Package 'mopsocd'

February 20, 2015

Type Package

Title MOPSOCD: Multi-objective Particle Swarm Optimization with Crowding Distance

Version 0.5.1

Date 2013-06-04

Author Pros Naval

Maintainer Pros Naval <pcnaval@dcs.upd.edu.ph>

Description A multi-objective optimization solver based on particle swarm optimization with crowding distance.

Suggests scatterplot3d

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2013-06-04 12:24:06

R topics documented:

`examples` *Unconstrained and Constrained Optimization Examples*

Description

Examples on how to use the MOPSOCD solver for unconstrained and constrained optimization.

See Also

mopsocd pareto

Examples

```
## Typical MOPSOCD usage:
## 1) Define Objectives
##    Format:
##     fn <- function(x) {
##        ## User-Defined Objective Functions
##        f1 <- Objective Function 1
##        f2 <- Objective Function 2
##              ...
##        fn <- Objective Function n
##        ## End of User-Defined Objective Functions
##        return(c(f1,f2, ...,fn))
##     }
##
## 2) Define Constraints
##    Format:
##     gn <- function(x) {
##        ## User-Defined Constraints
##        g1 <- Constraint 1
##        g2 <- Constraint 2
##              ...
##        gn <- Constraint n
##        ## End of User-Defined Constraints
##        return(c(g1,g2, ...,gn))
##     }
##
## 3) Set Arguments
## 4) Run Solver
## 5) Print Pareto Object Fields
## 6) Plot Non-Dominated Points

######################
## Example 1: Viennet Test Function (Unconstrained Optimization)
######################
## Define Objectives
viennet <- function(x){
  f1 <- 0.5*(x[1]^2+x[2]^2)+sin(x[1]^2+x[2]^2)
  f2 <- 0.125*(3*x[1]-2*x[2]+4)^2+(1.0/27.0)*(x[1]-x[2]+1)^2+15
  f3 <- 1.0/(x[1]^2+x[2]^2+1)-1.1*exp(-(x[1]^2+x[2]^2))
  return(c(f1,f2,f3))
}

## Set Arguments
varcount <- 2
fncount <- 3
lbound <- c(-3,-3)
ubound <- c(3,3)
optmin <- 0
```

```
## Run Solver (gn omitted)
ex1 <- mopsocd(viennet,varcnt=varcount,fncnt=fncount,
             lowerbound=lbound,upperbound=ubound,opt=optmin)

## Access Pareto Object Fields
print(ex1$numsols)
print(ex1$objfnvalues)
print(ex1$paramvalues)

## Plot
#library(scatterplot3d)
#scatterplot3d(ex1$objfnvalues[,1],ex1$objfnvalues[,2],ex1$objfnvalues[,3])

######################
## Example 2: Kita Test Function (Constrained Optimization)
######################
## Define Objectives
kita <- function(x) {
    f1 <-  -(x[1] * x[1]) + x[2]
    f2 <- (x[1]/2)+x[2]+1
    return(c(f1,f2))
}

## Define Constraints
gn <- function(x) {
  g1 <- x[1]/6.0 + x[2] - 13.0/2.0 <= 0.0
  g2 <- x[1]/2.0 + x[2] - 15.0/2.0 <= 0.0
  g3 <- 5.0*x[1]+x[2] - 30.0 <= 0.0
  return(c(g1,g2,g3))
}

## Set Arguments
varcount <- 2
fncount <- 2
lbound <- c(0,0)
ubound <- c(7,7)
optmax <- 1

## Run Solver
ex2 <- mopsocd(kita,gn,varcnt=varcount,fncnt=fncount,
             lowerbound=lbound,upperbound=ubound,opt=optmax)

## Access Pareto Object Fields
print(ex2$numsols)
print(ex2$objfnvalues)
print(ex2$paramvalues)

## Plot
plot(ex2$objfnvalues[,1],ex2$objfnvalues[,2])

######################
## Example 3: Osyczka-Kundu Test Function (Constrained Optimization)
```

```
#####################
## Define Objectives
osyczka <- function(x){
f1 <- -(25.0*(x[1]-2.0)^2+(x[2]-2.0)^2+(x[3]-1.0)^2+(x[4]-4.0)^2+(x[5]-1.0)^2)
f2 <- 0.0
for (i in 1:6) {
f2 <- f2+x[i]^2
}
return(c(f1,f2))
}

## Define Constraints
gn <- function(x) {
  ## User-Defined Constraints
  g1 <- x[1] + x[2] - 2.0 >= 0.0
  g2 <- 6.0 - x[1] - x[2] >= 0.0
  g3 <- 2.0 - x[2] + x[1] >= 0.0
  g4 <- 2.0 - x[1] + 3.0*x[2] >= 0.0
  g5 <- 4.0 - (x[3] - 3.0)^2 - x[4] >= 0.0
  g6 <- (x[5] - 3.0)^2 + x[6] - 4.0 >= 0.0
  ## End of User-Defined Constraints
  return(c(g1,g2,g3,g4,g5,g6))
}

## Set Arguments
varcount <- 6
fncount <- 2
lbound <- c(0,0,1,0,1,0)
ubound <- c(10,10,5,6,5,10)
optmin <- 0

## Run Solver
ex3 <- mopsocd(osyczka,gn,varcnt=varcount,fncnt=fncount,
             lowerbound=lbound,upperbound=ubound,opt=optmin,
 popsize=100,maxgen=25,archivesize=500)
# For better results, use the following settings:
#  popsize=1000,maxgen=1000.

## Access Pareto Object Fields
print(ex3$numsols)
print(ex3$objfnvalues)
print(ex3$paramvalues)

## Plot
plot(ex3$objfnvalues[,1],ex3$objfnvalues[,2])
```

`mopsocd` *MOPSOCD: Multi-objective Particle Swarm Optimization with Crowding Distance*

Description

Multi-objective optimization involves maximizing or minimizing multiple interacting and/or conflicting objective functions subject to a set of contraints. MOPSOCD is a multi-objective optimization solver based on particle swarm optimization that uses crowding distance computation to ensure an even spread of non-dominated solutions.

Usage

```
mopsocd(fn,
        gn,
        varcnt,
        fncnt,
        lowerbound,
        upperbound,
        opt,
        popsize,
        maxgen,
        archivesize,
        verbosity,
        pMut,
        w,
        c1,
        c2)
```

Arguments

fn	Objective functions to be optimized
gn	Constraints (optional)
varcnt	Number of Parameters
fncnt	Number of Objectives
lowerbound	Parameter Lower Bound
upperbound	Parameter Upper Bound
opt	Optimization type (0: minimization; 1: maximization)
popsize	Population Size (default: 100)
maxgen	Number of Generations (default: 100)
archivesize	Maximum size of archive containing non-dominated points (default: 250)
verbosity	Verbosity Levels : 0,1,2,3 (default: 1)
pMut	Mutation Probability (default: 0.5)
w	Inertia Weight (default: 0.4)
c1	Acceleration Coefficient 1 (default: 1.0)
c2	Acceleration Coefficient 2 (default: 1.0)

Value

The returned value is a pareto object with the following fields:

numsols	Number of Solutions Found
paramvalues	Estimated Parameter Values
objfnvalues	Values of the Objectives

Author(s)

Pros Naval

References

C. R. Raquel and P.C. Naval, "An Effective use of Crowding Distance in Multiobjective Particle Swarm Optimization", Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2005), Washington, D.C., June 25-29, 2005.

See Also

examples pareto

---

pareto *Pareto Object*

Description

MOPSOCD returns a pareto object that contains the optimization results as well as the settings used to obtain them.

Value

A pareto object has the following fields:

numsols	Number of solutions found
paramvalues	Estimated Parameter Values
objfnvalues	Values of the Objectives
fn	Objective functions optimized
gn	Constraints used
varcnt	Number of Parameters
fncnt	Number of Objectives
lowerbound	Parameter Lower Bound
upperbound	Parameter Upper Bound
opt	Optimization type (0: minimization; 1: maximization)
popsize	Population Size

maxgen	Number of Generations
archivesize	Maximum size of archive containing non-dominated points
verbosity	Verbosity Level
pMut	Mutation Probability
w	Inertia Weight
c1	Acceleration Coefficient 1
c2	Acceleration Coefficient 2

See Also

mopsocd examples

Index