Package 'soma'

May 2, 2022

Version 1.2.0

Date 2022-05-01

Title General-Purpose Optimisation with the Self-Organising Migrating Algorithm

Author Jon Clayden

Maintainer Jon Clayden <code@clayden.org>

Depends R (>= 2.5.0)

Imports reportr (>= 1.3.0)

Suggests tinytest, covr, shades

Description An R implementation of the Self-Organising Migrating Algorithm, a general-purpose, stochastic optimisation algorithm. The approach is similar to that of genetic algorithms, although it is based on the idea of a series of ``migrations'' by a fixed set of individuals, rather than the development of successive generations. It can be applied to any cost-minimisation problem with a bounded parameter space, and is robust to local minima.

License GPL-2

URL https://github.com/jonclayden/soma/

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-02 08:40:05 UTC

R topics documented:

all2one 2
soma 4

Index **7**

`all2one` *Options for the available SOMA variants*

Description

These functions generate option lists (and provide defaults) for the SOMA algorithm variants available in the package, which control how the algorithm will proceed and when it will terminate. Each function corresponds to a different top-level strategy, described in a different reference.

Usage

```
all2one(
  populationSize = 10L,
  nMigrations = 20L,
  pathLength = 3,
  stepLength = 0.11,
  perturbationChance = 0.1,
  minAbsoluteSep = 0,
  minRelativeSep = 0.001
)

t3a(
  populationSize = 30L,
  nMigrations = 20L,
  nSteps = 45L,
  migrantPoolSize = 10L,
  leaderPoolSize = 10L,
  nMigrants = 4L,
  minAbsoluteSep = 0,
  minRelativeSep = 0.001
)

pareto(
  populationSize = 100L,
  nMigrations = 20L,
  nSteps = 10L,
  perturbationFrequency = 1,
  stepFrequency = 1,
  minAbsoluteSep = 0,
  minRelativeSep = 0.001
)
```

Arguments

`populationSize`	The number of individuals in the population. It is recommended that this be somewhat larger than the number of parameters being optimised over, and it should not be less than 2. The default varies by strategy.

`nMigrations` The maximum number of migrations to complete.

`pathLength` The distance towards the leader that individuals may migrate. A value of 1 corresponds to the leader's position itself, and values greater than one (recommended) allow for some overshoot.

`stepLength` The granularity at which potential steps are evaluated. It is recommended that the `pathLength` not be a whole multiple of this value.

`perturbationChance`
The probability that individual parameters are changed on any given step.

`minAbsoluteSep` The smallest absolute difference between the maximum and minimum cost function values. If the difference falls below this minimum, the algorithm will terminate. The default is 0, meaning that this termination criterion will never be met.

`minRelativeSep` The smallest relative difference between the maximum and minimum cost function values. If the difference falls below this minimum, the algorithm will terminate.

`nSteps` The number of candidate steps towards the leader per migrating individual. This option is used instead of `pathLength` and `stepLength` under the T3A and Pareto strategies, where the step length is variable.

`migrantPoolSize, leaderPoolSize`
The number of randomly selected individuals to include in the migrant and leader pools, respectively, under the T3A strategy.

`nMigrants` The number of individuals that will migrate, at each migration, under the T3A strategy.

`perturbationFrequency, stepFrequency`
Scale factors affecting how rapidly the perturbation probability and step sizes fluctuate under the Pareto strategy.

Details

All To One (the `all2one` function) is the original SOMA strategy. At each "migration", the cost function is evaluated for all individuals in the population, and the one with the lowest value is designated the "leader". All other individuals migrate towards the leader's position in some or all dimensions of the parameter space, with a fixed probability of perturbation in each dimension. Each migration is evaluated against the cost function at several points on the line towards the leader, and the location with the lowest value becomes the individual's starting position for the next migration.

The Team To Team Adaptive (T3A) strategy (Diep, 2019) differs in that only a random subset of individuals are selected into a migrant pool and a leader pool for any given migration. A subset of most optimal migrants are then migrated towards the single most optimal individual from the leader pool. The perturbation probability and step length along the trajectory towards the leader also vary according to formulae given by the strategy author as the algorithm progresses through the migrations.

In the Pareto strategy (Diep et al., 2019), all individuals are sorted by cost function value at the start of each migration. The leader is selected randomly from the top 4% (20% of 20%) of most optimal individuals, and a single migrant is chosen at random from between the 20th and the 36th percentiles of the population (the top 20% of the bottom 80%). The perturbation probability and the step length again vary across migrations, but this time in a sinusoidal fashion, and the migrant is updated in all dimensions, but some more slowly than others.

Value

A list of class `"soma.options"`.

Author(s)

Jon Clayden <code@clayden.org>

References

I. Zelinka (2004). SOMA - self-organizing migrating algorithm. In G.C. Onwubolu & B.V. Babu, eds, New optimization techniques in engineering. Volume 141 of "Studies in Fuzziness and Soft Computing", pp. 167-217. Springer.

Q.B. Diep (2019). Self-Organizing Migrating Algorithm Team To Team Adaptive – SOMA T3A. In proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 1182-1187. IEEE.

Q.B. Diep, I. Zelinka & S. Das (2019). Pareto-Based Self-Organizing Migrating Algorithm. Mendel 25(1):111-120.

---

soma *The Self-Organising Migrating Algorithm*

Description

The Self-Organising Migrating Algorithm (SOMA) is a general-purpose, stochastic optimisation algorithm. The approach is similar to that of genetic algorithms, although it is based on the idea of a series of "migrations" by a fixed set of individuals, rather than the development of successive generations. It can be applied to any cost-minimisation problem with a bounded parameter space, and is robust to local minima.

Usage

```
soma(costFunction, bounds, options = list(), init = NULL, ...)

bounds(min, max)

## S3 method for class 'soma'
plot(x, y = NULL, add = FALSE, ...)
```

Arguments

`costFunction`	A cost function which takes a numeric vector of parameters as its first argument, and returns a numeric scalar representing the associated cost value.
`bounds`	A list with elements `min` and `max`, each a numeric vector giving the upper and lower bounds for each parameter, respectively.
`options`	A list of options for the SOMA algorithm itself, usually generated by functions like `all2one`.

`init`	An optional matrix giving the starting population's positions in parameter space, one per column. If omitted, initialisation is random (as is usual for SOMA), but specifying a starting state can be helpful when running the algorithm in stages or investigating the consistency of solutions.
`...`	Additional parameters to `costFunction` (for `soma`) or the default plotting method (for `plot.soma`).
`min, max`	Vectors of minimum and maximum bound values for each parameter to the `costFunction`.
`x`	An object of class `"soma"`.
`y`	Ignored.
`add`	If `TRUE`, add to an existing plot canvas.

Value

A list of class `"soma"`, containing the following elements.

leader The index of the "leader", the individual in the population with the lowest cost.

population A matrix whose columns give the parameter values for each individual in the population at convergence.

cost A vector giving the cost function values for each individual at convergence.

history A vector giving the cost of the leader for each migration during the optimisation. This should be nonincreasing.

migrations The number of migrations completed.

evaluations The number of times the `costFunction` was evaluated.

A `plot` method is available for this class, which shows the history of leader cost values during the optimisation.

Author(s)

R implementation by Jon Clayden <code@clayden.org>.

References

I. Zelinka (2004). SOMA - self-organizing migrating algorithm. In G.C. Onwubolu & B.V. Babu, eds, New optimization techniques in engineering. Volume 141 of "Studies in Fuzziness and Soft Computing", pp. 167-217. Springer.

See Also

`soma.options` for setting options. `optim` implements other general-purpose optimisation methods.

Examples

```
# Rastrigin's function, which contains many local minima
rastrigin <- function (x) 10 * length(x) + sum(x^2 - 10 * cos(2*pi*x))

# Find the global minimum over the range -5 to 5 in each parameter
x <- soma(rastrigin, bounds(c(-5,-5), c(5,5)))

# Find the location of the leader - should be near the true minimum of c(0,0)
print(x$population[,x$leader])

# Plot the cost history of the leaders
plot(x)
```

Index

all2one, 2, *4*

bounds *(*soma*)*, 4

optim, *5*

pareto *(*all2one*)*, 2
plot.soma *(*soma*)*, 4

soma, 4
soma.options, *5*
soma.options *(*all2one*)*, 2

t3a *(*all2one*)*, 2