Package 'EAS'

April 20, 2012

Type Package

Title The Evolutionary Annealing-Simplex Method

Version 0.0-1

Date 2012-04-08

Author Panagiotis Kossieris <pankoss@hotmail.com>, with Hristos Tyralis <montchrister@gmail.com>, Demetris Koutsoyiannis <D.Koutsoyiannis@itia.ntua.gr>, and Andreas Efstratiadis <A.Efstratiadis@itia.ntua.gr>.

Maintainer Panagiotis Kossieris <pankoss@hotmail.com>

Description EAS package includes an evolutionary annealing-simplex algorithm for global optimization problems.

License GPL (version 2 or later)

URL http://www.itia.ntua.gr/, http://itia.ntua.gr/en/docinfo/524/

R topics documented:

`eas` *The Evalutionary Annealing-Simplex Method*

Description

An enhanced version of the evolutionary annealing-simplex method for global optimization.

Usage

```
eas( n,m,xmin,xmax,xlow,xup,fn,maxeval=1500,ftol=1.e-07,ratio=0.99,pmut=0.9,
     beta=2,maxclimbs=5 )
```

Arguments

n	An integer that specifies the problem dimension (number of control variables).
m	A positive integer, greater than n, that specifies the population size (m >= n + 1).
xmin	A vector, with length equal to n, that defines the interior lower parameter bounds (feasible space of initial population).
xmax	A vector, with length equal to n, that defines the interior upper parameter bounds (feasible space of initial population).
xlow	A vector, with length equal to n, that defines the exterior lower parameter bounds.
xup	A vector, with length equal to n, that defines the exterior upper parameter bounds.
fn	Objective function that is to be optimized. A vector function that takes a real vector as argument and returns the value of the function at that point.
maxeval	A positive integer that specifies the maximum number of function evaluations. Default is maxeval=1500. Suggested value is maxeval>100*n.
ftol	A positive number that specifies the fractional convergence tolerance to be achieved in the function value. Default is ftol=1.e-07.
ratio	A positive number, typically between 0.80-0.99, that specifies the fraction of temperature reduction, when a local minimum is found. Default is ratio=0.99.
pmut	A positive number, between 0.5-0.95, that specifies the probability of accepting an offspring generated via mutation. Default is pmut=0.9. Higher values are suggested for very hard problems, when it is essential to increase randomness.
beta	A positive integer,greater than 1, that specifies the annealing schedule parameter. Default is beta=2.
maxclimbs	A positive integer, typically between 3-5, that specifies the maximum number of uphill steps. Default is maxclimbs=5.

Details

The evalutionary annealing-simplex algorithm (2002), for solving global optimisation problems have been proposed by Efstratiadis and Koutsoyiannis (2002). The evalutionary annealing-simplex algorithm (eas) is a probabilistic heuristic global optimization technique, combining the robustness of simulated annealing in rough response surface, with the efficiency of hill-climbing methods in convex areas.

During one generation, the population evolves as follows: First, a simplex-based pattern is formulated, using random sampling. Next, a candidate individual is selected to die, according to a modified objective function of the form:

$g(\mathbf{x}) = f(\mathbf{x}) + uT$

where f is the original objective function, T is the current "temperature" and u is a random number from the uniform distribution. The temperature is gradually reduced, according to an appropriate annealing cooling schedule, automatically adapted during the evolution. Consequently, the probability of replacing individuals with poor performance increases, since the procedure gradually moves from a random walk to a local search.

The recombination operator is based on the well-known downhill simplex transitions (Nelder and Mead, 1965). According to the relatives values of the objective function at the vertices, the simplex is reflected, expanded, contracted or shrinks, where quasi-stochastic sclae factors are employed instead of constant ones. To ensure more flexibility, additional transformations are introduced, namely multiple expansion towards the direction of reflection, when a downhill path (i.e., the gradient of

the function) is located, and similar expansions but on the opposite (uphill) direction, in order to escape from the nearest local minimum. If any of the above transitions improves the function value, the new individual is generated through nutation. The related oerator employs a random pertubation scheme outside of the usual range of the population, as determined on the basis of the average and standard deviation values of its coordinates.

R adaption, with some modifications, by Panagiotis Kossieris and Hristos Tyralis, from the original Pascal-Delphi code by Andreas Efstratiadis (2007).

In the current version, the calculation of the reflection point is changed, and the simplex is not reflected toward the direction of the geometrical centroid but towards a weighted centroid, where the weigths are assigned on the basis of the objective fuction value. This allowes for a more accurate estimator of the gradient, which ensures a much more efficient search in multidimensional feasible spaces.

For more details, the reader may refer either to the original work of Efstratiadis and Koutsoyiannis (2002).

Value

A list with the following components:

`bestpar`	A vector containing the best set of parameters found.
`bestval`	The value of `fn` corresponding to `bestpar`.
`nfeval`	Number of function `fn` evaluations.
`niter`	Number of iterations taken by algorithm.
`ftolpop`	Fractional convergence tolerance of population generated at the last iteration.
`pop`	The population generated at the last iteration.

Author(s)

Kossieris Panagiotis <pankoss@hotmail.com>, with Hristos Tyralis <montchrister@gmail.com> and Andreas Efstratiadis <A.Efstratiadis@itia.ntua.gr>.

References

The methodology of **the evolutionary annealing-simplex optimization method** and details of its application are described in:

- Efstratiadis, A., and D. Koutsoyiannis, An evolutionary annealing-simplex algorithm for global optimisation of water resource systems, Proceedings of the Fifth International Conference on Hydroinformatics, Cardiff, UK, 1423-1428, International Water Association, 2002. (http://itia.ntua.gr/en/docinfo/524/)
- Rozos, E., A. Efstratiadis, I. Nalbantis, and D. Koutsoyiannis, Calibration of a semi-distributed model for conjunctive simulation of surface and groundwater flows, Hydrological Sciences Journal, 49 (5), 819-842, 2004. (http://itia.ntua.gr/en/docinfo/630/)

Find the original Pascal-Delphi code of optimisation algorithms on http://itia.ntua.gr/en/docinfo/524/.

Examples

```
# Random Parameter Bartlett-Lewis model equations

# modelled mean

modmean <- function(a,l,v,k,f,mx,h=1) {

x <- (h*l*mx*v*(1+k/f))/(a-1)

return(x)

}

# modelled variance

modvar <- function(a,l,v,k,f,mx,h=1) {

A <- (2*l*(1+k/f)*(mx^2)*(v^a))/((f^2)*((f^2)-1)*(a-1)*(a-2)*(a-3))

B <- (2*(f^2)-2+k*f)*(f^2)*((a-3)*h*(v^(2-a))-(v^(3-a))+((v+h)^(3-a)))

C <- k*(f*(a-3)*h*(v^(2-a))-(v^(3-a))+((v+f*h)^(3-a)))

D <- A*(B-C)

return(D)

}

# modelled covariance

modcov <- function(a,l,v,k,f,mx,h=1,lag=1) {

A = (l*(1+k/f)*(mx^2)*(v^a))/((f^2)*((f^2)-1)*(a-1)*(a-2)*(a-3))

B = (2*(f^2)-2+k*f)*(f^2)*(((v+(lag+1)*h)^(3-a))-2*((v+lag*h)^(3-a))+((v+(lag-1)*h)^(3-a)))

C = k*(((v+(lag+1)*h*f)^(3-a))-(2*((v+h*lag*f)^(3-a)))+((v+(lag-1)*h*f)^(3-a)))

D <- A*(B-C)

return(D)

}

# modelled probability dry

modpdr <- function(a,l,v,k,f,h=1) {

mt = ((1+(f*(k+f))-(0.25*f*(k+f)*(k+4*f))+((f/72)*(k+f)*(4*(k^2)+27*k*f+72*(f^2))))*v)/(f*(a-1))

G00 = ((1-k-f+1.5*k*f+(f^2)+0.5*(k^2))*v)/(f*(a-1))

A = (f+(k*(v/(v+(k+f)*h))^(a-1)))/(f+k)

D <- l*(-h-mt+G00*A)
```

```
return(D)

}

# Historical statistics (National Technical University of Athens rain gauge, Athens)

mean1 = 0.1226;var1 = 0.6323;cov1lag1 = 0.3271;pdr1 = 0.9183
mean6 = 0.7358;var6 = 10.1490;cov6lag1 = 4.0773;pdr6 = 0.8251
mean12 = 1.4705;var12 = 29.357;cov12lag1 =7.6865;pdr12 = 0.7476
mean24 = 2.9410;var24 = 76.667;cov24lag1 =10.2886;pdr24 = 0.6238

# Objective function

objfun <- function(x) {

a <- x[1];l <- x[2];v <- x[3];k <- x[4];f <- x[5];mx <- x[6]

w1=1;w2=1;w3=1;w4=1

S1 = w1*abs((modmean(a,l,v,k,f,mx,h=1)-mean1)/mean1) + w2*abs((modvar(a,l,v,k,f,mx,h=1)-var1)/var1) +
w3*abs((modcov(a,l,v,k,f,mx,h=1)-cov1lag1)/cov1lag1) + w4*abs((modpdr(a,l,v,k,f,h=1)-log(pdr1))/log(pdr1))

S24 = w3*abs((modcov(a,l,v,k,f,mx,h=24)-cov24lag1)/cov24lag1) +
w4*abs((modpdr(a,l,v,k,f,h=24)-log(pdr24))/log(pdr24))

S=S1+S24

return(S)
                    }

# set the interior and exterior parameters bounds

xmin <- c(1.00001,0.00001,0.00001,0.00001,0.00001,0.00001)
xmax <- c(10,0.1,10,10,0.99999,100)
xlow <- c(1.00001,0.00001,0.00001,0.00001,0.00001,0.00001)
xup <- c(20,0.1,20,20,0.99999,100)

# apply the evalutionary annealing-simplex method

par <- eas(n=6,m=20,xmin,xmax,xlow,xup,fn=objfun,maxeval=1500,ftol=1.e-07,ratio=0.99,pmut=0.9,
           beta=2,maxclimbs=5)
```

Index