \iota = 1, \dots, M \end{aligned}$$

The resulting benchmark function can be parametrized by:

- (i) the number of points of change M , which determine the dimension of the optimization problem $D = 6 \cdot M$
- (ii) the number of predefined points P to which the trajectory should get close to
- (iii) the coefficient γ that scales the two objectives (trajectory length and closeness to the farthest predefined point)

Figure 2 shows a solution to one problem instance with $M = 2$, $P = 4$, and $\gamma = 100$ (with $N = 100$ as the discretization constant). Figure 3 shows the sensitivity of the objective function value on the first two components of θ_1 . It can readily be seen that the objective is multimodal and nonseparable, which are both desirable characteristics in benchmark functions.

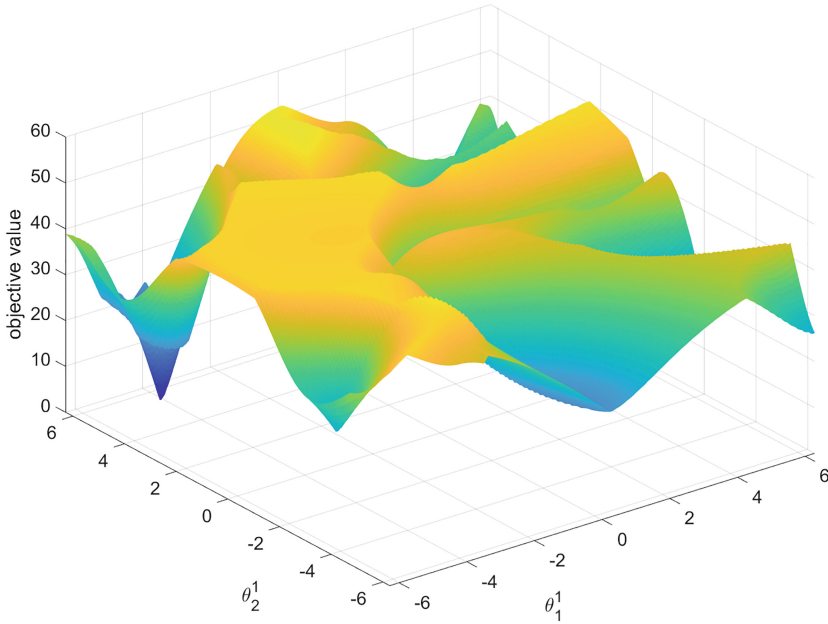


Fig. 3. Sensitivity of the objective value of the solution shown in Fig. 2 on the first two components of θ^1 .

4 Selected Algorithms for the Numerical Investigation

In order to showcase the capabilities of the proposed benchmark functions in differentiating various metaheuristics, we chose seven representative methods. Four of them constitute the “standard algorithms”:

PSO: One of the oldest selected methods for benchmarking is Particle swarm optimization (PSO) [18]. This method was designed by simulating a simplified social model inspired by the foraging behaviour of a bird flocking or fish schooling.

DE: Another of the old methods is Differential evolution (DE) [36]. In essence, DE represents a method that aims to maintain and create new populations of candidate solutions by combining existing ones according to given rules and keeping the candidate solution with the best properties in the defined optimization problem.

CMA-ES: The last old methods selected for benchmarking is the Covariance matrix adaptation evolution strategy (CMA-ES) [11]. CMA-ES combines the use of evolution strategy and covariance matrix adaptation to apply numerical optimization.

ABC: One of the more recent metaheuristics is Artificial bee colony (ABC) [16]. This method, like PSO, is inspired by the biological behaviour of animals, in this case, based on the intelligent foraging behaviour of a bee swarm.

The other three methods constitute some of the most successful algorithm in the CEC competitions.

- HSES: Hybrid Sampling Evolution Strategy (HSES) was the winner of the CEC'18 Competition. It is an evolution strategy optimization algorithm that combined CMA-ES and the univariate sampling method [42].
- AGSK: Adaptive Gaining-Sharing Knowledge (AGSK) was the runner-up of the CEC'20 competition. The algorithm improved the original GSK algorithm by adding adaptive settings to its two control parameters: the knowledge factor and ratio, which control junior and senior gaining and sharing phases during the optimization process [26].
- LSHADE: The last selected method is L-SHADE or Success-history based adaptive differential evolution with linear population size reduction [38]. This metaheuristic method has its basis in adaptive DE, which involves success-history-based parameter adaptation. The proposed method then provides an extension in the form of using linear population size reduction, which results in population size reduction according to a linear function.

We also decided to add to the comparison a random search (RS) method, that simply sampled (using uniform sampling) maximum available number of points and chose the best one among them.

5 Numerical Investigation

For the numerical investigation of the selected algorithms on the proposed benchmark functions we chose the problems with $M = [1, \dots, 5]$ and $P = [3, \dots, 6]$ (i.e., 20 possibilities). For each of the 20 problems, 10 random instances (random points in the reachable space of the robot) were generated. As the metaheuristics are stochastic, each of them was run 20 times on a given instance (to get statistically representative results). In total, each of the seven compared algorithms was run on 200 optimization problems. The maximum number of function evaluation (FES) was set to $FES = 10,000 \cdot D$. The benchmark functions (as well as the metaheuristic algorithms) were implemented in MATLAB and can be found at a public Zenodo² and GitHub repository³. We did not perform any parameter tuning [17].

A representative result of the computations can be seen in Fig. 4, where the best solutions/trajectories (out of the 20 runs) found by the different methods for one problem instance (with $P = 4$, $M = 2$) are shown. For this instance, CMAES found the best solution, followed by PSO, ABC, and LSHADE. An interesting observation is that these solutions are qualitatively quite different. Although they all come close to the desired points, the order in which they approach differs.

² <https://doi.org/10.5281/zenodo.7584647>.

³ <https://github.com/JakubKudela89/Robotics-Benchmarking>.

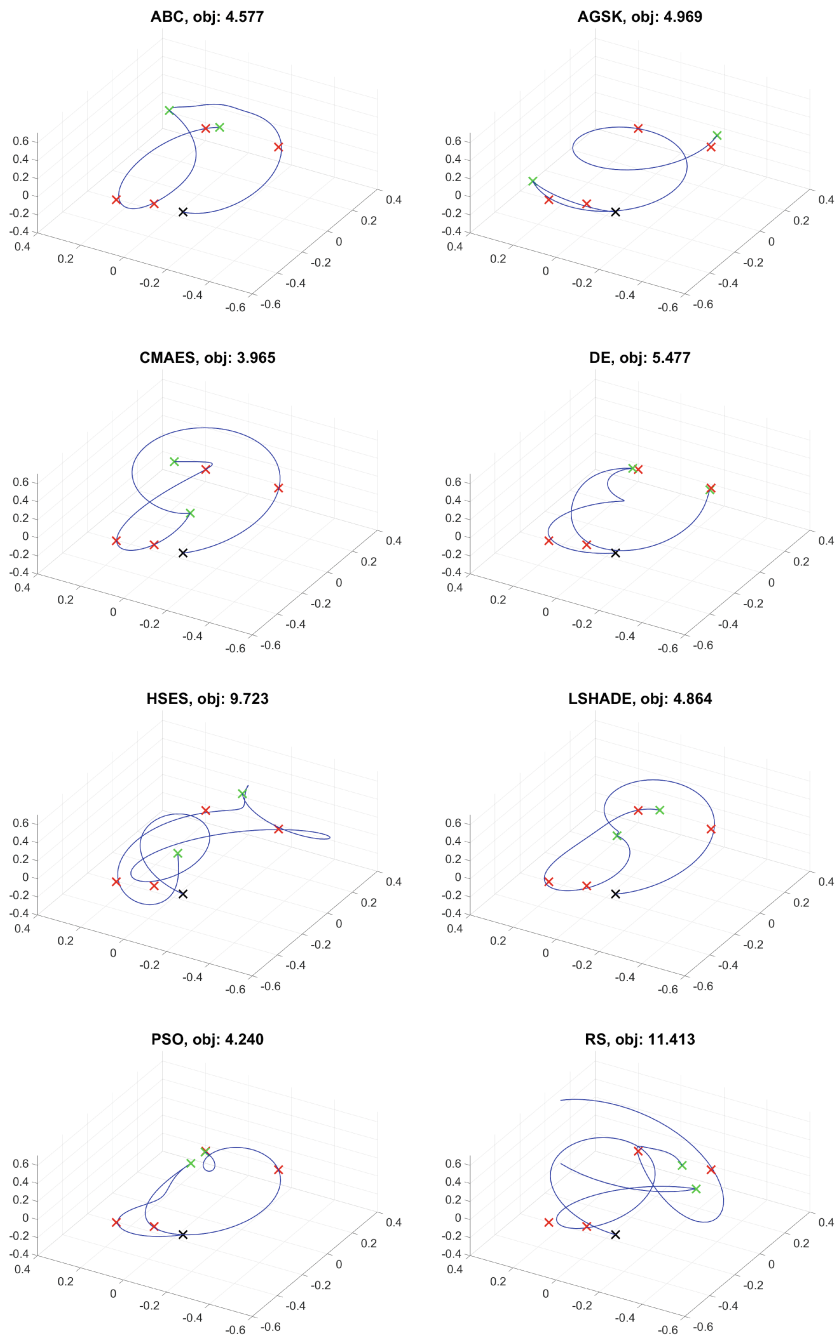


Fig. 4. Best solutions/trajectories (out of the 20 runs) found by the different methods for one problem instance (with $P = 4$, $M = 2$).

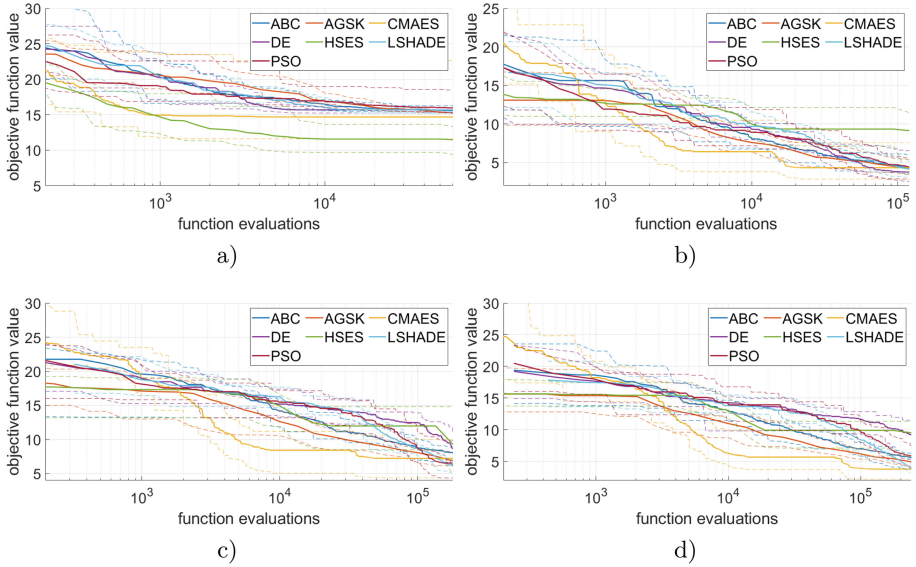


Fig. 5. Convergence plots for different instances with, a) $D = 6$ ($P = 4, M = 1$, instance 4), b) $D = 12$ ($P = 3, M = 2$, instance 1), c) $D = 18$ ($P = 5, M = 3$, instance 5), d) $D = 24$ ($P = 4, M = 4$, instance 7). Solid lines show the median values, while dashed lines show the worst and the best values (over the 20 runs).

Representative convergence plots for the considered methods are shown in Fig. 5, where the solid lines show the progression of the median values, while the dashed lines show the worst and best values. These convergence plots show that there was quite a large difference between the selected methods in basically all stages of the search. It also shows that there is a substantial variance in the performance of a given algorithm within the instances.

For the ranking of the methods, we chose to focus only on the values at end of the search (after using all the *FES* available function evaluations). The results of the Friedman rank tests on the 20 problems (each having 10 instances on which each of the methods was run 20 times) are shown in Table 2. From these results, we can see that the proposed benchmark function were successful in differentiating the various methods, especially for problems in higher dimensions. What is interesting is the effect of M and P on the resulting ranking. As M directly influences the problem dimension, it had, unsurprisingly, a substantial effect on the ranking. Interestingly, the effect of P was also noticeable - increasing P generally meant that the (relative) performance of DE and ABC deteriorated, while the performance of HSES and PSO improved. Overall, the best-performing method was LSHADE, followed by CMAES and AGKS. The relatively bad ranking of HSES (and also AGSK) might be explained by the fact that the CEC competitions allow for much more function evaluations (which is what both HSES and AGSK were designed and tuned for).

Table 2. Mean ranks from Friedman tests for the different benchmark problems. Best three methods are highlighted in bold.

$D = 6, \text{FES} = 60,000$								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 1$	3.70	2.36	6.18	3.93	4.87	2.83	4.98	7.16
$P = 4, M = 1$	4.24	2.87	5.96	4.34	2.78	3.24	5.52	7.06
$P = 5, M = 1$	4.06	2.99	6.28	3.63	2.97	3.44	5.63	7.02
$P = 6, M = 1$	4.38	2.98	5.98	3.94	2.40	3.48	5.69	7.17
$D = 12, \text{FES} = 120,000$								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 2$	4.65	4.18	3.72	2.20	6.41	2.58	4.48	7.81
$P = 4, M = 2$	4.83	4.08	4.43	2.77	5.31	2.43	4.26	7.91
$P = 5, M = 2$	4.76	3.78	4.63	2.78	5.66	2.16	4.35	7.91
$P = 6, M = 2$	4.60	3.76	5.15	2.97	5.20	2.44	4.06	7.86
$D = 18, \text{FES} = 180,000$								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 3$	4.22	3.97	2.73	4.90	6.15	2.00	4.17	7.87
$P = 4, M = 3$	4.58	3.50	3.08	5.12	5.72	1.70	4.40	7.92
$P = 5, M = 3$	4.61	3.63	3.23	5.18	5.65	2.00	3.86	7.86
$P = 6, M = 3$	4.37	3.38	3.32	6.47	5.15	1.75	3.74	7.83
$D = 24, \text{FES} = 240,000$								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 4$	4.29	3.89	2.13	6.28	5.55	1.65	4.27	7.96
$P = 4, M = 4$	4.31	3.66	2.41	6.43	5.70	1.51	4.04	7.96
$P = 5, M = 4$	4.40	3.62	2.71	6.45	5.69	1.54	3.71	7.91
$P = 6, M = 4$	4.51	3.66	2.10	6.96	5.18	1.89	3.88	7.84
$D = 30, \text{FES} = 300,000$								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 5$	4.26	4.36	1.80	6.67	5.24	1.61	4.10	7.98
$P = 4, M = 5$	4.13	3.96	2.03	6.84	5.55	1.49	4.05	7.98
$P = 5, M = 5$	4.28	3.85	2.03	6.90	5.67	1.56	3.79	7.95
$P = 6, M = 5$	4.45	3.62	1.88	7.00	5.43	1.67	4.03	7.93

6 Exploratory Landscape Analysis

We use ELA features to show how the proposed robotics problems compare to the BBOB and CEC 2014 benchmark suits. In order to calculate the ELA features, we used the flacco library [19]. As we are not able to supply exact function definitions (in our case, the function evaluates a simulation of the movement of the robotic arm), we chose ELA feature sets which only require samples of input and function value pairs: `ela_distr`, `ela_meta`, `disp`, `nbc`, `pca`, and `ic`. We used uniform sampling with $250D$ samples. As the dimensions of the problems in BBOB ($D = 2, 3, 5, 10, 20$, and 40), CEC 2014 ($D = 2, 10, 30, 50$, and 100),

Table 3. Minimum and Maximum values of the relevant ELA features on the three benchmark sets. Extremal values are highlighted in bold.

ELA feature	BBOB		CEC 2014		This study	
	min	max	min	max	min	max
ela_distr.skewness	-2.97E+00	8.28E+00	-6.63E-01	6.47E+00	-4.76E-01	9.54E-01
ela_distr.kurtosis	-4.94E-01	9.67E+01	-3.38E-01	6.50E+01	-8.43E-01	2.30E+00
ela_distr.number_of_peaks	1.00E+00	1.80E+01	1.00E+00	2.60E+01	1.00E+00	9.00E+00
ela_meta.lin_simple.adj_r2	1.38E-04	1.00E+00	-2.30E-03	8.23E-01	-1.11E-03	2.19E-01
ela_meta.lin_simple.intercept	-9.17E+02	9.62E+08	5.22E+02	5.63E+10	2.37E+01	4.37E+01
ela_meta.lin_w.interact.adj_r2	2.14E-04	1.00E+00	-9.41E-04	9.04E-01	5.78E-03	2.61E-01
ela_meta.quad_simple.adj_r2	3.98E-03	1.00E+00	-3.61E-03	9.88E-01	4.24E-02	2.52E-01
ela_meta.quad_w.interact.adj_r2	3.67E-05	1.00E+00	-1.26E-02	1.00E+00	1.64E-01	3.78E-01
disp_ratio_median_05	7.17E-01	1.01E+00	7.26E-01	1.02E+00	9.93E-01	1.05E+00
nbc.nb_fitness.cor	-6.41E-01	-1.78E-01	-6.30E-01	-1.90E-01	-5.83E-01	-4.83E-01
pca.expl_var.cor_init	8.18E-01	9.09E-01	8.18E-01	9.09E-01	9.23E-01	9.23E-01
pca.expl_var_PC1.cov_init	1.07E-01	1.00E+00	1.10E-01	1.00E+00	2.36E-01	4.36E-01

and our robotics problems ($D = 6, 12, 18, 24$, and 30) differ, we used $D = 10$ for the BBOB and CEC 2014 benchmarks, and $D = 12$ for our robotics problems, as these were the closest choices. There were 24 problems in the BBOB set, 30 problems in the CEC 2014 set, and 40 robotics problems (10 instances for $P = 3, 4, 5$, and 6).

We followed the methodology described in [35] for the selection and visualization of the relevant ELA features. The features that produced constant results on every problem and those that produced invalid values were removed. Another set of removed features were the ones that were sensitive to scaling and shifting. The last batch of features that got removed were the highly correlated ones. The 12 features that remained, along with their maximum and minimum values on the three benchmark sets are shown in Table 3.

For further analysis, the values of the ELA features on the three benchmark sets were normalized, and we used Principal Component Analysis (PCA) to reduce the number of features even further. Figure 6 shows a representation of the 12 PCA components obtained when comparing the ELA features (normalized) calculated on the combined set of CEC 2014, BBOB, and robotics problems. Using the first 8 components explained 99.85% of the variance.

For visualizing the results, we used the t-Distributed Stochastic Neighbor Embedding (t-sne). In this visualization, which is shown in Fig. 7, benchmark problems that have similar ELA features should be shown close to each other. We can see that the proposed robotics benchmarks are not very similar to functions in either BBOB or CEC 2014 sets. They are also not very similar to each other, at least in the sense of the performed analysis.

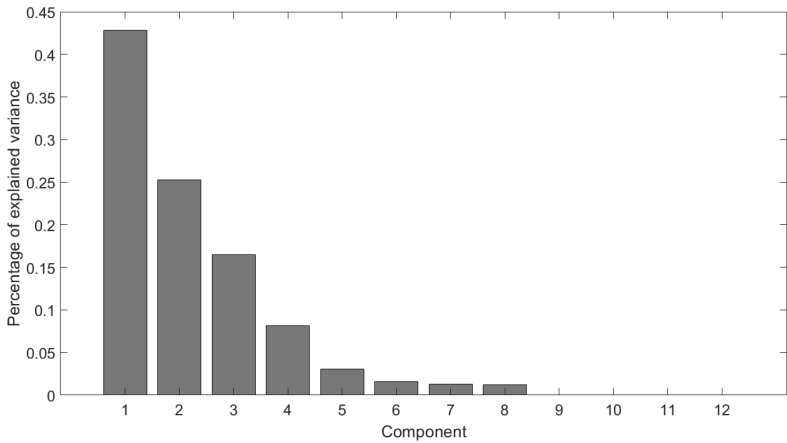


Fig. 6. The amount of explained variance per component when performing PCA on the ELA features calculated on the combined set of 2014 CEC, BBOB, and robotics problems.

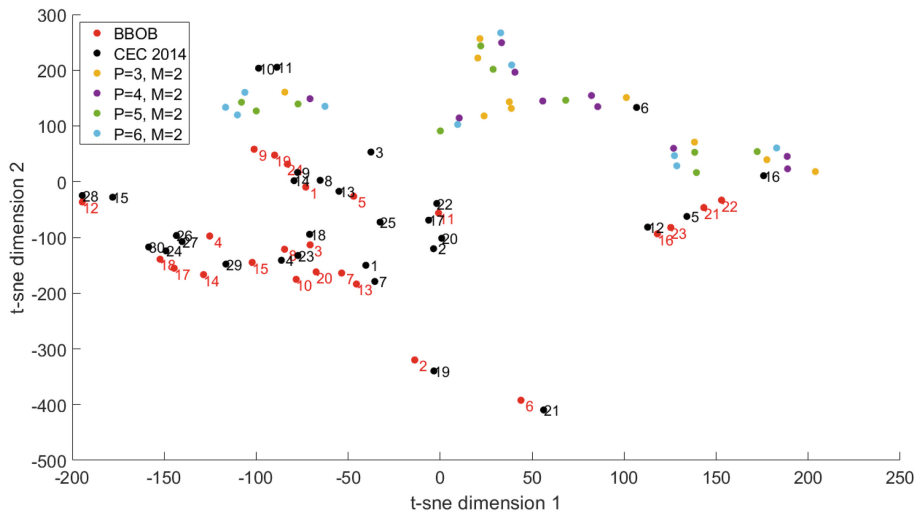


Fig. 7. The t-sne visualization of the ELA features (after normalization and using the first components from the PCA) of the three benchmark sets.

7 Conclusion

In this paper, we presented a collection of parametrizable benchmark functions for comparing EC methods. The benchmark functions were a blend of real-world robotics problems in inverse kinematics and path planning. We conducted a thorough numerical investigation of the proposed benchmark problems - each of the seven selected methods for numerical comparison was used to solve 200

benchmark problems. The results of this investigation are that the proposed benchmark problems are quite difficult (multimodal and non-separable) and that they can be successfully used for differentiating and ranking various metaheuristics. The proposed methods can be applied to different types of 6-DOF robotic manipulators by simply changing the parameters of the D-H table.

There is still further work to be done. The benchmark problems will be implemented in more languages, especially the ones that are most used for the development of EC methods (e.g., Python and C++). We also plan on the integration with a proper simulator of the robotic arm in Unity. Another research directions will be in investigating more problem types (such as energy optimization, or following a fixed sequence of points), the effect of different choices for the robotic arm model, and in comparing deterministic and surrogate-based techniques.

Acknowledgements. This work was supported by the IGA BUT No. FSI-S-23-8394 “Artificial intelligence methods in engineering tasks”.

References

1. Batista, J., Souza, D., Silva, J., Ramos, K., Costa, J., dos Reis, L., Braga, A.: Trajectory planning using artificial potential fields with metaheuristics. *IEEE Lat. Am. Trans.* **18**(05), 914–922 (2020)
2. Belge, E., Altan, A., Hacıoğlu, R.: Metaheuristic optimization-based path planning and tracking of quadcopter for payload hold-release mission. *Electronics* **11**(8), 1208 (2022)
3. Camacho Villalón, C.L., Stützle, T., Dorigo, M.: Grey wolf, firefly and bat algorithms: three widespread algorithms that do not contain any novelty. In: Dorigo, M., et al. (eds.) *ANTS 2020*. LNCS, vol. 12421, pp. 121–133. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60376-2_10
4. Campelo, F., Aranha, C.D.C.: Sharks, zombies and volleyball: lessons from the evolutionary computation bestiary. In: *CEUR Workshop Proceedings*, vol. 3007, p. 6. CEUR Workshop Proceedings (2021)
5. Cenikj, G., Lang, R.D., Engelbrecht, A.P., Doerr, C., Korošec, P., Eftimov, T.: Selector: Selecting a representative benchmark suite for reproducible statistical comparison. In: *Proceedings of the Genetic and Evolutionary Computation Conference. GECCO 2022*, New York, NY, USA, pp. 620–629. Association for Computing Machinery (2022)
6. Croucamp, M., Grobler, J.: Metaheuristics for the robot part sequencing and allocation problem with collision avoidance. In: Marreiros, G., Melo, F.S., Lau, N., Lopes Cardoso, H., Reis, L.P. (eds.) *EPIA 2021*. LNCS (LNAI), vol. 12981, pp. 469–481. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86230-5_37
7. Denavit, J., Hartenberg, R.S.: A kinematic notation for lower-pair mechanisms based on matrices. *J. Appl. Mech.* **22**(2), 215–221 (2021)
8. Dereli, S., Köker, R.: A meta-heuristic proposal for inverse kinematics solution of 7-dof serial robotic manipulator: quantum behaved particle swarm algorithm. *Artif. Intell. Rev.* **53**(2), 949–964 (2020)
9. García-Martínez, C., Gutiérrez, P.D., Molina, D., Lozano, M., Herrera, F.: Since cec 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis’s weakness. *Soft. Comput.* **21**(19), 5573–5583 (2017)

10. Garden, R.W., Engelbrecht, A.P.: Analysis and classification of optimisation benchmark functions and benchmark suites. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1641–1649. IEEE (2014)
11. Hansen, N.: The CMA evolution strategy: A tutorial. arXiv preprint, [arXiv:1604.00772](https://arxiv.org/abs/1604.00772) (2016). <https://doi.org/10.48550/ARXIV.1604.00772>
12. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: Coco: A platform for comparing continuous optimizers in a black-box setting. *Optim. Meth. Software* **36**(1), 114–144 (2021)
13. Hellwig, M., Beyer, H.G.: Benchmarking evolutionary algorithms for single objective real-valued constrained optimization—a critical review. *Swarm Evol. Comput.* **44**, 927–944 (2019)
14. Hulka, T., Matoušek, R., Dobrovský, L., Dosoudilová, M., Nolle, L.: Optimization of snake-like robot locomotion using GA: Serpenoid design. *Mendel J.* **26**(1), 1–6 (2020)
15. Kanagaraj, G., Masthan, S.S., Vincent, F.Y.: Meta-heuristics based inverse kinematics of robot manipulator’s path tracking capability under joint limits. *Mendel J.* **28**(1), 41–54 (2022)
16. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report TR06, Erciyes University (2005)
17. Kazikova, A., Pluhacek, M., Senkerik, R.: Why tuning the control parameters of metaheuristic algorithms is so important for fair comparison? In: *Mendel*. vol. 26, pp. 9–16 (2020)
18. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 (1995)
19. Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package Flacco. In: Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., Vichi, M. (eds.) *Applications in Statistical Computing. SCDAGO*, pp. 93–123. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25147-5_7
20. Khan, A.H., Li, S., Chen, D., Liao, L.: Tracking control of redundant mobile manipulator: an RNN based metaheuristic approach. *Neurocomputing* **400**, 272–284 (2020)
21. Kudela, J.: A critical problem in benchmarking and analysis of evolutionary computation methods. *Nature Mach. Intell.* **4**, 1238–1245 (2022)
22. Kudela, J., Matousek, R.: New benchmark functions for single-objective optimization based on a zigzag pattern. *IEEE Access* **10**, 8262–8278 (2022)
23. Kudela, J., Matousek, R.: Recent advances and applications of surrogate models for finite element method computations: a review. *Soft Comput.* 1–25 (2022)
24. Kumar, R., Singh, L., Tiwari, R.: Comparison of two meta-heuristic algorithms for path planning in robotics. In: *2020 International Conference on Contemporary Computing and Applications (IC3A)*, pp. 159–162. IEEE (2020)
25. Mersmann, O., Preuss, M., Trautmann, H.: Benchmarking evolutionary algorithms: towards exploratory landscape analysis. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) *PPSN 2010. LNCS*, vol. 6238, pp. 73–82. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_8
26. Mohamed, A.W., Hadi, A.A., Mohamed, A.K., Awad, N.H.: Evaluating the performance of adaptive gainsharing knowledge based algorithm on CEC 2020 benchmark problems. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE (2020)

27. Niu, P., Niu, S., Chang, L., et al.: The defect of the grey wolf optimization algorithm and its verification method. *Knowl.-Based Syst.* **171**, 37–43 (2019)
28. Nonoyama, K., Liu, Z., Fujiwara, T., Alam, M.M., Nishi, T.: Energy-efficient robot configuration and motion planning using genetic algorithm and particle swarm optimization. *Energies* **15**(6), 2074 (2022)
29. Parak, R., Matousek, R.: Comparison of multiple reinforcement learning and deep reinforcement learning methods for the task aimed at achieving the goal. *Mendel J.* **27**(1), 1–8 (2021)
30. Pattnaik, S., Mishra, D., Panda, S.: A comparative study of meta-heuristics for local path planning of a mobile robot. *Eng. Optim.* **54**(1), 134–152 (2022)
31. Piotrowski, A.P.: Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions. *Inf. Sci.* **297**, 191–201 (2015)
32. Qadir, Z., Zafar, M.H., Moosavi, S.K.R., Le, K.N., Mahmud, M.P.: Autonomous UAV path-planning optimization using metaheuristic approach for predisaster assessment. *IEEE Internet Things J.* **9**(14), 12505–12514 (2021)
33. Serrano-Pérez, O., Villarreal-Cervantes, M.G., González-Robles, J.C., Rodríguez-Molina, A.: Meta-heuristic algorithms for the control tuning of omnidirectional mobile robots. *Eng. Optim.* (2019)
34. Siciliano, B., Khatib, O. (eds.): Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-32552-1>
35. Škvorc, U., Eftimov, T., Korošec, P.: Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Appl. Soft Comput.* **90**, 106138 (2020)
36. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11**(4), 341–359 (1997)
37. Tanabe, R.: Benchmarking feature-based algorithm selection systems for black-box numerical optimization. *IEEE Trans. Evol. Comput.* **26**, 1321–1335 (2022)
38. Tanabe, R., Fukunaga, A.S.: Improving the search performance of shade using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1658–1665. IEEE (2014)
39. Tzanetos, A., Dounias, G.: Nature inspired optimization algorithms or simply variations of metaheuristics? *Artif. Intell. Rev.* **54**(3), 1841–1862 (2021)
40. Yadav, V., Botchway, R.K., Senkerik, R., Oplatkova, Z.K.: Robotic automation of software testing from a machine learning viewpoint. *Mendel J.* **27**(2), 68–73 (2021)
41. Yin, S., Luo, Q., Zhou, G., Zhou, Y., Zhu, B.: An equilibrium optimizer slime mould algorithm for inverse kinematics of the 7-dof robotic manipulator. *Sci. Rep.* **12**(1), 1–28 (2022)
42. Zhang, G., Shi, Y.: Hybrid sampling evolution strategy for solving single objective bound constrained problems. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–7. IEEE (2018)