Various benchmarks, comparisons and checks for the NMOF package

Enrico Schumann
es@enricoschumann.net

2013-08-11

Contents

1 A binary knapsack problem

The first example is taken from the adagio package (Borchers, 2012).

```
> require("NMOF")
> require("adagio")
```

With the package attached, we can run the example (see `?adagio:::knapsack`).

```
> ## Example 1
> p <- c(15, 100, 90, 60, 40, 15, 10,  1)
> w <- c( 2,  20, 20, 30, 40, 30, 60, 10)
> cap <- 102
> (is <- knapsack(w, p, cap))
```

```
$capacity
[1] 102

$profit
[1] 280

$indices
[1] 1 2 3 4 6
```

The aim will be to obtain the same results with Threshold Accepting. We collect all objects in a list `Data`.

```
> Data <- list(p = p, w = w, n = length(p), cap = cap)
```

The objective function is straightforward; the neighbourhood function takes care of the constraints.

```
> OF <- function(x, Data)
     -sum(x * Data$p)
> neighbour <- function(x, Data) {
     xn <- x
     p <- sample.int(Data$n, size = 1L)
     xn[p] <- !xn[p]
     if (sum(Data$w*xn) > Data$cap)
         x
     else
         xn
 }
```

We run `TAopt`.

```
> algo <- list(x0 = logical(Data$n),  ## a random start
              printDetail = TRUE, printBar = FALSE,
              q = 0.99, neighbour = neighbour,
              nS = 100)
> system.time(sol <- TAopt(OF, algo = algo, Data))
```

```
Threshold Accepting.

Computing thresholds ... OK.
Estimated remaining running time: 0.019 secs.


Running Threshold Accepting...
Initial solution:  0
Finished.
Best solution overall: -280
   user  system elapsed
  0.088   0.000   0.087
```

```
> OF(sol$xbest,Data)
```

```
[1] -280
```

We compare this with the example's solution.

```
> ## Example 1
> ## [1] 1 2 3 4 6 , capacity 102 and total profit 280
> xHWB <- logical(Data$n)
> xHWB[c(1:4,6)] <- TRUE
> OF(xHWB, Data)
```

```
[1] -280
```

And the second example.

```
> ## Example 2
> ## [1] 1 4 , capacity 50 and total profit 107
> p <- c(70, 20, 39, 37, 7, 5, 10)
> w <- c(31, 10, 20, 19, 4, 3,  6)
> cap <- 50
> (is <- knapsack(w, p, cap))
```

```
$capacity
[1] 50

$profit
[1] 107

$indices
[1] 1 4
```

With luck, we should get the same solution.

```
> Data <- list(p = p, w = w, n = length(p), cap = cap)
> algo <- list(x0 = logical(Data$n),  ## a random start
               printDetail = TRUE, printBar = FALSE,
               q = 0.99, neighbour = neighbour, nS = 100)
> system.time(sol <- TAopt(OF, algo = algo, Data))
```

```
Threshold Accepting.

Computing thresholds ... OK.
Estimated remaining running time: 0.019 secs.


Running Threshold Accepting...
Initial solution:  0
Finished.
Best solution overall: -96
   user  system elapsed
  0.084   0.004   0.088
```

```
> OF(sol$xbest, Data)
```

```
[1] -96
```

2 A subset sum problem

This example is taken from

`https://stat.ethz.ch/pipermail/r-help/2010-January/226267.html`

and was provided by Hans Werner Borchers. We call the solution he provided `xHWB`.

```
> set.seed(8232)
> X <- runif(100L)
> ## Find subset that sums up close to 2.0 !
> i <- sort(c(84,54,11,53,88,12,26,45,25,62,96,23,78,77,66,1))
> sum(X[i])
```

```
[1] 2.000451
```

```
> ## --> should be 2.000451
>
> xHWB <- logical(100L)
> xHWB[i] <- TRUE
> sum(X[xHWB]) ## check
```

```
[1] 2.000451
```

We can try to solve this problem with the `optim` function. The function will not allow us to pass arguments explicity, so we use functions `makeN` and `makeF`. But note that `optim` needs a numeric solution vector.

```
> ## try with optim/SANN
> makeN <- function(X, size = 1L) {
      function(x) {
          x <- x > 0L
          p <- sample.int(100, size = size)
          x[p] <- !x[p]
          x
      }
 }
> makeF <- function(X) {
      function(x) {
          x <- x > 0L
          abs(sum(X[x]) - 2)
      }
 }
> F <- makeF(X)
> N <- makeN(X)
```

```
> x0 <- runif(100)>0.5
> F(x0) ## initial solution
```

```
[1] 23.33506
```

```
> result <- optim(par = x0, fn = F,  N, method = "SANN",
                  control = list(maxit = 20000,
                  temp = 1))
> F(as.logical(result$par)) ## final solution
```

```
[1] 0.0002253845
```

Tackling that example various other heuristics is discussed in
http://enricoschumann.net/files/NMOF_Rmetrics2012.pdf .

3 Minimum-variance and the tangency portfolio

We define a function `resample` (defined on the help page of `sample`) and pass it with `data`. We work with random data.

```
> require("quadprog")
> na <- 50L   ## number of assets
> ns <- 100L  ## number of scenarios
> R  <- array(rnorm(ns*na, mean = 0.005, sd = 0.015),
              dim = c(ns, na))
> mu <- colMeans(R)
> rf <- 0.0001
> mu2 <- mu - rf
> ## TEST 1: minimum-variance portfolio (long/short)
> wsup <- 0.05
> winf <- -0.05
> Q <- 2*cov(R)
> A <- array( 1, dim = c(1,na)); a <- 1
> B <- rbind(-diag(na),diag(na))
> b <- rbind(array(-wsup, dim = c(na,1)),
             array( winf, dim = c(na,1)))
> result <- solve.QP(Dmat = Q, dvec = rep(0, na),
                     Amat = t(rbind(A,B)), bvec = rbind(a, b),
                     meq = 1)
> wqp <- result$solution
> resample <- function(x, ...)
     x[sample.int(length(x), ...)]
> Data <- list(RR = cov(R), na = na, ns = ns,
               eps = 0.10/100, winf = winf, wsup = wsup,
               resample = resample)
> neighbour <- function(w, Data){
     toSell <- w > Data$winf
```

```
     toBuy  <- w < Data$wsup
     i <- resample(which(toSell), size = 1L)
     j <- resample(which(toBuy),  size = 1L)
     eps <- runif(1L) * Data$eps
     eps <- min(w[i] - Data$winf, Data$wsup - w[j], eps)
     w[i] <- w[i] - eps
     w[j] <- w[j] + eps
     w
 }
> OF <- function(w, Data) {
     aux <- crossprod(Data$RR,w)
     crossprod(w,aux)
 }
> w0 <- runif(na)
> w0 <- w0/sum(w0)
> algo <- list(x0 = w0, neighbour = neighbour,
              nS = 5000L, nT = 10L, nD = 2000L, q = 0.02,
              printBar = FALSE, printDetail = FALSE)
> res <- TAopt(OF,algo,Data)
> as.numeric(16 * 100 *sqrt(res$OFvalue)) -
     as.numeric(16 * 100 *sqrt(result$value))
[1] 1.655177e-07

> ## check constraints
> wSummary <- function(w)
     cat("min weight: ", min(w), "\n",
         "max weight: ", max(w), "\n",
         "sum of weights: ", sum(w), "\n",
         "no. of assets: ",  sum(w > 1e-12), "\n", sep ="")
> wSummary(res$xbest)
min weight: -0.04182952
max weight: 0.05
sum of weights: 1
no. of assets: 42

> wSummary(wqp)
min weight: -0.04184285
max weight: 0.05
sum of weights: 1
no. of assets: 42

> cat("Compare results: \n",
     "QP:", 100 * sqrt( crossprod(R %*% wqp)/Data$ns ),"\n",
     "TA:", 100 * sqrt( crossprod(R %*% res$xbest)/Data$ns ) ,"\n")
```

```
Compare results:
 QP: 0.5781987
 TA: 0.5782048
```

```
> ## TEST 2: tangency portfolio with non-negative weights
> winf <- 0; Q <- cov(R)
> A <- array(mu2, dim = c(1L, na)); a <- 1
> B <- diag(na); b <- array( winf, dim = c(na,1L))
> result <- solve.QP(Dmat = Q, dvec = rep(0,na),
                     Amat = t(rbind(A,B)), bvec = rbind(a,b),
                     meq = 1)
> w <- as.matrix(result$solution/sum(result$solution))
> SR <- t(w) %*% mu2 / sqrt(t(w) %*% Q %*% w)
> OF2 <- function(w, Data) {
     aux <- crossprod(Data$RR,w)
     sqrt(crossprod(w,aux)) / t(w) %*% Data$mu2
 }
> w0 <- runif(na); w0 <- w0/sum(w0)
> Data <- list(RR = cov(R), na = na, ns = ns, mu2 = mu2,
              eps = 0.10/100, winf = winf, wsup = 1)
> res <- TAopt(OF2,algo,Data)
> wSummary(res$xbest)
```

```
min weight: 0
max weight: 0.05915265
sum of weights: 1
no. of assets: 42
```

```
> wSummary(w)
```

```
min weight: -3.873481e-18
max weight: 0.05915861
sum of weights: 1
no. of assets: 42
```

```
> ## check difference between Sharpe ratios
> 1/res$OFvalue - as.numeric(SR)
```

```
              [,1]
[1,] -1.610566e-07
```

A Resources

You can download all the book's code examples from the book's home page,

`http://nmof.net`

The latest version of the NMOF package is available from

`http://enricoschumann.net/R/packages/NMOF/index.htm`

but note that this is the development version. More stable versions are available from CRAN.

New versions of the package and other news are announced through the `NMOF-news` mailing list; to browse the archives or to subscribe, go to

`https://lists.r-forge.r-project.org/cgi-bin/mailman/listinfo/nmof-news`

B Package version

```
> toLatex(sessionInfo())
```

- R version 3.0.1 (2013-05-16), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_GB.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=en_GB.UTF-8, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_GB.UTF-8, LC_PAPER=C, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: adagio 0.4.3, NMOF 0.28-1, quadprog 1.5-5
- Loaded via a namespace (and not attached): tools 3.0.1

References

Hans W Borchers. *adagio: Discrete and Global Optimization Routines*, 2012. URL `http://CRAN.R-project.org/package=adagio`.

Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical Methods and Optimization in Finance*. Academic Press, 2011.