HTCOOR Program Documentation[‡]

Jordi Castro[†] and J.Antonio González[†]
Statistics and Operations Research Dept.
UPC

Date 07/2000
DR 2000/12

[†] Statistics and Operations Research Dept.
Universitat Politècnica de Catalunya
Campus Sud, c. Pau Gargallo 5
08071 Barcelona (Spain)
e-mails: `jcastro@eio.upc.es` and `jose.a.gonzalez@upc.es`

[‡] This work has been partially supported by the European ESPRIT Project 22652, and by the

CICYT Project TAP99-1075-C02-02.

Abstract: HTCOOR is a computer package to solve the long-term hydrothermal coordination problem of electricity generation. HTCOOR was developed for the SLOEGAT European Esprit Project 22652. This document describes the main features of the package. Some computational results obtained in the solution of real world problems are also reported.

Keywords: Electricity generation, Long-term hydrothermal coordination, Nonlinear optimization, Optimization codes.

1 Formulation of the hydrothermal coordination problem

This section just outlines the exact formulation of the hydrothermal coordination problem solved by the HTCOOR package. In Section 2 we will refer to some of the constraints of this model. A thorough description of the model and the resulting optimization problem can be found in [1].

The final formulation of the hydrothermal coordination problem can be written as:

$$\min_{X,V,E,F,G,P} \quad \sum_{i=1}^{Ni} \left(\sum_{j=1}^{Nu} \sum_{f=1}^{Nf} y_{f,j}^{i} Z_{f,j}^{i} + y_X^i E_X^i \right) \tag{1.1}$$

$$\text{s.t.} \quad \sum_{a \in A_n^k} (X_a^i)^k + (V_n^{i-1})^k + (I_n^i)^k = \sum_{a \in \Omega_n} (X_a^i)^k + (V_n^i)^k \qquad k = 0 \dots K-1, i = 1 \dots Ni, n = 1 \dots Nr \tag{1.2}$$

$$\begin{aligned} &\sum_{k=0}^{K-1} (X_a^i)^k \le \overline{X_a} && \text{for all arc } a, i = 1 \dots Ni \\ &\sum_{k=0}^{K-1} (V_n^i)^k \le \overline{V_n} && i = 1 \dots Ni, n = 1 \dots Nr \end{aligned} \tag{1.3}$$

$$\begin{aligned} &(X_a^i)^k \le \overline{X_a^k} && \text{for all arc } a, k = 0 \dots K-1, i = 1 \dots Ni \\ &(V_n^i)^k \le \overline{V_n^k} && k = 0 \dots K-1, i = 1 \dots Ni, n = 1 \dots Nr. \end{aligned} \tag{1.4}$$

$$F_{f,j}^{i-1} + Z_{f,j}^{i} = F_{f,j}^{i} + \varepsilon_j E_{f,j}^{i} \qquad i = 1 \dots Ni, j = 1 \dots Nu, f = 1 \dots Nf \tag{1.5}$$

$$(G^i)^k = \Gamma^i \big((X^i)^k, (V^{i-1})^k, (V^i)^k \big) \qquad k = 0 \dots K-1, i = 1 \dots Ni \tag{1.6}$$

$$G_\Delta^i = \frac{1}{2} \left[\sum_{k=1}^{K-1} \pi_k \left((G^i)^{0k} + (G^i)^{0k-1} \right) \right] - (G^i)^0 \qquad i = 1 \dots Ni \tag{1.7}$$

$$G_\Delta^i = \sum_{j=1}^{Nu} G_{\Delta j}^i \qquad i = 1 \dots Ni \tag{1.8}$$

$$P_j^i = PE^i \left(E_j^i + \sum_{m=1}^{j-1} E_m^i + G_{\Delta m}^i \right) - PE^i \left(\sum_{m=1}^{j-1} E_m^i + G_{\Delta m}^i \right) \qquad i = 1 \dots Ni, j = 1 \dots Nu \tag{1.9}$$

$$P_j^i \leq \overline{P_j} \qquad i = 1 \ldots Ni, j = 1 \ldots Nu \quad (1.10)$$

$$PE^i \left(\sum_{j=1}^{Nu} E_j^i + G_\Delta^i \right) = \tilde{P}_\Delta^i + \sum_{j=1}^{Nu} P_j^i \qquad i = 1 \ldots Ni \quad (1.11)$$

The main variables involved in the optimization are the X and V for the hydro network, the E, F and G for the thermal network, and the P for the power of the thermal units—computed through the PEC [1].

The meaning of the different equations is as follows:

- The objective function (1.1) is the minimization of the overall cost and it is made of two parts: the purchases of fuels and the use of emergency energy. The symbols $y_{f,j}^i$ and y_X^i refer to unitary costs for fuels and emergency energy, respectively.
- Equations (1.2–1.4) are related with the hydro network. (1.2) are the balance constraints, while (1.3) and (1.4) refer to the mutual—for all the commodities—and individual—for each commodity— capacity constraints of the hydro network arcs.
- Constraints (1.5) are the balance equations for the thermal network.
- (1.6) uses the notation $\Gamma(\cdot)$ as a compact form for the equations that compute the hydro generation for each commodity as a function of the arcs of the hydro network. These generations are used in (1.7) to obtain the stochastic hydro generation, which, as imposed in (1.8), must equal the total energy of the pseudo-hydro units used to cover the SGDC.
- (1.9) introduces the new variables P_j^i (thermal power for each unit and interval). (1.10) imposes bounds on these variables.
- Finally, (1.11) imposes the coverage of the the SGDC (smoothed generation duration curve).

2 Description of the HTCOOR package

2.1 General overview

The HTCOOR package implements the model formulated in Section 1 (see [1] for more details). It is mainly written in Fortran (75% of the source, aside from the optimizer code). Only the main program and several routines are coded in C, to allow the dynamic allocation of the main vectors needed during the optimization stage. The length of the program is roughly 10,000 lines.

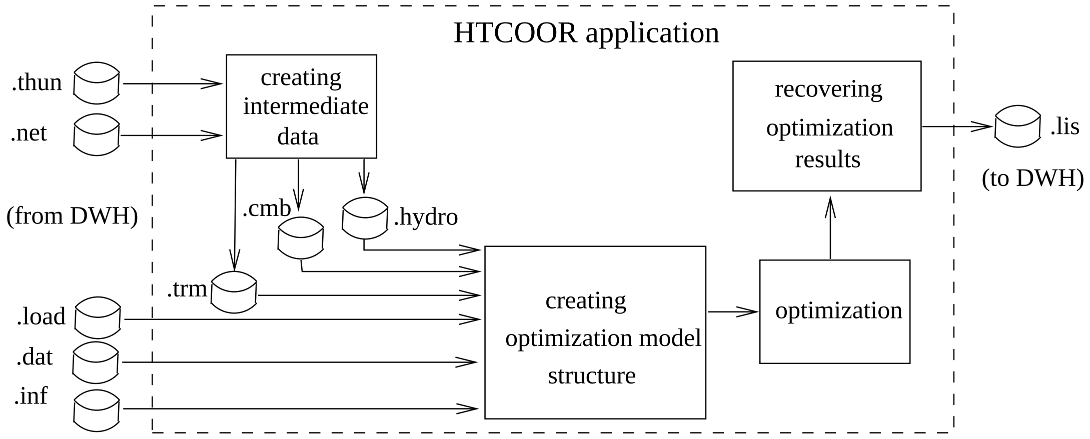
HTCOOR application
.thun
.net
(from DWH)
.load
.dat
.inf
creating intermediate data
.cmb
.hydro
.trm
creating optimization model structure
optimization
recovering optimization results
.lis
(to DWH)

Figure 2.1. Structure of the HTCOOR application.

The structure of the application is depicted in Fig. 2.1. The current version of the package has not been completely integrated with any data warehouse or data base, so the input/output data are still read/written from/to files. From the figure we see that the problem description is done through five input files:

- file ``‘‘model’’.thun``: describes the thermal system.
- file ``‘‘model’’.net``: describes the hydraulic system.
- file ``‘‘model’’.load``: contains the information about the load demand.
- file ``‘‘model’’.dat``: with general information (e.g., number of intervals, length of each interval, number of commodities, etc.)
- files ``‘‘reservoir’’.inf``: with historical data about inflows at the reservoirs of the hydraulic system.

Firstly, from the `.thun` and `.net` files, the application generates three more intermediate files:

- file `‘‘model’’.trm`: for the thermal system.
- file `‘‘model’’.hydro`: for the hydraulic system.
- file `‘‘model’’.cmb`: for the fuel information (prices, availability, etc.)

Thereafter, from the three intermediate files, and the original `.load`, `.dat` and `.inf` ones, the package creates the structure of the optimization model to be solved. This procedure is dependent of the type of solver to be applied. Up to now only the solvers Minos 5.5 [2] and Snopt 5.3 [3] have been used, which, basically, share the same model description structure. Finally, the problem is optimized and the solution obtained—if the solver finishes successfully—is written to the file `.lis`.

It is worth to note that the HTCOOR package does not make use of any modeling language (as AMPL or GAMS). Although the use of such kind of tools would extremely simplify the implementation task, we got some limitations. First, we needed to perform several optimizations to obtain some intermediate data for the final optimization stage. The adhoc implementation developed permitted and efficient communication among these different optimization problems. The second reason is related to the execution time: adhoc models implemented in C and Fortran are much more efficient than those obtained with modeling languages—though the latter can be coded in a fraction of the time required by the former. This second point is specially relevant since, due to the size and nonlinearity of the instances to be solved, execution times tend to be large. The third and last reason was which really forced us to develop an ad-hoc package: up to now modeling languages are not able to deal with constraints given by parametric curves, as those that define the power as a function of the energy through a Bézier curve representation of the PEC (Power-Energy Curve) [1]. The best solution would be to use an external function, which is not still a straightforward task with current modeling systems. Another alternative would be to replace the equation $p = x_p(x_e^{-1}(e))$ (power p as a function of the energy e) by the two equations $p = x_p(t), e = x_e(t)$. However, this would mean adding an extra nonlinear constraint and variable–the t parameter—, for each interval and thermal unit, which would approximately double the number of nonlinear constraints of the problem.

The following sections describe some of the main stages of the HTCOOR package.

2.2 Input data

In order to solve a problem, HTCOOR must be fed with a complete description of the system involved, that is:

- characteristics of the thermal park,
- characteristics of the hydraulic park,
- environment:
  - — planning horizon,
  - — load forecast,
  - — random distribution parameters.

2.2.1 Thermal data

The thermal system is assumed to consist of a set of units, where each one can be either a Thermal Unit or a Contract. Both types almost have the same attributes. These are:

- identifier,
- capacity [MW],
- availability coefficient [%],
- fuel used,
- cost of generation. The cost is evaluated in a different way depending on whether is a Thermal Unit or a Contract:
  - for Thermal Units, the cost is computed taking into account the heat rate consumption curve, the price of the primary energy and the additional operational costs, assuming maximum efficiency;
  - for Contracts, the unitary cost depends on the amount of energy produced according to a zoned-prices system, and it is approximated through a quadratic function without independent term.
- maintenance schedule (not used in the current version).

Besides of this information, an estimated price for emergency energy must also be provided.

Once this information is read, the program sorts the units on an economical basis, taking into account the cost of generation (CG) of one MWh. The cost for unit j is computed as

$$\mathrm{CG}_j = 3.6\frac{\mathrm{PEP}_j}{\mathrm{Efic}_j} + \mathrm{AOC}_j,$$

where PEP_j is the Primary Energy Price for the j-th unit, AOC_j stands for the Additional Operational Costs, and Efic_j measures the energy efficiency of the unit:

$$\mathrm{Efic}_j = \frac{3600 \times \mathrm{Cap}_j}{\mathrm{HRC}_{2,j}\,\mathrm{Cap}_j^2 + \mathrm{HRC}_{1,j}\,\mathrm{Cap}_j + \mathrm{HRC}_{0,j}},$$

Cap_j being the capacity and $\mathrm{HRC}_{l,j}$ the coefficients of the quadratic polynomial that defines the Heat Rate Consumption of the unit.

2.2.2 Hydro data

The basin is modeled as a graph, with reservoirs as nodes. The arcs of the graph represent the water flows from one reservoir or interval to another, and include hydro and pumping units. The information retrieved to describe the hydraulic system is:

- set of reservoirs; for each one:
  - identifier,
  - is sink?,
  - minimum and maximum storage capacity,
  - initial volume at first interval and desired volume at the end of the planning horizon,

  - elevation over the maximum capacity level,
  - reservoir characteristic (a fourth degree polynomial is fitted),
  - historical records of inflows; daily natural inflows data over the past,
- set of arcs; for each one:
  - source and target reservoirs,
  - type (discharge, pump, spill),
  - maximum and minimum flow,
  - elevation over the sea level (for discharges only),
  - characteristic:
    - for discharges, a quadratic polynomial on the net head and the release is fitted,
    - for pumps, a fourth degree polynomial on the net head is fitted,
  - maximum power of the turbine/pump.

The attribute *is sink?* is used to identify a reservoir which could be used as a sink (infinitely big) for the arcs whose target is not in the considered set of reservoirs or it is non-existent.

2.2.3 Environment

Besides of the description of the generating resources, supplementary information should be provided to run a case:

- Planning horizon. The begin and final dates must be known, and, in addition, the user has to specify a division for the period under study. Taking into account that the usual length is one year, the subperiods could be months, or weeks into natural months, or any division specifying time lengths for each subperiod.
- Load forecast. It is an hourly array with the forecast of the load to be covered along the whole planning horizon.
- Random distribution parameters. The user must provide with suitable values of K (number of blocks of the multiblock random distribution for hydro variables) and Π (quantiles of the multiblock distribution) [1]. Actually, the option is not critical: it is rather a tradeoff between accuracy and speed of the solution to be achieved. High values of K are not good, since the problem would grow in size without any appreciable advantage. Values from 4 to 8 are considered good choices. As for Π, a reasonable choice would be to provide equally distributed values for all the π_k.

2.3 Structure of the optimization model

The original model of equations (1.1–1.11) is implemented in a slightly different way in practice, to deal with a suitable formulation for the optimizer to be used. This means, for instance, including extra variables and constraints to simplify the coding task. This new formulation should be appropriate for most optimizers, though the code can be specific for each one due to the particular problem structure they require. However, the two optimizers up to now used, Minos 5.5 and Snopt 5.3, share the same internal problem description structure. In this case the same model generation routines can be applied to both solvers.

Since both Minos 5.5 and Snopt 5.3 permit receiving the model structure through vectors (this is the way HTCOOR is implemented), it is not compulsory to provide names for all the variables and constraints involved. However, and to clarify this exposition (even for reading the solution file provided by either Minos or Snopt), we will detail in the next two subsections the (up to eight characters) names given to all the variables and constraints. The last subsection will show the structure of the Jacobian of the constraints.

2.3.1 Variables of the problem formulation

The following notation will be considered for variables—and constraints—names.

- `k` denotes the commodity (one character).
- `x` denotes the type of variable of the hydro network (one character). The different values of `x` are:
  - — `d`, `e`, `f` or `g` for discharges.
  - — `b`, `c`, `w` or `x` for pumpings.
  - — `v`, `o`, `p` or `q` for spillages.
  - — `m` for the variables that represent volumes at the reservoirs.
- `rrr` denotes the reservoir (three characters).
- `iii` refers to the interval (three characters).
- `ttt` denotes the thermal unit (three characters).
- `c` denotes the fuel type (one character).

All the variables are considered nonlinear. They can be grouped as follows:

1) Hydro network variables. These variables represent the topology of the hydraulic system. The name is of the form `xkrrriii`.
2) Hydro generation variables. These variables store the hydro generation for each commodity and interval. The name is of the form HD`kiii`.
3) Hydro energy inserted in the smoothed generation duration curve, together with the thermal generation. The name of these variables is of the form HI`tttiii` (hydro energy at interval `iii` inserted over thermal unit `ttt`).
4) Amount purchased by each thermal unit at each interval for each fuel type. The name is of the form C`ctttiii`.
5) Amount not purchased for each fuel type. The name is of the form N`c`.
6) Fuel not consumed at the last interval by each thermal unit. The name is R`cttt`.
7) Remaining fuel for the next interval, for a given thermal unit. The name is of the form Z`ctttiii` (this represents the available amount of fuel `c`, for the thermal unit `ttt`, at the end of interval `iii`).
8) Thermal energy generation variables, for each thermal unit, interval, and fuel type. The name is of the form G`ctttiii`.
9) Thermal power generation variables, for each thermal unit and interval. The name is of the form PG`tttiii`.
10) Stochastic hydro power generation variables, for each interval. The name is of the form PHD`iii`.

2.3.2 Constraints of the problem formulation

The constraints can be classified in nonlinear and linear. The nonlinear constraints are:

1) Constraints to compute the hydro generation for each interval and commodity (stored in variables HD`kiii`). The name of the constraints is of the form HD`kiii`. These constraints correspond to equations (1.6) of the model formulation.
2) Constraints to compute the power generation for each thermal unit and interval (stored in variables PG`tttiii`). The name of the constraints is of the form PG`tttiii`. These constraints correspond to equations (1.9) of the model formulation.
3) Constraints to guarantee the coverage of the generation duration curve at each interval. The name of the constraints is of the form CO`iii`. These constraints correspond to equations (1.11) of the model formulation.

The linear constraints considered in the problem formulation are:

1) Constraints to impose that the hydro generation must be the same than the hydro energy used in the coverage of the generation duration curve. The name of the constraints is of the form HINT`iii`. These constraints correspond to equations (1.7) and (1.8) of the model formulation.
2) Balance equations for the hydro network. The name of the constraints is of the form B`krrriii` (B`k...iii` for an artificial sink node), and they correspond to equations (1.2) of the model formulation.
3) Mutual capacity constraints of the arcs of the hydro network with respect to all the commodities. The name of the constraints is of the form T`xrrriii`, and they correspond to equations (1.3) of the model formulation.
4) Balance equations for the available and purchased fuel. The name of the constraints is of the form XC`c`. These constraints are not detailed in equations (1.2–1.11).
5) Balance equations for the thermal system (for each fuel, thermal unit and interval). The name of the constraints is of the form X`ctttiii`. These constraints correspond to equations (1.5).
6) Auxiliary constraints to guarantee that the thermal plus hydro generation at each interval is over a minimum value (to avoid numerical problems when evaluating the PE curves). The name of the constraints is of the form PGM`iii`. These constraints are not detailed in equations (1.2–1.11).
7) Auxiliary constraints to guarantee that the power generated by the hydro system (through the stochastic plus deterministic hydro generation) is always below the maximum available hydro power, at each interval. The name of the constraints is of the form PHD`iii`. These constraints are not detailed in equations (1.2–1.11).

2.3.3 Jacobian structure

Figure 2.2 shows the structure of the Jacobian of the matrix constraints. The columns represent the ten kind of variables considered in the problem formulation. The rows are related with the ten different kind of constraints of the problem. The first three rows correspond to the nonlinear constraints, and the last seven to the linear ones. An X at cell (i, j) denotes that the i-th type of variables appears in the j-th group of constraints (the particular structure for each cell is not provided).

Clearly, the sparsity pattern for each cell of the Jacobian represented in Table 2.2 can be very

	xkrrriii	HDkiii	HItttiii	Cctttiii	Nc	Rcttt	Zctttiii	Gctttiii	PGtttiii	PHDiii
HDkiii	X	X								
PGtttiii		X	X					X	X	X
COiii		X	X					X	X	X
HINTiii		X	X							
Bkrrriii	X									
Txrrriii	X									
XCc				X	X					
Xctttiii				X		X	X	X		
PGMiii		X							X	
PHDiii		X								X

Figure 2.2. Jacobian structure. The X's denote which groups of variables appear in each group of constraints. The pattern structure for each X is different.

different from a case to another, since it depends of the number of intervals, thermal units, reservoirs, etc., considered in the problem. For instance, Figure 2.3 represents the particular Jacobian structure for problem WW of Table 3.1 (see Section 3). This Jacobian has 3194 variables, 380 nonlinear and 1283 linear constraints, and 21001 nonzero elements.

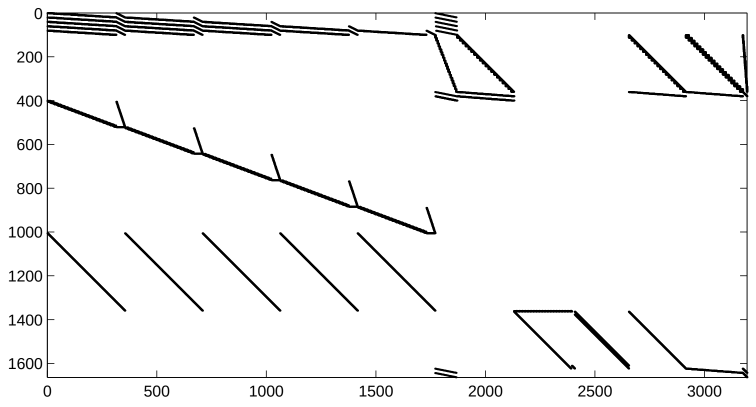

Figure 2.3. Jacobian structure for problem WW of Table 3.1.

2.4 Output data

From the solution vector returned by the optimizer it is possible to recover all the needed information about the model. However, in order to simplify this procedure, HTCOOR already computes the most relevant data (by unscaling and/or transforming some components of the solution vector). Among these we find:

- Objective function cost (in the original currency units that appeared in the input data).
- Problem dimensions of the optimization problem: number of variables, number of linear and nonlinear constraints, and number of nonzero elements (linear and nonlinear) of the constraints

matrix.

- Generation (in MWh) for each thermal unit at each interval.
- Total generation (in MWh) for each thermal unit for all the intervals
- Total cost (in the original currency units) due to the fuel consumed for each thermal unit for all the intervals.
- Energy purchased (in MWh) for each contract at each interval.
- Total energy purchased (in MWh) for each contract for all the intervals
- Total cost (in the original currency units) of the energy purchased for each contract for all the intervals.
- Total energy generated and purchased (in MWh) by the whole thermal+contract system.
- Total cost (in the original currency units) of the energy generated and purchased by the whole thermal+contract system.
- Deterministic hydro generation (in MWh) at each interval.
- Total deterministic hydro generation (in MWh) for all the intervals.
- Stochastic hydro generation (in MWh) at each interval.
- Total stochastic hydro generation (in MWh) for all the intervals.
- Energy to be purchased to an external company (in MWh) at each interval.
- Total energy to be purchased to an external company (in MWh) for all the intervals.
- Cost (in the original currency units) of the total energy to be purchased to an external company for all the intervals.

Up to now, HTCOOR writes all this information to a file, though it could be easily stored in a data warehouse.

2.5 Error codes

Several kind of errors are handled by the program. The current version just warns the user and finishes the execution. The different error codes and associated messages are listed below. Messages marked with a † could be adapted to fit with a particular data warehouse environment.

1) "Error: dimensions can not be obtained. Parameters data file is missing". †
2) "Error: dimensions can not be obtained. Fuels data file is missing". †
3) "Error: dimensions can not be obtained. Thermal units data is missing". †
4) "Error: dimensions can not be obtained. Hydro model data is missing". †
5) "Error: too many intervals. Increase MAXINT". MAXINT is the constant for the maximum number of intervals of the problem.
6) "Error: too many hydro commodities. Increase MAXNAT". MAXNAT is the constant for the maximum number of commodities considered in the problem.
7) "Error: too many fuels. Increase MAXCOMB". MAXCOMB is the constant for the maximum number of different fuels considered in the problem.
8) "Error: too many thermal units. Increase MAXTERM". MAXTERM is the constant for the maximum number of thermal units supported by the model.

9) "Error: a thermal unit uses more than the available number of fuels".
10) "Error: a thermal unit uses an unknown fuel".
11) "Error: in initial or final interval for some contract".
12) "Error: an inflows data file could not be opened for some reservoir". †
13) "Error: a commodity volumes data file could not be opened for some reservoir". †
14) "Error: file with coefficients of the fourth degree polynomial arc efficiency functions could not be opened".
15) "Error: dimension of vector of pointers to hydro part of Jacobian must be increased".
16) "Error: dimension of vector of pointers to thermal part of Jacobian must be increased".
17) "Error: Load file is missing". †
18) "Error: Not enough hours in load file". †
19) "Error: Hydro model data is missing".
20) "Error: Too many reservoirs [get_cdf_inflows]"
21) "Error: Not enough space reading inflows".
22) "Error: Can't save time division of horizon".
23) "Error: Can't read time division of horizon".
24) "Error: Dates don't match with length duration in file".
25) "Error: Can't read description of basin". †
26) "Error: Can't save file with hydro model data". †
27) "Error: Not enough space [getLDCshapes]".
28) "Error: Thermal park description file not found [get_thermal_park]". †
29) "Error: Unknown reservoir identifier".

2.6 Getting the code

HTCOOR can be obtained for academic and research purposes from

`http://www-eio.upc.es/~jcastro` ,

at the software entry. The distribution includes the source code for both the Minos 5.5 and Snopt 5.3 versions, as well as all the problem instances of the computational results section (see Section 3). A copy of either Minos 5.5 or Snopt 5.3 are required to run the package (they are not included with the distribution).

Slight modifications to the original code should permit solving the optimization problem through alternative solvers. In this sense, HTCOOR can be a good test for evaluating new optimization methods.

3 Computational results

3.1 Test cases

The current release of the code HTCOOR has been evaluated through a set of pseudo-real test cases. All of them have been obtained by defining different configurations for the thermal and hydraulic subsystems of a basic test case. This basic test case is a slight modification of a real one provided by the Spanish power utility Iberdrola S.A. The remaining instances are a subset of this real problem.

Table 3.1 gives the characteristics for each instance. We grouped them into two subsets: small problems (first four rows) and bigger ones (last five rows). Problem AA corresponds to the real case provided by Iberdrola, and, by its size, it is a good representative of the type of problems to be solved with HTCOOR. The meaning of the columns of the table is:

- Problem: name of the problem.
- #Int.: number of intervals of the planning horizon.
- #Therm.: number of thermal units.
- #Resvrs.: number of reservoirs.
- #Disch.: total number of discharge arcs considered in the hydraulic subsystem.
- #Comm.: number of commodities used to model stochasticity in the hydraulic subsystem.
- #Vars.: number of variables of the optimization problem.
- #Linc.: number of linear constraints of the optimization problem.
- #Nonlinc.: number of nonlinear constraints of the optimization problem.
- #Nnz.: number of nonzero elements (linear and nonlinear) of the constraints matrix.

Table 3.1. Characteristics of the test cases

Problem	#Int.	#Therm.	#Resvrs.	#Disch.	#Comm.	#Vars.	#Linc.	#Nonlinc.	#Nnz.
pp	3	13	2	3	3	250	86	51	1707
mm	6	13	3	6	5	685	246	114	4530
nn	6	13	3	6	5	685	246	114	4530
zz	8	13	1	2	5	687	200	152	4738
WW	20	13	6	12	5	3194	1283	380	21001
DD	40	13	6	12	5	6414	2563	760	42281
BB	15	13	41	85	5	10314	5173	285	62286
MM	12	70	41	85	5	11634	4819	912	171162
AA	33	70	41	85	5	32340	13303	2508	473982

3.2 Computational results

The instances presented in the previous section have been solved with both Minos 5.5 [2] and Snopt 5.3 [3]. Tables 3.2 and 3.3 show the results obtained with each solver. The *penalty parameter* was set to 50 for Minos 5.5, to deal with the nonlinearities of the functions [2]. The parameter *scale option* was set to 1 in all the Snopt 5.3 runs, otherwise even for the simplest problems the algorithm stopped at a nonoptimal and nonfeasible point with a "the current point cannot be improved" message. The default values have been used for the rest of parameters (both for Minos 5.5 and Snopt 5.3), except for some few executions which are clearly marked in the tables.

For each problem the following information is provided:

- Problem: name of the problem.
- Status: exit status provided by the solver. This column can take the values "Optimal" (optimal solution point has been obtained), "Infeasible" (problem reported as infeasible by the solver) or "Too many iter." (too many iterations).
- #iter: overall number of minor iterations performed by the solver in the solutions of the subproblems. For Minos 5.5 they correspond to the minimization of a nonlinear—augmented Lagrangian—function subject to linear constraints, whereas for Snopt 5.3 they are related to quadratic programming subproblems.
- f^*: value of the objective function at the optimal solution obtained (in the same currency units used for the input data).
- CPU: CPU seconds required to perform the run. All runs were carried out on a Sun/Ultra2 2200 workstation with 200 MHz clock, 256 Mbytes of main memory, ≈68 Mflops Linpack, 14.7 Specfp95 and 7.8 Specint95.

Table 3.2. Results obtained with Minos 5.5

Problem	Status	#iter	f^*	CPU
pp	Optimal	543	8.704343672e+9	0.41
mm	Optimal	1101	1.8336647408e+10	2.57
nn	Optimal	775	1.8462822133e+10	1.32
zz	Optimal	916	2.4208240477e+10	1.49
WW	Optimal	6769	2.9152512944e+10	81.52
DD	Optimal	10896	2.7910024022e+10	302.50
BB	Optimal	72640	2.0327043893e+10	4521.08
MM	Infeasible	281830	—	7756.70
AA	Optimal	831183	1.13014855496e+11	108916.51

Table 3.3. Results obtained with Snopt 5.3

Problem	Status	#iter	f^*	CPU
pp	Optimal	700	8.704343596e+9	0.90
mm	Optimal	2615	1.8349202899e+10	8.02
nn	Optimal	2833	1.8471176724e+10	8.58
zz	Optimal	1352	2.4203322733e+10	2.67
WW$^{(*)}$	Optimal	32552	2.9154089151e+10	947.21
DD	Optimal	79596	2.7965381364e+10	15600.22
BB	Optimal	192137	2.0348450144e+10	195936.96
MM	Optimal	81060	1.96798275462e+11	301440.21
MM$^{(**)}$	Optimal	74744	1.96827415432e+11	160868.84
AA$^{(***)}$	Too many iter.	1500000	1.14929788079e+11	704391.3

$^{(*)}$ real workspace was increased

$^{(**)}$ problem solved with *scale option* parameter not set to 1

$^{(***)}$ the code got stuck; it stopped in a feasible but nonoptimal point

It is worth to note that the parameters had to be specifically tuned for some problems. In general we can say that no combination of values—of those tested—for the Minos and Snopt parameters guaranteed the solution of all the problems. This made the solution of the different instances a nontrivial task—usually through a trial-and-error adjusting parameter procedure. Moreover, the efficiency of the code was quite different depending of the values of the parameters (see, e.g., rows MM and MM$^{(**)}$) of Table 3.3). Minos solved all the problems but case MM, which reported as infeasible. However, this problem is in fact feasible, as shown by the Snopt run. Snopt also failed in the solution of only one problem, instance AA, where it seemed to get stuck, moving from one major iteration to another without performing any minor iteration. For problem AA Snopt stopped in a feasible but nonoptimal point. For problem WW and Snopt, and following one of the author's—P.E. Gill— suggestions, we considerably increased the real workspace used by the solver. Otherwise, Snopt got stuck with a final "too many major iterations" message.

From Tables 3.2 and 3.3 we can also conclude than Minos was fairly more efficient than Snopt in the solution of the model implemented in HTCOOR. As for the number of iterations performed, Snopt performed many more than Minos for all the cases but problem MM. Looking at the f^* columns, we see that here is no significative difference between Minos and Snopt, and both packages returned similar objective function values.

From these results it can be stated that the model implemented in the HTCOOR package seems to be a difficult one for nonlinear solvers based on projected Lagrangians (Minos) and Sequential Quadratic Programming (Snopt) algorithms. Moreover, it requires the tuning of some of the default settings of these algorithms to work properly. The model can thus be useful to evaluate alternative optimization algorithms (e.g., interior-point methods for nonlinear nonconvex problems). Having a more reliable code would permit solving larger and more difficult problems, even extending the model for the auction market system. In fact, one of the reasons that stopped this latter extension was the unavailability of a robust and efficient solver for this model.

References

[1] J.A. González and J. Castro, *The long-term hydrothermal coordination model implemented in the HTCOOR package*, Technical Report DR2000-11, Statistics and Operations Research Dept., Universitat Politècnica de Catalunya, 2000. Available from `http://www-eio.upc.es/~jcastro`.

[2] B.A. Murtagh and M.A. Saunders, "MINOS 5.0. User's guide", Dept. of Operations Research, Stanford University, USA, 1983.

[3] P.E. Gill, M.A. Saunders and W. Murray, "SNOPT: An SQP algorithm for large-scale constrained optimization", Report NA 97-2, Department of Mathematics, University of California, San Diego, 1997.

Appendix A

Description of the internal variables of the HTCOOR package

A.1 Maximum dimensions

- `MAXINT`: maximum number of intervals (default: 200)
- `MAXTERM`: maximum number of thermal units plus contracts (default: 300)
- `MAXCOMB`: maximum number of fuels (default: 100)
- `MAXNAT`: maximum number of hydro commodities (types of inflows) (default: 10)
- `MAXNEB`: maximum number of reservoirs (default: 100)
- `MAXNAR`: maximum number of hydro arcs (including discharge + spillage + transshipment arcs) (default: 300)
- `MANTEN`: for maintenances (default: 100)
- `MAXLOADS`: for observations of load in an interval ($\simeq$ hours/interval) (default: 1500)

A.2 General data about the model (common `general_data`)

- `nint`: number of intervals
- `nhores(nint)`: duration of each interval (hours)
- `ppunta(nint)`: maximum load (MW)
- `monot()`: discretization of l.d.c.
- `nforma(nint)`: type of l.d.c
- `nat`: number of hydro commodities (types of inflows)
- `probhd(nat-1)`: probabilities for each type of inflow
- `phdmax`: maximum hydraulic power (MW)
- `escal_potencia`: scaling factor for power
- `escal_hores`: scaling factor for time
- `escal_energia`: scaling factor for energy
- `ordre_preu`: scaling factor for prices

A.3 Thermal system data (common `thermal_system`)

- `nterm`: number of thermal units
- `nomterm(nterm)`: name of the thermal units
- `idtem(nterm)`: three character identifier (obsolete, maintained for Minos debugging)
- `pmaxterm(nterm)`: maximum power for each thermal unit (MW)
- `pminterm(nterm)`: minimum power for each thermal unit (MW)
- `te_limits_comb(nterm)`: tells if thermal units have fuel limitations
- `pmaxlim(nterm)`: approx. maximum power in case of fuel limitations
- `panes(nterm)`: failure probability for each thermal unit ([0,1])
- `nartterm(nterm)`: number of operational fuels of each thermal unit
- `artterm(t,f)`: types of fuel used for each thermal unit
- `rendiment_g(t,f)`: efficiency of thermal unit t with fuel f (in %)
- `potcomprar(t,f)`: maximum purchase of fuel f by thermal unit t (MWh)
- `potguardar(t,f)`: maximum storage of fuel f by thermal unit t (MWh)
- `es_contracte(nterm)`: tells if it is a thermal unit or a contract
- `cost_lin(nterm)`: linear cost for contracts
- `cost_quad(nterm)`: quadratic cost for contracts
- `ini_contract(nterm)`: initial interval of contract availability
- `fi_contract(nterm)`: final interval of contract availability
- `ncomb`: number of fuels
- `nomcomb(ncomb)`: names of the fuels
- `preu(ncomb)`: price of each fuel (Euro/MWh)
- `totalcomb(ncomb)`: maximum availability for each fuel (MWh)
- `apreciacio(ncomb)`: price apreciation (1= no apreciation/deprec.)
- `preuext`: price of external energy (Euro/MWh)
- `nthrmarticles`: total number of fuels used for all the thermal units (contracts use one fuel)
- `nman`: number of maintenances
- `minimcomb(ncomb)`: minimum availability for each fuel (MWh)
- `thu_man(nman)`: name of thermal unit with maintenance
- `pmx_man(nman)`: maximum power allowed during maintenance
- `from(nman)`: initial week of maintenance
- `until(nman)`: final week of maintenance

A.4 Bézier curves information for model generation and optimization (common `bezier_curves_gen_opt`)

- `tbase(nint)`: base time (hours) at each interval
- `pmin(nint)`: minimum power related to base time
- `max_ho(nint)`: maximum deterministic water at each interval
- `min_gt(nint)`: thermal generation decrease per interval permitted over maximum power generation to avoid problems with PEC (Bézier)
- `energia(nint)`: total energy under the PEC
- `e0a,e0b,e0c (nint)`: external energy function parameters for each interval

- `t0a,t0b,t0c (nint)`: function parameters of the time related to the external energy, for each interval
- The following data are local to the optimization stage. They could be converted to local variables and removed from here.
  - `MAXBEZ`: number of PEC points that define the parametric Bézier curve
  - `e(MAXBEZ)`: energy component of the Bézier points
  - `p(MAXBEZ)`: power component of the Bézier points
  - `ehd(MAXBEZ)`: derivatives of e() with respect to hd (hydro energy)
  - `phd(MAXBEZ)`: derivatives of p() with respect to hd (hydro energy)
  - `epg(MAXBEZ)`: derivatives of e() with respect to pg (thermal energy)
  - `ppg(MAXBEZ)`: derivatives of p() with respect to pg (thermal energy)
  - `ee(MAXBEZ)`: coefficients of the 3rd degree polynomial that represents the energy component of the Bézier curve
  - `pp(MAXBEZ)`: coefficients of the 3rd degree polynomial that represents the power component of the Bezier curve
  - `dehd(MAXBEZ)`: derivatives of ee() with respect to hd (hydro energy)
  - `dphd(MAXBEZ)`: derivatives of pp() with respect to hd (hydro energy)
  - `depg(MAXBEZ)`: derivatives of ee() with respect to pg (thermal energy)
  - `dppg(MAXBEZ)`: derivatives of pp() with respect to pg (thermal energy)

A.5 Hydro system data (commons `hydro_system_read_data` and `hydro_sys_generate_-and_optimize`)

- The following data should be read from the input files/data warehouse.
  - `neb`: number of reservoirs
  - `nar`: number of arcs (discharge+spillage+transshipment)
  - `ncc`: number of discharge arcs
  - `nvv`: number of spillage arcs
  - `ntv`: number of transshipment arcs
  - `conca(nar,2)`: hydro network topology (for each arc, origin and destination reservoirs)
  - `capmut(nar,nint)`: mutual capacities of the arcs in m^3/s
  - `capemb(neb,nint)`: reservoirs capacities in Hm^3
  - `vlx(neb)`: reservoirs capacities again in Hm^3 (auxiliary vector)
  - `aportacio(nat,neb,nint)`: reservoir inflows in m^3/s
  - `nomembas(neb)`: names of the reservoirs
  - `embas(neb)`: three character identifiers for the reservoirs
  - `tiparc(nar)`: types of the arcs (spillage, discharge or transsh.)
  - `limarc(nat,nar,nint)`: arc bounds for commodity and interval in m^3/s
  - `limemb(nat,neb,nint)`: reservoir capacity bounds for commodity and interval in Hm^3/s
  - `volini(nat,neb)`: initial volumes for commodity and reservoir in Hm^3
  - `volfi(nat,neb)`: final volumes for commodity and reservoir in Hm^3
  - `minima(nar,nint)`: minimum flows at arcs in m^3/s

— `minime(neb,nint)`: minimum capacities at reservoirs in Hm^3/s
— `sbx(nar)`: maximum net head
— `ctx(neb)`: maximum elevation
— `retard(nint)`: arcs delays
— `cvb,cvl,cvq,cvc,cvcu (neb)`: coefficients of the fourth degree polynomial head volume function
— `ehe5,ehe4,ehe3,ehe2,ehe1,ehe0 (ncc)`: coefficients of the 2/4 degree polynomial arc efficiency function

- The following data are created and used by HTCOOR during the model generation and optimization stages, respectively

— `nax`: number of arcs in the multiinterval (replicated) hydro network
— `nnx`: number of nodes in the multiinterval (replicated) hydro network
— `pvi`: penalty parameter for spillages at non-full reservoirs
— `xarxa((nar+neb)*nint-neb,2)`: topology of the replicated network (origin and destination reservoir for each arc)
— `tiparx(nar+neb)*nint-neb)`: arc types of the replicated network
— `mnk(nar*nint+neb*(nit-1))`: vector of origin nodes for all the arcs of the multiinterval hydro network
— `ipl(neb*nint+1)`: vector of pointers in mnk() for each node, to the list of arcs that arrive to this node
— `rin(neb*nat)`: initial volumes accumulated by commodity, for all the reservoirs
— `rfl(neb*nat)`: final volumes accumulated by commodity, for all the reservoirs
— `can(nar*nint)`: minimum flows of the discharge and pumping arcs for all the intervals
— `jcl(nar)`: vector of codes (1,2,3) of the discharge and pumping arcs
— `jeo(nar)`: vector of origin reservoirs of the discharge and pumping arcs
— `jed(nar)`: vector of destination reservoirs of the discharge and pumping arcs
— `snq(nar)`: vector of head differences between discharges
— `jll(nar)`: vector for identifying pumping arcs between those of type b, s or d
— `ivo(nar*nint),lvo(nar*nint)`: indices to arcs of initial and final volumes of the origin reservoirs for discharge and pumping arcs in the multiinterval network.
— `ivd(nar*nint),lvd(nar*nint)`: indices to arcs of initial and final volumes of the destination reservoirs for discharge and pumping arcs in the multiinterval network.
— `jca(nar*nint)`: vector of numbers of the arcs in the multiinterval network of the arcs b, s or d
— `cxv(nar)`: vector of maximum capacities of the origin reservoirs for the spillage arcs
— `iov(nar)`: vector of origin reservoirs for the spillage arcs
— `ivv(nar*nint)`: vector of spillage arcs in the multiinteral network
— `mfv(nar*nint)`: vector of arcs with final volumes of the origin reservoirs of the spillage arcs, in the multiinterval network
— `von(neb*(nint+1))`: vector derived from minime() used during the optimization stage
— `elm(neb),eqt(neb),eqd(neb),ecq(neb),ecua(neb)`: auxiliary vectors for computing derivatives during optimization phase at funcon()
— `jabom(nar)`: points to information about pumping arcs in variable "salto"
— `salto(6*nar)`: coefficients for evaluating net jump between reservoirs (according

to IBD expression)

A.6 Other variables

- Variables to store optimization model in Minos/Snopt format (common `minos_vars`)
  - `n`: number of variables
  - `m`: number of constraints
  - `nb`: number of variables+slacks (nb=n+m)
  - `ne`: number of nonzero entries in Jacobian (including linear and nonlinear constraints)
  - `nncon`: number of nonlinear constraints
  - `nnobj`: number of nonlinear objective function variables
  - `nnjac`: number of nonlinear Jacobian variables
- Variables to store data (indices and auxiliary values) required by the funobj and funcon routines (common `funobj_funcon`)
  - `sumpmaxterm(nint)`: sum of the maximum power for all the thermal units
  - `tipuscomb(nterm*ncomb)`: codes for types of fuels used by each thermal unit
  - `c(nint)`: indices to variables of purchase of fuel at each interval
  - `r(neterm*ncomb)`: indices to residual fuel at last interval for each thermal unit and fuel used
  - `hn(nint)`: indices to variables of hydraulic generation HD`kiii`
  - `hl(nint)`: indices to variables of inserted hydraulic generation HI`tttiii`
  - `pg(nint)`: indices to variables PG`tttiii` of power generated per thermal unit and interval
  - `t(nint)`: indices to variables G`ctttiii` of generation per thermal unit and interval
  - `iphd(nint)`: indices to variables PHD`iii` of power generated by stochastic water
  - `elemblochd`: see routine crea_indexs_hd of the package for an explanation
  - `p_ijachd4, p_ijachd8`: pointers to ijachd() array (for addresses of 4 and 8 bytes).
  - `p_ijacterm4, p_ijacterm8`: pointers to ijacterm() array (for addresses of 4 and 8 bytes).
  - `4_bytes_adresses`: boolean to decide whether 4 or 8 bytes addresses are being used.
  - `maxjac`: stores an overestimation of the number of elements in ijachd() and ijacterm()
  - `inixpgt`: index to beginning of PG`tttiii` variables in solution vector x(). Used to reoptimize adjusting lower bounds for variables PG`tttiii`, if necessary.