Package ‘mcka’

February 15, 2013

Type Package

Title Machine coded genetic algorithms for real-valued optimization problems

Version 2.0.6

Date 2012-10-31

Author Mehmet Hakan Satman

Maintainer Mehmet Hakan Satman <mhsatman@istanbul.edu.tr>

Description Machine coded genetic algorithm (MCGA) is a fast tool for real-valued optimization problems. It uses the byte representation of variables rather than real-values. It performs the classical crossover operations (uniform) on these byte representations. Mutation operator is also similar to classical mutation operator, which is to say, it changes a randomly selected byte value of a chromosome by +1 or -1 with probability 1/2. In MCGAs there is no need for encoding-decoding process and the classical operators are directly applicable on real-values. It is fast and can handle a wide range of a search space with high precision. Using a 256-unary alphabet is the main disadvantage of this algorithm but a moderate size population is convenient for many problems. Package also includes multi_mcka function for multi objective optimization problems. This function sorts the chromosomes using their ranks calculated from the non-dominated sorting algorithm.

License GPL

LazyLoad yes

Repository CRAN

Date/Publication 2012-10-31 09:31:45

NeedsCompilation yes
**R topics documented:**

- mcga-package .................................................. 2
- mcga ................................................................. 3
- multi_mcga ......................................................... 4

**Index**

|--------------|--------------------------------------------------------------------------------|

**Description**

Machine coded genetic algorithm (MCGA) is a fast tool for real-valued optimization problems. It uses the byte representation of variables rather than real-values. It performs the classical crossover operations (uniform) on these byte representations. Mutation operator is also similar to classical mutation operator, which is to say, it changes a randomly selected byte value of a chromosome by +1 or -1 with probability 1/2. In MCGAs there is no need for encoding-decoding process and the classical operators are directly applicable on real-values. It is fast and can handle a wide range of a search space with high precision. Using a 256-unary alphabet is the main disadvantage of this algorithm but a moderate size population is convenient for many problems.

**Details**

- Package: mcga
- Type: Package
- Version: 2.0.3
- Date: 2012-01-06
- License: GPL
- LazyLoad: yes

**Author(s)**

Mehmet Hakan Satman
Maintainer: Mehmet Hakan Satman <mhsatman@istanbul.edu.tr>

**Examples**

```r
# A sample optimization problem
# Min f(xi) = (x1-7)^2 + (x2-77)^2 + (x3-777)^2 + (x4-7777)^2 + (x5-77777)^2
# The range of xi is unknown. The solution is
# x1 = 7
# x2 = 77
# x3 = 777
```
# x4 = 7777
# x5 = 77777
# Min f(xi) = 0
require("mcga")

f<-function(x){
  return ((x[1]-7)^2 + (x[2]-77)^2 +(x[3]-777)^2 +(x[4]-7777)^2 +(x[5]-77777)^2)
}

m<-mcga(popsize=200, chsize=5, minval=0.0, maxval=999999999.9, maxiter=2500, crossprob=1.0, mutateprob=0.01, elitism=1)
cat("Best chromosome:
")
print(m$population[1,])
cat("Cost: ",m$costs[1],"
")

mcga  

Performs machine coded genetic algorithms on a function subject to be minimized.

Description

Machine coded genetic algorithm (MCGA) is a fast tool for real-valued optimization problems. It uses the byte representation of variables rather than real-values. It performs the classical crossover operations (uniform) on these byte representations. Mutation operator is also similar to classical mutation operator, which is to say, it changes a randomly selected byte value of a chromosome by +1 or -1 with probability 1/2. In MCGAs there is no need for encoding-decoding process and the classical operators are directly applicable on real-values. It is fast and can handle a wide range of a search space with high precision. Using a 256-unary alphabet is the main disadvantage of this algorithm but a moderate size population is convenient for many problems.

Usage

mcga(popsize, chsize, crossprob = 1.0, mutateprob = 0.01, elitism = 1, minval, maxval, maxiter = 10)

Arguments

  popsize Number of chromosomes.
  chsize Number of parameters.
  crossprob Crossover probability. By default it is 1.0
  mutateprob Mutation probability. By default it is 0.01
  elitism Number of best chromosomes to be copied directly into next generation. By default it is 1
  minval The lower bound of the randomized initial population. This is not a constraint for parameters.
  maxval The upper bound of the randomized initial population. This is not a constraint for parameters.
  maxiter The maximum number of generations. By default it is 10
  evalFunc An R function. By default, each problem is a minimization.
multi_mcga

Performs multi objective machine coded genetic algorithms.

Description

Machine coded genetic algorithm (MCGA) is a fast tool for real-valued optimization problems. It uses the byte representation of variables rather than real-values. It performs the classical crossover operations (uniform) on these byte representations. Mutation operator is also similar to classical mutation operator, which is to say, it changes a randomly selected byte value of a chromosome by +1 or -1 with probability 1/2. In MCGAs there is no need for encoding-decoding process and the classical operators are directly applicable on real-values. It is fast and can handle a wide range of a search space with high precision. Using a 256-unary alphabet is the main disadvantage of this algorithm but a moderate size population is convenient for many problems.

This function performs multi objective optimization using the same logic underlying the mcga. Chromosomes are sorted by their objective values using a non-dominated sorting algorithm.
Usage
multi_mcga(popsize, chsize, crossprob = 1.0, mutateprob = 0.01, elitism = 1, minval, maxval, maxiter, numfunc, evalFunc)

Arguments

popsize    Number of chromosomes.
chsize     Number of parameters.
crossprob  Crossover probability. By default it is 1.0
mutateprob Mutation probability. By default it is 0.01
elitism    Number of best chromosomes to be copied directly into next generation. By
default it is 1
minval     The lower bound of the randomized initial population. This is not a constraint
for parameters.
maxval     The upper bound of the randomized initial population. This is not a constraint
for parameters.
maxiter    The maximum number of generations. By default it is 10.
numfunc    Number of objective functions.
evalFunc   An R function. By default, each problem is a minimization. This function must
return a cost vector with dimension of numfunc. Each element of this vector
points to the corresponding function to optimize.

Value
population Sorted population resulted after generations
costs       Cost values for each chromosomes in the resulted population
ranks       Calculated ranks using a non-dominated sorting for each chromosome

Author(s)
Mehmet Hakan Satman - mhsatman@istanbul.edu.tr

Examples
# We have two objective functions.
f1<-function(x){
  return(sin(x))
}

f2<-function(x){
  return(sin(2*x))
}

# This function returns a vector of cost functions for a given x sent from mcga
f<-function(x){
  return(c(f1(x), f2(x)))
}
# main loop
m<-multi_mcgapopsize=200, chsize=1, minval= 0, elitism=2,
    maxval= 2.0 * pi, maxiter=1000, crossprob=1.0,
    mutateprob=0.01, evalFunc=f, numfunc=2)

# Points show best five solutions.
curve(f1, 0, 2*pi)
curve(f2, 0, 2*pi, add=TRUE)

p <- m$population[1:5,]
points(p, f1(p))
points(p, f2(p))
Index

mcga, 3
mcga-package, 2
multi_mcga, 4