Package ‘ConConPiWiFun’

November 18, 2015

Type Package
Title Optimisation with Continuous Convex Piecewise (Linear and Quadratic) Functions
Version 0.4.6
Date 2013-06-05
Author Robin Girard
Maintainer Robin Girard <robin.girard@mines-paristech.fr>
Description Continuous convex piecewise linear (ccpl) resp. quadratic (ccpq) functions can be implemented with sorted breakpoints and slopes. This includes functions that are ccpl (resp. ccpq) on a convex set (i.e. an interval or a point) and infinite out of the domain. These functions can be very useful for a large class of optimisation problems. Efficient manipulation (such as log(N) insertion) of such data structure is obtained with map standard template library of C++ (that hides balanced trees). This package is a wrapper on such a class based on Rcpp modules.
License GPL (>= 2)
Depends methods, graphics, Rcpp (>= 0.10.3)
LinkingTo Rcpp
RcppModules mod_cplfunction,mod_cpqfunction
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-11-18 09:31:01

R topics documented:

ConConPiWiFun-package .................................................. 2
cplfunction .............................................................. 3
cplfunctionvec ......................................................... 4
cpqfunction ............................................................ 5
cpqfunctionvec ......................................................... 7
OptimPriceStorage ...................................................... 8

Index 10
ConConPiWiFun-package

This package contains an implementation of continuous convex piecewise (linear) functions (quadratic coming soon)

Description

Continuous convex piecewise linear (ccpl) resp. quadratic (ccpq) functions can be implemented with sorted breakpoints and slopes. This includes functions that are ccpl (resp. ccpq) on a convex set (i.e. an interval or a point) and infinite out of the domain. These functions can be very useful for a large class of optimization problems. Efficient manipulation (such as log(N) insertion) of such data structure is obtained with map standard template library of C++ (that hides balanced trees). This package is a wrapper on such a class based on Rcpp modules.

Details

Package: ConConPiWiFun
Type: Package
Version: 0.3.0
Date: 2013-02-08
License: GPL

Author(s)

Robin Girard
Maintainer: <robin.girard@mines-paristech.fr>

References

Related Papers are

Examples

library(ConConPiWiFun)
### See
#? cplfunction for continuous convex piecewise functions
#? cplfunctionvec for (optimized) list of continuous convex piecewise functions
This class implements continuous convex piecewise linear functions

Description

This includes functions that are ccpl (resp. ccpq) on a convex set (i.e. an interval or a point) and infinite out of the domain. These functions can be very useful for a large class of optimisation problems. Efficient manipulation (such as log(N) insertion) of such data structure is obtained with map standard template library of C++ (that hides balanced trees). This package is a wrapper on such a class based on Rcpp modules.

Author(s)

Robin Girard

See Also

to See Also as cplfunction.

Examples

```r
## Construction of a piecewise linear function
##
Slopes=c(-1,2,Inf) # increasing ! convexity is required
Breakpoints=c(-Inf,2,4) # increasing. length is number of slopes +1
FirstNonInfBreakpointVal=3
CCPWLFunc1=new(cplfunction,Slopes,Breakpoints,FirstNonInfBreakpointVal)
plot(CCPWLFunc1) #visualisation method

### Etoile transformation (legendre transform of f)
# Changes f no return value
CCPWLFunc1$Etoile()
plot(CCPWLFunc1) #if f = CCPWLFunc1 CCPWLFunc1 becomes is f^*(y) =inf_x {xy-f(x)}
CCPWLFunc1$Etoile()
plot(CCPWLFunc1) ## (f^*)^* is f !

### Squeeze function
# Changes f, no return value
left=-Inf; right=3
CCPWLFunc1$Squeeze(left,right) # CCPWLFunc1 is now infinite (or not definite) out of [left,right]
# i.e. all breakpoints out of [left,right] removed

### Swap function
```
# Changes f no return value!
y=2;
CCPWLFunc$Swap(y)
plot(CCPWLFunc); #now f = CCPWLFunc1 is replaced by x -> f(y-x)

### Sum function (uses fast insertion) do not affect operands
CCPWLFunc1=new(cplfuncvec,c(-1,2,Inf),c(-Inf,2,4),0)
CCPWLFunc2=new(cplfuncvec,c(-1,2,Inf),c(-Inf,1,3),0)
CCPWLFunc1plus2=Sum1(CCPWLFunc1,CCPWLFunc2)
CCPWLFunc1plus2

par(mfrow=c(1,3))
plot(CCPWLFunc2,col='red');
plot(CCPWLFunc1,col='blue');
plot(CCPWLFunc1plus2);

rm(list=ls())
gc()

cplfunctionvec

This class implements "optimized list" of continuous convex piecewise linear functions

Description

This is a wrapper to stl vector of convex piecewise linear functions. Allows to loop efficiently on such list.

Author(s)

Robin Girard

See Also

See Also as cplfunction, cpqfunctionvec

Examples

####
# construction of a vector of
# continuous convex piecewise linear functions

CCPWLFuncList=new(cplfunctionvec)
This class implements continuous convex piecewise quadratic functions

Description

This includes functions that are ccpq on a convex set (i.e. an interval or a point) and infinite out of the domain. These functions can be very useful for a large class of optimisation problems. Efficient manipulation (such as log(N) insertion) of such data structure is obtained with map standard template library of C++ (that hides balanced trees). This package is a wrapper on such a class based on Rcpp modules.

Author(s)

Robin Girard
See Also

to See Also as \texttt{cplfunction}.

Examples

\begin{verbatim}
##
#Construction of a piecewise quadratic function
##
Slopes1=c(-1,2)
Slopes0=c(-2,0)# increasing ! convexity is required
Breakpoints=c(-Inf,2,4) # increasing. length is number of slopes +1
FirstNonInfBreakpointVal=3
CCPWLFunc1=new(cpqfunction,Slopes0,Slopes1,Breakpoints,FirstNonInfBreakpointVal)
CCPWLFunc1$get_BreakPoints_() ## return Breaks AND Slopes
plot(CCPWLFunc1)

###Etoile transformation (legendre transform of f)
# Changes f no return value
CCPWLFunc1$Etoile()
CCPWLFunc1$get_BreakPoints_()
CCPWLFunc1$Etoile()
CCPWLFunc1$get_BreakPoints_() ## \(f^*\) is \(f\) !

###Squeeze function
# Changes f, no return value
left=-1; right=4
CCPWLFunc1$Squeeze(left,right) # CCPWLfunc1 is now infinite (or not definite) out of \([left, right]\])
# i.e. all breakpoints out of \([left, right]\] removed
CCPWLFunc1$get_BreakPoints_()

###Swap function
# Changes f no return value !
y=2;
CCPWLFunc1$Swap(y)
CCPWLFunc1$get_BreakPoints_() #now f = CCPWLFunc1 is replaced by \(x \to f(y-x)\)

### Sum function (uses fast insertion) do not affect operands
CCPWLFunc1=\texttt{new(cpqfunction,Slopes0,Slopes1,Breakpoints,FirstNonInfBreakpointVal)}
CCPWLFunc2=\texttt{new(cpqfunction,Slopes0,Slopes1+1,Breakpoints,FirstNonInfBreakpointVal)}
CCPWLFunc1plus2=\texttt{Sum(CCPWLfunc1,CCPWLfunc2)}
CCPWLFunc1plus2$get_BreakPoints_()

rm(list=ls())
gc()
\end{verbatim}
This class implements "optimized list" of continuous convex piecewise quadratic functions

Description
This is a wrapper to std vector of convex piecewise quadratic functions. Allows to loop efficiently on such list.

Author(s)
Robin Girard

See Also
to See Also as cpqfunction, cplfunctionvec

Examples
CCPWlfunclist=new(cpqfunctionvec)
CCPWlfunclist$push_back(new(cpqfunction,c(0),c(1),c(-2, 2),0))
CCPWlfunclist$push_back(new(cpqfunction,c(0),c(1),c(-2, 2),0))
CCPWlfunclist=new(cpqfunctionvec)
n=1000; Y=rnorm(n); S0=array(0,n)+Y; S1=array(1,n)+Y; B0=array(-Inf,n); B1=array(Inf,n);
for (i in 1:n){
  CCPWLfunclist$push_back(new(cpqfunction,S0[i],S1[i],c(B0[i],B1[i]),0))
}
CCPWLFunclist$size() ## gives the size
## The same but faster
CCPWLFunclist=new(cpqfunctionvec)
CCPWLFunclist$SerialPush_0Breaks_Functions(S0,S1);

#### method OptimMargInt solves
# min_x sum_i=1^n C_i(x_i)
# Pmoin_i<= x_i <=Pplus_i i=1,...,n
# Cmoin_i<= x_j <=Cplus_i i=1,...,n
Pmoin=array(-1,n); Pplus=array(1,n); Cmoin=array(0,n); Cplus=array(5,n);
res=CCPWLFunclist$OptimMargInt(Pmoin,Pplus,Cmoin,Cplus)
par(mfrow=c(1,2))
plot(Y,type='l')
lines(y=Pmoin,x=1:n,col='blue'); lines(y=Pplus,x=1:n,col='blue');
lines(y=res$xEtoile,x=1:n,col='red')
text(x=800,y=3,paste("Optimum=",signif(sum(abs(res$xEtoile-Y)),digits=6)))
plot(Y,type='l',ylim=c(min(Y),max(diffinv(res$xEtoile)[1:n+1])))
lines(y=Cmoin,x=1:n,col='blue'); lines(y=Cplus,x=1:n,col='blue');
lines(y=diffinv(res$xEtoile)[1:n+1],x=1:n,col='red')
rm(list=ls())
gc()

---

**OptimPriceStorage**

*Optimisation of storage operation with market prices taking into account storage efficiency and network taxes.*

---

**Description**

Optimisation of storage operation with market prices taking into account storage efficiency and network taxes.

**Usage**

```r
OptimPriceStorage(Prices,Pplus,Pmoins,Cplus,Cmoins=0,
                   efficiencyS=0,efficiencyP=efficiencyS,networnTax=0)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prices</td>
<td>A vector of prices</td>
</tr>
<tr>
<td>Pplus</td>
<td>A value for the upper power constraint or a vector of values with the same size as Prices</td>
</tr>
<tr>
<td>Pmoins</td>
<td>A value for the lower power constraint or a vector of values with the same size as Prices</td>
</tr>
<tr>
<td>Cplus</td>
<td>A value for the upper capacity constraint or a vector of values with the same size as Prices</td>
</tr>
<tr>
<td>Cmoins</td>
<td>A value for the lower capacity constraint or a vector of values with the same size as Prices</td>
</tr>
<tr>
<td>efficiencyS</td>
<td>storage efficiency when storing electricity</td>
</tr>
<tr>
<td>efficiencyP</td>
<td>storage efficiency when producing electricity</td>
</tr>
<tr>
<td>networkTax</td>
<td>networkTax</td>
</tr>
</tbody>
</table>

**Details**

function OptimPriceStorage solves

\[
\min_x \sum_{i=1}^n Y_i \cdot \text{efficiencyP} \cdot x_i \cdot (x_i < 0) + (Y_i \cdot \text{efficiencyS} + \text{networkTax}) \cdot x_i \cdot (x_i > 0) \\
\text{Pmoins}_i \leq x_i \leq \text{Pplus}_i \quad i=1,\ldots,n \\
\text{Cmoins}_i \leq \sum_{j=1}^i x_j \leq \text{Cplus}_i \quad i=1,\ldots,n
\]

when efficiency=1 and networkTax=0 this gives

\[
\min_x \sum_{i=1}^n Y_i \cdot x_i \\
\text{Pmoins}_i \leq x_i \leq \text{Pplus}_i \quad i=1,\ldots,n \\
\text{Cmoins}_i \leq \sum_{j=1}^i x_j \leq \text{Cplus}_i \quad i=1,\ldots,n
\]
Value
A list with
- Operation the optimal operation for each time step
- Revenue the revenue for each time step

Note
TODO

Author(s)
Robin Girard

References
TODO

See Also
See Also `cplfunction` (method OptimMargInt that is more general)

Examples

```r
n=8760
Prices=runif(n,1,100) #uniform random prices in [1;100] in Euro/MWh
Pmax=1; Pmin=-1; Cmax=5; ## 1MW maximum during 5 hours.
res=optimpricestorage(Prices,Pmax,Pmin,Cmax) # solving the optimization problem
sum(res$Revenue)## Revenue
res=optimpricestorage(Prices,Pmax,Pmin,Cmax,efficiencyS=0.8) # solving the optimization problem
sum(res$Revenue)## Revenue
```
Index

*Topic Optimisation, Dynamic programming
  ConConPiWiFun-package, 2
*Topic \textasciitilde{kwd1}
cplfuction, 3
cplfuctionvec, 4
cpqfuction, 5
cpqfuctionvec, 7
OptimPriceStorage, 8
*Topic \textasciitilde{kwd2}
cplfuction, 3
cplfuctionvec, 4
cpqfuction, 5
cpqfuctionvec, 7
OptimPriceStorage, 8

ComputeMarketPrices
  (OptimPriceStorage), 8
ConConPiWiFun (ConConPiWiFun-package), 2
ConConPiWiFun-package, 2
cplfuction, 3, 3, 4, 6, 9
cplfuctionvec, 4, 7
cpqfuction, 5, 7
cpqfuctionvec, 4, 7

InfConv1 (cplfuction), 3
InfConvq (cpqfuction), 5

OptimPriceMarket_1 (OptimPriceStorage), 8
OptimPriceMarket_q (OptimPriceStorage), 8
OptimPriceStorage, 8
OptimPriceStorage_ (OptimPriceStorage), 8

plot, ANY-method (cplfuction), 3
plot, Rcpp_cplfuction-method (cplfuction), 3
plot-methods (cplfuction), 3