Notes on the use of TAPF[1]

September 2009

Manfred Gilli
Department of Econometrics, University of Geneva
Manfred.Gilli@unige.ch

Gerda Cabej
Department of Econometrics, University of Geneva
Gerda.Cabej@unige.ch

Enrico Schumann
Department of Econometrics, University of Geneva
Enrico.Schumann@unige.ch

Distribution set

The folder contains the files `TAPF.p`, `goTAs.m`, `OFVaR.m` and `Data.mat` which have been created under Matlab R2006a.

The Matlab function file `TAPF.p` implements the threshold accepting optimization heuristic as used in Gilli *et al.* (2006). Further information about implementation details of the threshold accepting heuristic can be found in (Gilli and Winker, 2009; Gilli and Schumann, 2010; Winker, 2001). For more examples on the use of heuristics in portfolio optimization see (Maringer, 2005; Gilli and Këllezi, 2002*a*; Gilli and Këllezi, 2002*b*).

File `goTAs.m`

This script provides an example on how to call the function `TAPF.p`. The example concerns the optimization of a function defining the `VaR` for a given set `Ps` of 1560 price scenarios for 130 assets. The data are loaded from the file `Data.mat` which also contains a vector `pT` of asset prices for the current period T.

Function `TAPF`

The function `TAPF` has two sets of input arguments. The first set is fixed and is composed by the following three arguments:

`OF`	Character string with name of objective function
`TAopt`	Structure with parameters for the algorithm
`OD`	Structure containing problem specific parameters

Parameters in `TAopt` are set by default and can be overwritten by the user. In the example we provide values adapted to the problem at hand. Allowing for more function evaluations, i.e. higher values for `TAopt.Restarts`×`TAopt.Rounds`×`TAopt.Steps` enhances the quality of the solution. The working of the algorithm can be visualized with the option `TAopt.Figures = 'yes'` which may help detecting malfunctioning of the heuristic.

The second set, in our example `OD.vT,rd,rf,beta`, is an arbitrary list of optional parameters which can be passed to the objective function provided by the user.

The output argument of `TAPF` is a structure with the following elements:

`J`	Set of asset indices in portfolio
`x`	Integer vector of assets' quantities
`w`	Vector of assets' weights
`cash`	Residual cash
`F`	Value of objective function

Objective function

The objective function provided by the user has also two sets of input arguments. Again the first set is:

`x`	Integer vector of assets' quantities
`cash`	Residual cash
`J`	Set of asset indices in portfolio
`Psx`	Vector of portfolio values (`Ps * x`)

These arguments may or may not be used in the objective function but need always to be present in the definition of the function. The remaining set of arguments are optional parameters, in our case `OD.vT,rd,rf,beta`.

There are two output argument:

`F`	Value of objective function
`pen`	Row vector with three elements

The value of the vector `pen` at the solution is attached in `TAPF` to the structure forming the output argument and can be used to check to what extent the constraints have been satisfied.

[1]Project partially supported by the Marie Curie RTN COMISEF (www.comisef.eu). E. Këllezi and H. Hysi contributed at several stages to the project.

References

Gilli, M. and E. Këllezi (2002*a*). A Global Optimization Heuristic for Portfolio Choice with VaR and Expected Shortfall. In: *Computational Methods in Decision-making, Economics and Finance* (E. J. Kontoghiorghes, B. Rustem and S. Siokos, Ed.). Applied Optimization Series. Kluwer Academic Publishers. pp. 167–183.

Gilli, M. and E. Këllezi (2002*b*). The Threshold Accepting Heuristic for Index Tracking. In: *Financial Engineering, E-Commerce and Supply Chain* (P. Pardalos and V. K. Tsitsiringos, Ed.). Applied Optimization Series. Kluwer Academic Publishers, Boston. pp. 1–18.

Gilli, M., E. Këllezi and H. Hysi (2006). A data-driven optimization heuristic for downside risk minimization. *Journal of Risk* **8**(4), 1–18.

Gilli, Manfred and Enrico Schumann (2010). Portfolio optimization with "Threshold Accepting": a practical guide. In: *Optimizing Optimization: The Next Generation of Optimization Applications and Theory* (Stephen E. Satchell, Eds.). Elsevier.

Gilli, Manfred and Peter Winker (2009). Heuristic optimization methods in econometrics. In: *Handbook of Computational Econometrics* (David A. Belsley and Erricos Kontoghiorghes, Ed.). Wiley.

Maringer, Dietmar (2005). *Portfolio Management with Heuristic Optimization*. Vol. 8 of *Advances in Computational Management Science*. Springer.

Winker, P. (2001). *Optimization Heuristics in Econometrics*. Wiley. Chichester.

Listing 1: goTAs.m

```
clear all, close all hidden
load Data     % Load Ps matrix of price scenarios and pT

% -- Portfolio and objective function parameters --
OF       = 'OFVaR';    % Name of objective function
OD.Ps    = Ps;         % Matrix with price scenarios
OD.nA    = size(Ps,2);% Number of assets (columns)
OD.pT    = pT;         % Asset price for planning period T
OD.vT    = 1e7;        % Initial wealth
OD.vinf  = 0.001;      % Minimum holdings (can be a vector)
OD.vsup  = 0.070;      % Maximum holdings (can be a vector)
OD.csup  = 0.005;      % Maximum weight of cash
OD.K     = 30;         % Maximum cardinality of portfolio
rf   = 0.0001;    % Risk free return
rd   = 0.0010;    % Desired return
beta = 0.05;      % Right tail considered

% -- TA parameters --
TAopt = TAPF('defaults');
TAopt.Restarts =    5;
TAopt.Rounds   =   10;
TAopt.Steps    = 3000;
TAopt.Figures  = 'yes';

start = clock;
% -- Start portfolio optimization --
if TAopt.Restarts > 1
    TAopt.OnlyThresholdSequence = 'yes';
    output = TAPF(OF,TAopt,OD, OD.vT,rd,rf,beta);
    TAopt.ThresholdSequence = output.th;
    TAopt.OnlyThresholdSequence = 'no';
end

for i = 1:TAopt.Restarts
    j = unidrnd(1e6);  % Set random seed for generator
    TAopt.State = j;
    results(i) =  TAPF(OF,TAopt,OD, OD.vT,rd,rf,beta);
end
ttime = etime(clock,start);

if TAopt.Restarts > 1
    % -- Find best result among restarts
    Fbest = realmax; Fvec = NaN(1,TAopt.Restarts); nc=0;
    for i = 1:TAopt.Restarts
        if ~isempty(results(i))
            nc = nc + 1; Fvec(nc) = results(i).F;
            if results(i).F < Fbest
                ibest = i;
                Fbest = results(i).F;
            end
        end
    end
    if strcmp(TAopt.Figures,'yes')
        figure(2),subplot(313)
        h = cdfplot(Fvec);
        set(h,'Color','m','LineWidth',2);
        title('Empirical CDF of solutions');
        set(gca,'FontSize',12); grid on;
        ylabel(''); xlabel('');
    end
else
    ibest = 1;
end

% -- Results
J      = results(ibest).J;
% Indices of assets in portfolio
x      = results(ibest).x;    % Long vector (1:nA)
cash   = results(ibest).cash;
w      = results(ibest).w;    % Long vector (1:nA)
F      = results(ibest).F;
% Value of objective function

% Plot final portfolio weights and print results
if strcmp(TAopt.Figures,'yes')
    figure(1),subplot(211),hold off
    bar(w,'FaceColor',[0 0 1]), hold on, grid on
    title('Final portfolio weights')
    hb = bar(OD.nA+1,cash/OD.vT,'r');
    set(hb,'Edgecolor','r');
    ylim([0 1.1*max(w)]);
end
fprintf('\n Value of objective function: %7.0f',F);
fprintf('  (%i assets selected)',length(J));
pen = results(i).Pen; % pen = [f(pen1,pen2) pen1 pen2]
if pen(1) ~= 0
    fprintf(['\n Constraints not satisfied:',...
             'return is %6.4f\n'],-pen(2));
end
fprintf('\n Asset  Weight (%%)   Quantity');
for i = 1:length(J)
    j = J(i);
    fprintf('\n %3i    %5.2f
%6i',j,100*w(j),x(j));
end
fprintf('\n');

fprintf('\n %i restarts in %i sec\n',TAopt.Restarts,...
    fix(ttime));
fprintf('   (%i sec per restart)\n',...
    fix(ttime/TAopt.Restarts));
```

Listing 2: OFVaR.m

```
function [F,pen] = OFVaR(x,cash,J,Psx, vT,rd,rf,beta)
% x and J not used (need to stay in list of arguments)
c1 = 10;
nS = length(Psx);          %(Psx=Ps*x)
vs   = Psx + (1+rf)*cash;
loss = sort(vT - vs);
iVaR = ceil((1-beta)*nS);
VaR  = loss(iVaR);
El   = mean(loss);

% Return constraint
rr   = El/vT;
pen1 = max(rd+rr,0);

penF = c1*vT*pen1;
F    = VaR + penF;
pen  = [penF pen1 -rr];  % Must have 3 elements
```