Package 'marinepredator'

January 30, 2026

Type Package

Title Marine Predators Algorithm

Version 0.0.0.9000

Description Implementation of the Marine Predators Algorithm (MPA) in R. MPA is a nature-inspired optimization algorithm that follows the rules governing optimal foraging strategy and encounter rate policy between predator and prey in marine ecosystems. Based on the paper by Faramarzi et al. (2020) <doi:10.1016/j.eswa.2020.113377>.

License file LICENSE

URL https://github.com/urbs-dev/marinepredator

BugReports https://github.com/urbs-dev/marinepredator/issues

Language en

Encoding UTF-8

Imports dplyr, stats

Suggests testthat (>= 3.0.0), covr, pkgdown, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Marc Grossouvre [trl, cre] (R implementation),
Afshin Faramarzi [aut] (Original MATLAB code),
Seyedali Mirjalili [aut] (Original MATLAB code)

Maintainer Marc Grossouvre <marc@grossouvre.fr>

Contents

`marinepredator-package`

mpa: Marine Predators Algorithm

Description

Implementation of the Marine Predators Algorithm (MPA) in R. MPA is a nature-inspired meta-heuristic optimization algorithm that follows the rules governing optimal foraging strategy and encounter rate policy between predator and prey in marine ecosystems.

Details

The Marine Predators Algorithm is based on the foraging behavior of marine predators and their interactions with prey. The algorithm operates in three phases that balance exploration and exploitation:

1. **Phase 1** (High velocity ratio): Prey moves faster than predator. Exploration is emphasized using Brownian motion.

2. **Phase 2** (Unit velocity ratio): Predator and prey move at similar speeds. Mixed strategy with both Brownian motion and Levy flight.
3. **Phase 3** (Low velocity ratio): Predator moves faster than prey. Exploitation is emphasized using Levy flight.

Main Functions

[mpa()] Main optimization function implementing the MPA algorithm

[get_function_details()] Retrieve benchmark function parameters

Helper Functions

[levy()] Generate Levy flight random steps

[initialize_population()] Initialize search agent population

Benchmark Functions

The package includes 23 standard benchmark functions (F1-F23) for testing optimization algorithms. See `test-functions` for a complete list.

Getting Started

```
library(marinepredator)

# Basic optimization with the Sphere function
result <- mpa(
  SearchAgents_no = 30,
  Max_iter = 100,
  lb = -100, ub = 100,
  dim = 30,
  fobj = F01
)
print(result)
```

Author(s)

Maintainer: Marc Grossouvre <marc@grossouvre.fr> (R implementation) [translator]

Authors:

- Afshin Faramarzi (Original MATLAB code)
- Seyedali Mirjalili (Original MATLAB code)

References

Faramarzi, A., Heidarinejad, M., Mirjalili, S., & Gandomi, A. H. (2020). Marine Predators Algorithm: A Nature-inspired Metaheuristic. Expert Systems with Applications, 152, 113377. doi:10.1016/j.eswa.2020.113377

See Also

Useful links:

- GitHub repository: https://github.com/urbs-dev/marinepredator
- Report bugs: https://github.com/urbs-dev/marinepredator/issues

---

`F01` *Sphere Function (F01)*

Description

The Sphere function is a simple unimodal test function commonly used to evaluate the performance of optimization algorithms. It has a single global minimum at the origin.

Usage

```
F01(x)
```

Arguments

`x`	Numeric vector of input values.

Details

Formula:

$$f(x) = \sum_{i=1}^{n} x_i^2$$

Global minimum: $f(0, 0, ..., 0) = 0$

Characteristics:

- Type: Unimodal
- Separable: Yes
- Differentiable: Yes
- Convex: Yes
- Default bounds: $[-100, 100]^n$
- Default dimensions: 50

Value

Numeric scalar representing the function value.

See Also

`test-functions` for an overview of all test functions, `get_function_details` to retrieve function parameters.

Examples

```
F01(c(0, 0, 0))    # Returns 0 (global minimum)
F01(c(1, 2, 3))    # Returns 1 + 4 + 9 = 14
F01(rep(0, 50))    # Returns 0 in 50 dimensions
```

F02 *Sum of Absolute Values and Products (F02)*

Description

A unimodal test function that combines the sum of absolute values and the product of absolute values.

Usage

```
F02(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$$

Global minimum: $f(0, 0, ..., 0) = 0$

Characteristics:

- Type: Unimodal
- Separable: No (due to product term)
- Differentiable: No (at points where any x_i = 0)
- Convex: Yes
- Default bounds: $[-10, 10]^n$
- Default dimensions: 50

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F02(c(0, 0))      # Returns 0 (global minimum)
F02(c(1, 2))      # Returns |1| + |2| + |1|*|2| = 1 + 2 + 2 = 5
F02(c(-1, -2))    # Returns 1 + 2 + 2 = 5
```

F03 *Sum of Squared Cumulative Sums (F03)*

Description

A unimodal test function that calculates the sum of squared cumulative sums, also known as the Schwefel 2.22 variant.

Usage

```
F03(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$$

Global minimum: $f(0, 0, ..., 0) = 0$

Characteristics:

- Type: Unimodal
- Separable: No
- Differentiable: Yes
- Convex: Yes
- Default bounds: $[-100, 100]^n$
- Default dimensions: 50

This function is non-separable because each term depends on all previous variables, making it useful for testing algorithms' ability to handle variable dependencies.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F03(c(0, 0, 0))    # Returns 0 (global minimum)
F03(c(1, 2, 3))    # Returns 1^2 + (1+2)^2 + (1+2+3)^2 = 1 + 9 + 36 = 46
```

F04 *Maximum Absolute Value (F04)*

Description

A unimodal test function that returns the maximum absolute value in the input vector, also known as the Schwefel 2.21 function.

Usage

```
F04(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = \max_i |x_i|$$

Global minimum: $f(0, 0, ..., 0) = 0$

Characteristics:

- Type: Unimodal
- Separable: No
- Differentiable: No
- Convex: Yes
- Default bounds: $[-100, 100]^n$
- Default dimensions: 50

This function is particularly challenging because it only depends on the single variable with the largest absolute value, making gradient-based information less useful.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F04(c(0, 0, 0))    # Returns 0 (global minimum)
F04(c(-1, 2, -3))  # Returns 3
F04(c(5, -5, 5))   # Returns 5
```

F05 *Rosenbrock Function (F05)*

Description

The Rosenbrock function (also known as Rosenbrock's valley or banana function) is a classic test function with a narrow, curved valley. While often described as unimodal, it is technically multimodal for $n \geq 4$.

Usage

```
F05(x)
```

Arguments

x	Numeric vector of input values (minimum length 2).

Details

Formula:

$$f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

Global minimum: $f(1, 1, ..., 1) = 0$

Characteristics:

- Type: Unimodal (for n < 4), Multimodal (for n >= 4)
- Separable: No
- Differentiable: Yes
- Convex: No
- Default bounds: $[-30, 30]^n$
- Default dimensions: 50

The global minimum lies inside a long, narrow, parabolic-shaped flat valley. Finding the valley is trivial, but converging to the global minimum is difficult.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F05(c(1, 1))       # Returns 0 (global minimum)
F05(c(0, 0))       # Returns 1
F05(c(1, 1, 1, 1)) # Returns 0 (global minimum in 4D)
```

F06 *Step Function / Shifted Sphere (F06)*

Description

A shifted version of the Sphere function with the minimum at $(-0.5, -0.5, ..., -0.5)$, also known as the Step function.

Usage

```
F06(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$$

Global minimum: $f(-0.5, -0.5, ..., -0.5) = 0$

Characteristics:

- Type: Unimodal
- Separable: Yes
- Differentiable: Yes
- Convex: Yes
- Default bounds: $[-100, 100]^n$
- Default dimensions: 50

This function tests the algorithm's ability to find optima that are not located at the origin.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F06(c(-0.5, -0.5))       # Returns 0 (global minimum)
F06(c(0, 0))             # Returns 0.5
F06(rep(-0.5, 50))       # Returns 0 in 50 dimensions
```

F07 *Quartic Function with Noise (F07)*

Description

A unimodal test function with quartic terms and added uniform random noise. The noise makes this a stochastic function, useful for testing robustness.

Usage

```
F07(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = \sum_{i=1}^{n} i \cdot x_i^4 + \text{random}[0, 1)$$

Global minimum: $f(0, 0, ..., 0) \approx 0$ (stochastic, depends on noise)

Characteristics:

- Type: Unimodal (with noise)
- Separable: Yes (deterministic part)
- Differentiable: Yes
- Stochastic: Yes (random noise added)
- Default bounds: $[-1.28, 1.28]^n$
- Default dimensions: 50

The random noise component makes function evaluations non-deterministic, testing an algorithm's ability to handle noisy objective functions.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F07(c(0, 0, 0))  # Returns a value close to 0 (with some noise)
# Multiple calls may return different values due to noise:
replicate(5, F07(c(0, 0, 0)))
```

F08 *Schwefel Function (F08)*

Description

A multimodal test function with many local minima. The global minimum is geometrically distant from the next best local minima, making this function deceptive and challenging for optimization algorithms.

Usage

```
F08(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$$

Global minimum: $f(420.9687, ..., 420.9687) \approx -418.9829 \times n$

Characteristics:

- Type: Multimodal
- Separable: Yes
- Differentiable: Yes (except at x_i = 0)
- Default bounds: $[-500, 500]^n$
- Default dimensions: 50

The Schwefel function is deceptive in that the global minimum is geometrically distant from the next best local minima. This tests an algorithm's ability to escape local optima and explore widely.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F08(c(420.9687, 420.9687))  # Returns approximately -837.97 (near global minimum)
F08(c(0, 0))                # Returns 0
```

F09 *Rastrigin Function (F09)*

Description

A highly multimodal test function with many local minima arranged in a regular lattice pattern. The global minimum is at the origin.

Usage

```
F09(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Global minimum: $f(0, 0, ..., 0) = 0$

Characteristics:

- Type: Multimodal
- Separable: Yes
- Differentiable: Yes
- Number of local minima: $\approx 10^n$
- Default bounds: $[-5.12, 5.12]^n$
- Default dimensions: 50

The Rastrigin function is a typical example of non-linear multimodal function. The large number of local minima makes it difficult for optimization algorithms to find the global minimum.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F09(c(0, 0))    # Returns 0 (global minimum)
F09(c(1, 1))    # Returns approximately 2
F09(rep(0, 10)) # Returns 0 in 10 dimensions
```

F10 *Ackley Function (F10)*

Description

A widely used multimodal test function characterized by a nearly flat outer region with a large central hole at the origin. It poses a risk for optimization algorithms to be trapped in local minima.

Usage

```
F10(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

Global minimum: $f(0, 0, ..., 0) = 0$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Default bounds: $[-32, 32]^n$
- Default dimensions: 50

The Ackley function has an exponential term covering its surface with numerous local minima. The function poses a risk of premature convergence for hill-climbing algorithms.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F10(c(0, 0))  # Returns approximately 0 (global minimum)
F10(c(1, 1))  # Returns approximately 3.6
```

F11 *Griewank Function (F11)*

Description

A multimodal test function with many regularly distributed local minima. The complexity increases with the number of dimensions.

Usage

```
F11(x)
```

Arguments

x	Numeric vector of input values.

Details

Formula:

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Global minimum: $f(0, 0, ..., 0) = 0$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Default bounds: $[-600, 600]^n$
- Default dimensions: 50

The Griewank function has a product term that introduces interdependence among the variables. As dimension increases, the function becomes more difficult to optimize.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F11(c(0, 0))  # Returns 0 (global minimum)
F11(c(1, 1))  # Returns approximately 0.007
```

F12 *Penalized Function 1 (F12)*

Description

A multimodal test function with penalty terms that create a complex search landscape. Uses the Ufun helper function.

Usage

```
F12(x)
```

Arguments

x	Numeric vector of input values (minimum length 2).

Details

Formula:

$$f(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2(1+10\sin^2(\pi y_{i+1})) + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$$

where $y_i = 1 + (x_i + 1)/4$ and $u(x, a, k, m)$ is a penalty function.

Global minimum: $f(-1, -1, ..., -1) = 0$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes (in the interior)
- Penalized: Yes (boundary penalty)
- Default bounds: $[-50, 50]^n$

- Default dimensions: 50

The penalty term $u(x, a, k, m)$ adds a cost when variables exceed the threshold a, helping to constrain solutions to a feasible region.

Value

Numeric scalar representing the function value.

See Also

`test-functions` for an overview of all test functions, `get_function_details` to retrieve function parameters, `Ufun` for the penalty function.

Examples

```
F12(c(-1, -1, -1))  # Returns approximately 0 (near global minimum)
F12(c(0, 0, 0))     # Returns a small positive value
```

`F13` *Penalized Function 2 (F13)*

Description

A multimodal test function with different penalty terms than F12, creating a complex search landscape. Uses the `Ufun` helper function.

Usage

```
F13(x)
```

Arguments

x	Numeric vector of input values (minimum length 2).

Details

Formula:

$$f(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1})) + (x_n - 1)^2 (1 + \sin^2(2\pi x_n)) \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$$

where $u(x, a, k, m)$ is a penalty function.

Global minimum: $f(1, 1, ..., 1) = 0$

Characteristics:

- Type: Multimodal

- Separable: No
- Differentiable: Yes (in the interior)
- Penalized: Yes (boundary penalty)
- Default bounds: $[-50, 50]^n$
- Default dimensions: 50

This function differs from F12 in the sinusoidal terms and penalty threshold, resulting in different landscape characteristics.

Value

Numeric scalar representing the function value.

See Also

`test-functions` for an overview of all test functions, `get_function_details` to retrieve function parameters, `Ufun` for the penalty function.

Examples

```
F13(c(1, 1, 1))  # Returns approximately 0 (near global minimum)
F13(c(0, 0, 0))  # Returns a small positive value
```

F14 *Shekel's Foxholes Function (F14)*

Description

A multimodal test function with 25 local minima of different depths, arranged in a grid pattern. Fixed dimension of 2.

Usage

```
F14(x)
```

Arguments

x	Numeric vector of length 2 (2-dimensional input).

Details

Formula:

$$f(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right)^{-1}$$

where a_{ij} are predefined constants forming a 5x5 grid.

Global minimum: $f(-32, -32) \approx 0.998$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 2
- Number of local minima: 25
- Default bounds: $[-65.536, 65.536]^2$

The 25 minima are located at points from a 5x5 grid with coordinates $(-32, -16, 0, 16, 32)$ in each dimension.

Value

Numeric scalar representing the function value.

See Also

`test-functions` for an overview of all test functions, `get_function_details` to retrieve function parameters.

Examples

```
F14(c(-32, -32))  # Returns approximately 0.998 (global minimum)
F14(c(0, 0))      # Returns approximately 10
```

F15 *Kowalik Function (F15)*

Description

A multimodal test function used for testing optimization algorithms, based on fitting experimental data. Fixed dimension of 4.

Usage

```
F15(x)
```

Arguments

x	Numeric vector of length 4 (4-dimensional input).

Details

Formula:

$$f(x) = \sum_{i=1}^{11} \left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right)^2$$

where a_i and b_i are predefined constants from experimental data.

Global minimum: $f(0.1928, 0.1908, 0.1231, 0.1358) \approx 0.0003075$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 4
- Default bounds: $[-5, 5]^4$

This function is derived from a curve-fitting problem and has several local minima near the global minimum.

Value

Numeric scalar representing the function value.

See Also

`test-functions` for an overview of all test functions, `get_function_details` to retrieve function parameters.

Examples

```
F15(c(0.1928, 0.1908, 0.1231, 0.1358))  # Returns approximately 0.0003
F15(c(0, 0, 0, 0))  # Returns a larger value
```

F16 *Six-Hump Camel Back Function (F16)*

Description

A multimodal test function with six local minima, two of which are global. Fixed dimension of 2.

Usage

```
F16(x)
```

Arguments

x	Numeric vector of length 2 (2-dimensional input).

Details

Formula:

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4$$

Global minimum: $f(\pm 0.0898, \mp 0.7126) \approx -1.0316$

There are two global minima at approximately $(0.0898, -0.7126)$ and $(-0.0898, 0.7126)$.

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 2
- Number of local minima: 6
- Number of global minima: 2
- Default bounds: $[-5, 5]^2$

The function has a shape resembling a camel's back with six humps (minima).

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F16(c(0.0898, -0.7126))   # Returns approximately -1.0316 (global minimum)
F16(c(-0.0898, 0.7126))   # Returns approximately -1.0316 (global minimum)
F16(c(0, 0))              # Returns 0
```

F17 *Branin Function (F17)*

Description

A multimodal test function with three global minima, commonly used for testing optimization algorithms. Fixed dimension of 2.

Usage

```
F17(x)
```

Arguments

x	Numeric vector of length 2 (2-dimensional input).

Details

Formula:

$$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

Global minimum: $f^* \approx 0.397887$ at three locations:

- $(-\pi, 12.275)$
- $(\pi, 2.275)$
- $(9.42478, 2.475)$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 2
- Number of global minima: 3
- Default bounds: $x_1 \in [-5, 10]$, $x_2 \in [0, 15]$

The Branin function is often used as a benchmark because it has three global minima that are well-separated.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F17(c(-pi, 12.275))      # Returns approximately 0.398 (global minimum)
F17(c(pi, 2.275))        # Returns approximately 0.398 (global minimum)
F17(c(9.42478, 2.475))   # Returns approximately 0.398 (global minimum)
```

F18 *Goldstein-Price Function (F18)*

Description

A multimodal test function with four local minima and one global minimum. Fixed dimension of 2.

Usage

```
F18(x)
```

Arguments

x	Numeric vector of length 2 (2-dimensional input).

Details

Formula:

$f(x) = (1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2))\times(30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27$

Global minimum: $f(0, -1) = 3$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 2
- Number of local minima: 4
- Default bounds: $[-2, 2]^2$

The Goldstein-Price function has a complex landscape with the global minimum surrounded by local minima of increasing value.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters.

Examples

```
F18(c(0, -1))  # Returns 3 (global minimum)
F18(c(0, 0))   # Returns 600
```

F19 *Hartmann 3D Function (F19)*

Description

A multimodal test function with 4 local minima in 3 dimensions. Fixed dimension of 3.

Usage

```
F19(x)
```

Arguments

x	Numeric vector of length 3 (3-dimensional input).

Details

Formula:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$$

where a_{ij}, c_i, and p_{ij} are predefined constants.

Global minimum: $f(0.114614, 0.555649, 0.852547) \approx -3.86278$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 3
- Number of local minima: 4
- Default bounds: $[0, 1]^3$

The Hartmann functions are a family of multimodal test functions commonly used in optimization benchmarks.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters, F20 for the 6D version.

Examples

```
F19(c(0.114614, 0.555649, 0.852547))  # Returns approximately -3.86278
F19(c(0.5, 0.5, 0.5))  # Returns a value > -3.86
```

F20 *Hartmann 6D Function (F20)*

Description

A multimodal test function with 6 local minima in 6 dimensions. Fixed dimension of 6.

Usage

```
F20(x)
```

Arguments

x	Numeric vector of length 6 (6-dimensional input).

Details

Formula:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$$

where a_{ij}, c_i, and p_{ij} are predefined constants.

Global minimum: $f(0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573) \approx -3.32237$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 6
- Number of local minima: 6
- Default bounds: $[0, 1]^6$

The Hartmann 6D function is more challenging than the 3D version due to the higher dimensionality.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters, F19 for the 3D version.

Examples

```
F20(c(0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573))
# Returns approximately -3.32237
```

F21 *Shekel 5 Function (F21)*

Description

A multimodal test function from the Shekel family with 5 local minima. Fixed dimension of 4.

Usage

```
F21(x)
```

Arguments

x	Numeric vector of length 4 (4-dimensional input).

Details

Formula:

$$f(x) = -\sum_{i=1}^{5} \frac{1}{(x - a_i)^T (x - a_i) + c_i}$$

where a_i are 4-dimensional vectors and c_i are scalars.

Global minimum: $f(4, 4, 4, 4) \approx -10.1532$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 4
- Number of local minima: 5
- Default bounds: $[0, 10]^4$

The Shekel functions are parameterized by the number of terms m (here m=5). As m increases, the function becomes more challenging.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters, F22 for Shekel 7, F23 for Shekel 10.

Examples

```
F21(c(4, 4, 4, 4))  # Returns approximately -10.15 (near global minimum)
F21(c(0, 0, 0, 0))  # Returns a value close to 0
```

F22 *Shekel 7 Function (F22)*

Description

A multimodal test function from the Shekel family with 7 local minima. Fixed dimension of 4.

Usage

```
F22(x)
```

Arguments

x	Numeric vector of length 4 (4-dimensional input).

Details

Formula:

$$f(x) = -\sum_{i=1}^{7} \frac{1}{(x - a_i)^T (x - a_i) + c_i}$$

where a_i are 4-dimensional vectors and c_i are scalars.

Global minimum: $f(4, 4, 4, 4) \approx -10.4029$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 4
- Number of local minima: 7
- Default bounds: $[0, 10]^4$

The Shekel 7 function has more local minima than Shekel 5, making it slightly more challenging.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters, F21 for Shekel 5, F23 for Shekel 10.

Examples

```
F22(c(4, 4, 4, 4))  # Returns approximately -10.40 (near global minimum)
F22(c(0, 0, 0, 0))  # Returns a value close to 0
```

F23 *Shekel 10 Function (F23)*

Description

A multimodal test function from the Shekel family with 10 local minima. Fixed dimension of 4.

Usage

```
F23(x)
```

Arguments

x	Numeric vector of length 4 (4-dimensional input).

Details

Formula:

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(x - a_i)^T (x - a_i) + c_i}$$

where a_i are 4-dimensional vectors and c_i are scalars.

Global minimum: $f(4, 4, 4, 4) \approx -10.5364$

Characteristics:

- Type: Multimodal
- Separable: No
- Differentiable: Yes
- Fixed dimension: 4
- Number of local minima: 10
- Default bounds: $[0, 10]^4$

The Shekel 10 function is the most challenging of the Shekel family due to having the most local minima.

Value

Numeric scalar representing the function value.

See Also

test-functions for an overview of all test functions, get_function_details to retrieve function parameters, F21 for Shekel 5, F22 for Shekel 7.

Examples

```
F23(c(4, 4, 4, 4))  # Returns approximately -10.54 (near global minimum)
F23(c(0, 0, 0, 0))  # Returns a value close to 0
```

---

get_function_details *Get Function Details*

Description

Retrieves the details (bounds, dimension, and function object) for a specific benchmark test function.

Usage

```
get_function_details(function_name)
```

Arguments

function_name	Name of the test function (e.g., "F01", "F02", ..., "F23")

Value

A list containing:

lb	Lower bounds for the function
ub	Upper bounds for the function
dim	Number of dimensions
fobj	The objective function

Examples

```
# Get details for the Sphere function (F01)
details <- get_function_details("F01")
str(details)
```

initialize_population *Initialize Population*

Description

Initializes the first population of search agents (prey) for the Marine Predators Algorithm.

Usage

```
initialize_population(SearchAgents_no, dim, ub, lb)
```

Arguments

SearchAgents_no	Number of search agents
dim	Number of dimensions
ub	Upper bounds for each dimension
lb	Lower bounds for each dimension

Value

A matrix of initialized positions

Examples

```
# Initialize a population of 25 agents in 30 dimensions with bounds [-100, 100]
population <- initialize_population(25, 30, 100, -100)
```

levy *Levy Flight Random Step Generator*

Description

Generates Levy flight random steps for optimization algorithms. Levy flight is characterized by a heavy-tailed power-law distribution and is used to model the movement patterns of marine predators, allowing for occasional long jumps that help escape local optima.

Usage

```
levy(n, m, beta)
```

Arguments

n	Number of rows (typically the number of search agents).
m	Number of columns (typically the number of dimensions).
beta	Power law exponent controlling the tail heaviness of the distribution. Must satisfy 1 < beta < 2. Lower values produce heavier tails (more frequent long jumps). The value 1.5 is commonly used.

Details

The Levy flight step is computed using Mantegna's algorithm:

$$L = u/|v|^{1/\beta}$$

where $u \sim N(0, \sigma_u^2)$ and $v \sim N(0, 1)$, with:

$$\sigma_u = \left(\frac{\Gamma(1+\beta) \cdot \sin(\pi\beta/2)}{\Gamma((1+\beta)/2) \cdot \beta \cdot 2^{(\beta-1)/2}} \right)^{1/\beta}$$

Levy flights are used in MPA during Phase 2 (for half the population) and Phase 3 (for all agents) to balance exploration and exploitation.

Value

A numeric matrix of dimension n x m containing Levy flight step values. Each element represents a random step drawn from the Levy distribution.

References

Mantegna, R. N. (1994). Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Physical Review E, 49(5), 4677.

Yang, X. S., & Deb, S. (2013). Multiobjective cuckoo search for design optimization. Computers & Operations Research, 40(6), 1616-1624.

See Also

[mpa()] for the main algorithm that uses Levy flights.

Examples

```
# Generate 10 Levy flight steps in 2 dimensions with beta = 1.5
steps <- levy(10, 2, 1.5)
dim(steps)  # 10 x 2

# Visualize the distribution of Levy steps
steps_1d <- levy(1000, 1, 1.5)
hist(steps_1d, breaks = 50, main = "Levy Flight Distribution")
```

mpa *Marine Predators Algorithm (MPA)*

Description

Implementation of the Marine Predators Algorithm (MPA) in R. MPA is a nature-inspired optimization algorithm that follows the rules governing optimal foraging strategy and encounter rate policy between predator and prey in marine ecosystems.

Usage

```
mpa(SearchAgents_no, Max_iter, lb, ub, dim, fobj, logFile = NULL, ...)
```

Arguments

SearchAgents_no	Number of search agents (predators). At least 2. Typical values range from 20 to 50.
Max_iter	Maximum number of iterations. At least 3. Typical values range from 100 to 500 depending on problem complexity.
lb	Lower bounds for each dimension. Can be a single value (applied to all dimensions) or a vector of length dim.
ub	Upper bounds for each dimension. Can be a single value (applied to all dimensions) or a vector of length dim.
dim	Number of dimensions (decision variables) in the optimization problem.
fobj	Objective function to minimize. Must accept a numeric vector of length dim and return a single numeric value.
logFile	A path for logging (text file). Defaulted to NULL. If NULL, no logging is performed.
...	Additional arguments. Currently supports prefix for log message prefixing. See Details.

Details

This is a **minimization** algorithm. To maximize a function, negate its output (e.g., use function(x) -f(x) instead of f).

Value

An object of class mpa_result, which is a list containing:

Top_predator_fit	Best fitness value found (numeric scalar)
Top_predator_pos	Best position found (numeric vector of length dim)
Convergence_curve	Convergence curve over iterations (numeric vector of length Max_iter)

Algorithm Phases

The MPA algorithm operates in three distinct phases based on the iteration count:

Phase 1 (iterations 0 to Max_iter/3) High velocity ratio - The prey moves faster than the predator. Exploration is emphasized using Brownian motion. This phase promotes global search across the solution space.

Phase 2 (iterations Max_iter/3 to 2*Max_iter/3) Unit velocity ratio - Predator and prey move at similar speeds. The population is split: half uses Brownian motion (exploitation), half uses Levy flight (exploration). This balances exploration and exploitation.

Phase 3 (iterations 2*Max_iter/3 to Max_iter) Low velocity ratio - The predator moves faster than the prey. Levy flight is used for all agents, focusing on exploitation around the best solution found.

Internal Parameters

The algorithm uses two internal parameters that are not exposed:

FADs Fish Aggregating Devices effect parameter, set to 0.2. Controls the probability of applying the FADs effect which helps escape local optima.

P Prey movement probability, set to 0.5. Controls the step size scaling factor during position updates.

Memory Mechanism

MPA implements a memory mechanism (Marine Memory) that preserves the best positions found by each agent. If a new position has worse fitness than the previous one, the agent reverts to its previous position.

Additional Arguments

The . . . parameter currently accepts:

prefix A character string to prefix log messages. Defaults to empty string if not provided.

References

Faramarzi, A., Heidarinejad, M., Mirjalili, S., & Gandomi, A. H. (2020). Marine Predators Algorithm: A Nature-inspired Metaheuristic. Expert Systems with Applications, 152, 113377. doi:10.1016/j.eswa.2020.113377

See Also

[get_function_details()] for benchmark functions, [levy()] for Levy flight implementation, [initialize_population()] for population initialization.

Examples

```
# Basic usage with the Sphere function (F01)
result <- mpa(
  SearchAgents_no = 25, Max_iter = 100, lb = -100, ub = 100,
  dim = 30, fobj = F01
)
print(result)

# Using different bounds per dimension
result2 <- mpa(
  SearchAgents_no = 20, Max_iter = 50,
  lb = c(-5, 0), ub = c(10, 15),
  dim = 2, fobj = F17
)

# Maximization example (negate the objective function)
maximize_f <- function(x) -sum(x^2)
result3 <- mpa(
  SearchAgents_no = 20, Max_iter = 50,
  lb = -10, ub = 10, dim = 5,
  fobj = function(x) -maximize_f(x)
)
# The actual maximum value is -result3$Top_predator_fit
```

---

print.mpa_result *Print method for MPA results*

Description

Prints a summary of the Marine Predators Algorithm optimization results.

Usage

```
## S3 method for class 'mpa_result'
print(x, ...)
```

Arguments

x	An object of class mpa_result returned by [mpa()].
...	Additional arguments (currently unused).

Value

Invisibly returns the input object x.

Examples

```
result <- mpa(SearchAgents_no = 10, Max_iter = 20, lb = -10, ub = 10,
              dim = 5, fobj = F01)
print(result)
```

`test-functions` *Test Functions for Optimization*

Description

Collection of benchmark test functions for evaluating optimization algorithms. These functions are commonly used in the literature to test the performance of metaheuristic optimization algorithms.

Details

The following test functions are implemented:

Unimodal Functions:

- **F01** - Sphere function: Simple quadratic function with a single minimum at the origin. `F01`
- **F02** - Sum of absolute values and products: Combines sum and product of absolute values. `F02`
- **F03** - Sum of squared terms: Sum of squared cumulative sums. `F03`
- **F04** - Maximum absolute value: Returns the maximum absolute value in the vector. `F04`
- **F05** - Rosenbrock function: Classic test function with a banana-shaped valley. `F05`
- **F06** - Shifted sphere function: Sphere function shifted to (-0.5, -0.5, ...). `F06`
- **F07** - Quartic function with noise: Quartic terms with added random noise. `F07`

Multimodal Functions:

- **F08** - Schwefel function: Multimodal function with many local minima. `F08`
- **F09** - Rastrigin function: Many local minima arranged in a regular pattern. `F09`
- **F10** - Ackley function: Global minimum at origin with many local minima. `F10`
- **F11** - Griewank function: Many local minima regularly distributed. `F11`
- **F12** - Penalized function 1: Complex landscape with penalty terms. `F12`
- **F13** - Penalized function 2: Different penalty terms creating complex landscape. `F13`
- **F14** - Shekel's Foxholes: 25 minima of different depths. `F14`
- **F15** - Kowalik function: Multimodal function for testing optimization. `F15`
- **F16** - Six-Hump Camel Back: Six local minima, two global. `F16`
- **F17** - Branin function: Commonly used for testing optimization algorithms. `F17`
- **F18** - Goldstein-Price function: Four local minima. `F18`
- **F19** - Hartman 3D function: 3 dimensions with multiple local minima. `F19`
- **F20** - Hartman 6D function: 6 dimensions with multiple local minima. `F20`
- **F21** - Shekel 5 function: Based on Shekel's family with 5 terms. `F21`
- **F22** - Shekel 7 function: Based on Shekel's family with 7 terms. `F22`
- **F23** - Shekel 10 function: Based on Shekel's family with 10 terms. `F23`

References

Faramarzi, A., Heidarinejad, M., Mirjalili, S., & Gandomi, A. H. (2020). Marine Predators Algorithm: A Nature-inspired Metaheuristic. Expert Systems with Applications, 113377. DOI: 10.1016/j.eswa.2020.113377

See Also

`F01` through `F23` for individual function documentation. `get_function_details` to retrieve function details programmatically.

Examples

```
# Get details for a specific function
details <- get_function_details("F01")

# Use a test function directly
F01(c(0, 0, 0))  # Returns 0 (global minimum)

# List all available test functions
available_functions <- c("F01", "F02", "F03", "F04", "F05", "F06", "F07", "F08", "F09", "F10",
                         "F11", "F12", "F13", "F14", "F15", "F16", "F17", "F18", "F19", "F20",
                            "F21", "F22", "F23")
```

Ufun *Penalty Function for Penalized Test Functions*

Description

Helper function used in penalized test functions (F12, F13). This function applies a penalty when variable values exceed specified bounds.

Usage

```
Ufun(x, a, k, m)
```

Arguments

`x`	Numeric vector of input values to penalize.
`a`	Threshold value. Penalty is applied when $\|x_i\| > a$.
`k`	Penalty coefficient controlling the penalty magnitude.
`m`	Exponent for the penalty term.

Details

Formula:

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \leq x_i \leq a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases}$$

The penalty function is zero within the interval $[-a, a]$ and grows polynomially outside this region. This is used to softly constrain optimization to a bounded region without hard constraints.

Value

Numeric vector of penalty values (same length as `x`).

See Also

`F12`, `F13` for functions that use this helper.

Index