

AlphaECP

Tapio Westerlund and Toni Lastusilta, Åbo Akademi University, Finland, twesterl@abo.fi

Contents

1	Introduction	1
1.1	Licensing and software requirements	1
1.2	Running GAMS/AlphaECP	2
2	GAMS/AlphaECP Output	2
3	Notes about Options	5
4	GAMS/AlphaECP Options	6
4.1	Basic options	6
4.2	Algorithmic options for advanced users	6
4.3	MIP Solver related options	6
4.4	NLP Solver related options	6
5	Detailed Descriptions of AlphaECP Options	7
6	FAQ	11
7	AlphaECP References	11

1 Introduction

GAMS/AlphaECP is a MINLP (Mixed-Integer Non-Linear Programming) solver based on the extended cutting plane (ECP) method. The solver can be applied to general MINLP problems and global optimal solutions can be ensured for pseudo-convex MINLP problems.

The ECP method is an extension of Kelley's cutting plane method which was originally given for convex NLP problems (Kelley, 1960). The method requires only the solution of a MIP sub-problem in each iteration. The MIP sub-problems may be solved to optimality, but can also be solved to feasibility or only to an integer relaxed solution in intermediate iterations. This makes the ECP algorithm efficient and easy to implement. Further information about the underlying algorithm can be found in Westerlund T. and Pörn R. (2002). Solving Pseudo-Convex Mixed Integer Optimization Problems by Cutting Plane Techniques. *Optimization and Engineering*, 3, 253-280.

The GAMS/AlphaECP algorithm has been further developed by introducing some additional functionality. A NLP solver can be called at MIP solutions which improve AlphaECP in finding feasible and accurate solutions, especially for MINLP problems, containing mainly of continuous variables. Furthermore, a heuristic that reselects cutting planes during the iteration procedure can be used to improve the capability of solving non-convex problems.

1.1 Licensing and software requirements

In order to use GAMS/AlphaECP, users will need to have a GAMS/AlphaECP license. Additionally a licensed MIP solver is required for solving the mixed integer sub-problem. If the NLP option is used a licensed NLP solver is additionally required.

1.2 Running GAMS/AlphaECP

GAMS/AlphaECP solves MINLP models. If GAMS/AlphaECP is not specified as the default solver for these models, it can be invoked by issuing the following command before the solve statement:

```
option minlp=alphaecp, miqcp=alphaecp;
```

In principle, GAMS/AlphaECP can also handle NLP models, but is more suitable for MINLP problems. Especially in combination with an NLP solver it can find solutions the NLP solver by itself does not find. In this case it acts as a good starting point generator. If you want to solve NLPs with GAMS/AlphaECP you need to *trick* the GAMS system by solving your NLP as an MINLP:

```
solve mynlpmodel minimizing obj using minlp;
```

Throughout the progress of GAMS/AlphaECP and at the end of the algorithm, constraint violations are reported. The violation reported is for the non-linear constraints only. The violation of the linear constraints is subject to the feasibility tolerance of the MIP/NLP solver.

2 GAMS/AlphaECP Output

The log output below is obtained for the MINLP model fuel.gms from the GAMS model library:

GAMS Rev 238 Copyright (C) 1987-2011 GAMS Development. All rights reserved

```
-----
                Welcome to Alpha-ECP v2.09.01
MINLP Problem Solver using the Extended Cutting Plane Approach.
Method development - T.Westerlund, Abo Akademi University, FIN
Algorithm implementation - T.Lastusilta, Abo Akademi University, FIN
Westerlund Tapio and Poern Ray (2002). Optimization & Engineering, 3, 253-280
-----
Minimization problem: "fuel.gms"
The GAMS-model has in total 39 elements of which 15% are non-linear(NL)
included in 16 constraints of which 25% are NL
The NL constraint signs: =E=(3), =G=(1), =L=(0)
The number of variables in NL elements are 6 from a total of 16
variables: Continuous(13), Binary(3), Integer(0)
-----
Using following settings
AlphaECP option file                optfile=0
Time limit for AlphaECP (in seconds)  reslim=1000
SolveLink for NLP and MIP sub-solver  solvelink=5
Cutting plane strategy (0-3)         CUTdelcrit=3
Cut generation pace                  CUTnrcuts=0
Updating multiplier if MIP is infeasible ECPbeta=1.3
Write encountered solutions to gdx files ECPdumpsol=0
Updating multiplier when verifying solution ECPgamma=2
Maximum number of AlphaECP iterations ECPiterlim=-1
Level of AlphaECP output to statusfile (0-4) ECPloglevel=0
User specified start-point (0-3)      ECPstart=3
Return solution (1.MIP/2.NLP/...)    ECPretsol=2
AlphaECP strategy (1-5)              ECPstrategy=2
Upper limit of considered MIP solutions per MIP call MIPnrsols=50
```

```

Relative MIP gap in intermediate sub-problems (0->1.0) MIPoptcr=1.00
Initial MIPoptcr interval before MIPoptcr reduction MIPoptcrlim=200
Strategy for multiple MIP solutions MIPsolstrat=1
MIP solver for sub-problems MIPsolver=cplex
NLP strategy. Inactive:0 Active strategy:1-5 NLPcall=5
NLP solver call at next (incremental) iteration NLPcalliter=0
NLP time limit per call (in seconds or auto=0) NLPreslim=30
NLP solver for sub-problems NLPsolver=conopt
Constraint tolerance TOLepsg=0.001
Distance tolerance for a new linearization TOLepsz=0.1
Gradient tolerance TOLgrad=1e-006
Infinity bound (MIP variable bound) TOLinfbnd=1e+010

```

```

-----
Itera Stepcode, Number Point Alpha OPT Movement Viol Maximum MIPobjval
tion Problems of Cuts usage Upd. CR Norm Cons Violation
Start-point: NL constraint (1) infeasible
0 H 0 0 0 1 0 4 1.8E+003 NA
1 SAFGI 1 1 1 1 9.3E+003 0 0 8566.12
1 FOUND SOLUTION: 8566.12 (NLP) in 1 sec.

2 SAFH 1 1 0 1 6.6E+003 4 1.8E+003 4844.02
3 SAFH 3 2 0 1 9.3E+003 4 1.8E+003 4844.02
4 SAH 4 3 0 1 3.6E+003 4 9E+002 4844.02
5 SAFH 7 4 0 1 3.4E+003 3 1.2E+003 10606.3
6 SAH 9 5 0 1 2.8E+003 4 5.2E+002 6130.21
7 SAH 12 6 0 1 2.3E+003 4 3.3E+002 7098.42
8 SAFH 15 7 0 1 3.2E+003 4 3E+002 7480.99
9 SAFH 18 8 0 1 4.1E+003 4 3.2E+002 7649.91
10 SAH 19 9 0 1 1.5E+003 4 1.9E+002 7657.39
11 SAH 22 10 0 1 1.5E+003 4 1.3E+002 7964.39
12 SAH 26 11 0 1 2E+003 4 97 8154.14
13 SAH 30 12 0 1 1.3E+003 4 58 8287.43
14 SAH 33 13 0 1 7.4E+002 4 29 8405.27
...
88 SAH 129 87 0 1 3.9 2 0.0013 8566.11
89 SAH 130 88 0 1 4.7 2 0.0011 8566.11
90 SAH 131 89 0 1 4.5 2 0.001 8566.11
91 SAH 132 90 0 1 3.5 1 0.001 8566.11
92 SAFI 133 91 41 1 2.7 0 0.00075 8566.12
92 FOUND SOLUTION: 8566.12 in 3 sec.

93 SAI 134 92 35 1 4.3E-009 0 0.00075 8566.12
...
107 SAJ 144 102 0 0.1 0 0 0.00075 8566.12
108 AJ 144 102 0 0 0 0 0.00075 8566.12
108 Pointusage 6/90 Cutusage 15/341 ( 0,135 )
109 AH 27 18 0 0 4.1 1 0.0011 8566.12
110 AIJ 28 19 0 0 3 0 0.00075 8566.12
AlphaECP: Iteration procedure terminated normally

```

```

-----
Problem : fuel.gms
Solver Status : Normal Completion
Model Status : Locally Optimal
Exit comment : No Issues
Final solution : NLP
Objective value : 8566.1189616876654

```

```

Max constraint (1)      : -0
Alternative solution   : MIP
Alt. objective value   : 8566.1150498670522
Max constraint (4)     : 0.00075412533010421612
Time used (seconds)    : 2.98
Time limit (seconds)  : 1000
Iterations used        : 110
Iteration limit        : -1
Function evaluations   : 1724
Gradient evaluations   : 359
Domain violations      : 0
Gradients unusable    : 0
Alphamax bound violations : 0
ECP time usage         : 1.0 %
NLP time usage         : 3.7 %
MIP time usage         : 95.2 %
Optimal/total MIPs    : 3/110
NLP solver calls       : 7

```

In every iteration, information of the MIP problem and the modifications to the MIP problem, is given in 10 columns. Here is a description of the different columns:

Iteration: Iteration identifier.

Stepcode, Problems: Letter for what actions were taken in this iteration, i.e. MIP problem modifications before the next iteration.

A: MIP solver feasible.

B: MIP solver feasible after moving cutting planes, i.e. alpha update.

C: MIP solver feasible after moving cutting planes close to their generation point. The movement is done to make it easier to satisfy nonlinear equality constraints.

D: Line search was successful (in *ECPstrategy 3*).

E: Line search failed (in *ECPstrategy 3*).

F: A NLP solver was called.

G: Found a MINLP solution.

H: Added linearization(s) to the next MIP problem.

I: Updated alpha values and possibly added linearizations.

J: All cutting planes are valid underestimators for the pseudo-convex constraints, except for the nonlinear objective function constraint.

K: The nonlinear objective function constraint value and MIP solution value differ more than *epsilon_f*. A

linearization was done to reduce the difference (in *ECPstrategy 3*).

L: Removed all temporal linearizations.

M: Domain violation(s), some of the constraint could not be evaluated.

N: Some cutting plane(s) could not be generated because of gradient problems.

O: No cutting planes could be generated.

P: Reselecting cuts because cutting planes are repeatedly moved close to their generation point.

Q: Added temporal linearization(s).

R: Failed to add temporal linearization(s).

S: MIP solver strategy to find encountered solutions selected.

T: MIP solver strategy to require *MIPnrsols* solutions selected.

U: MIP solver strategy to require *MIPnrsols* solutions with a *MIPoptcr* ≤ 0.2 selected.

Number of Cuts: The number of cutting planes the solved MIP problem had.

Point usage: Number of points used to generate the cuts in the solved MIP problem.

Alpha Upd.: The number of times the alpha values has been increased.

OPTCR: Requirement of the relative distance to the relaxed MIP solution for the current MIP solution.

Movement Norm: The Euclidean norm of the current and previous MIP solution.

Viol Cons: Number of unsatisfied (violating) nonlinear constraints.

Maximum Violation: The most violating nonlinear constraint value.

MIPobjval: MIP objective variable value.

NLobjval: **MIPobjval** is replaced with the nonlinear objective function value, **NLobjval**, when *ECPstrategy 3*) is used.

The cut reselection heuristic is called in the following cases:

- 1) If the MIP solver would otherwise return infeasible.
- 2) When the violation is not reducing, but the cutting planes are repeatedly moved close to their generation point.
- 3) When the violation is not reducing and domain violations are repeatedly encountered.

The heuristic reselects cutting planes in different ways, but always ensures that the same point can not be found twice. When the cut reduction heuristic is called a printout is given, here is an example:

```
Pointusage      6/90      Cutusage      15/341      (      0,135      )
```

Pointusage informs how many points of all usable points have been used to generate the cutting planes. **Cutusage** tells how many cuts of all usable cuts have been used. The first number in (0,135) tells how many cuts is required by the user, see *CUTnrcuts* and the second number gives the sum of added and removed cuts, i.e. a measure of how much the MIP problem has been modified. AlphaECP may fix some cuts and remove points and cuts during the cut reselection procedure in order to save memory.

At the end of each solve AlphaECP gives a summary with Problem, Solver Status, Model Status, etc. Note the following lines **Exit comment**, **Domain violations**, **Gradients unusable** and **Alphamax bound violations**. **Exit comment** may give further information than solverstatus on why the solution procedure stopped. **Domain violations** (function evaluation failed) or **Gradients unusable** (all gradients $< TOLgrad$) might be caused by poor variable bounds. **Alphamax bound violations** inform how many times an alphamax value was calculated to be $> 10^{154}$ and was reset to 10^{154} .

3 Notes about Options

To instruct AlphaECP to read an option file use **ModelName.OptFile = 1**. The name of the option file is `alphaecp.opt`, see further information from the GAMS manual.

The following information is worth to notice when you are interested of AlphaECP options. A linearization of a nonlinear constraint is called a cutting plane or cut. Here a point refers to the variable levels. Global optimality can be guaranteed for pseudo-convex problems, however, if the objective variable is in a nonlinear constraint and pseudo-convex then *ECPstrategy* ≥ 3 needs to be used to guarantee global optimality. Recall that already one non-linear equality constraint makes a problem non-pseudoconvex, hence also non-convex. The basic options might impact significantly on the solution procedure and the best values are likely to be problem specific. The user is therefore encouraged to try different values for the basic options.

4 GAMS/AlphaECP Options

4.1 Basic options

CUTnrcuts	Cut generation pace
MIPnrsols	Upper limit of considered MIP solutions per MIP call
MIPsolstrat	MIP solution collection strategy
MIPsolver	MIP solver for sub-problems
NLPsolver	NLP solver for sub-problems
reslim	Time limit for AlphaECP (in seconds)

4.2 Algorithmic options for advanced users

CUTdelcrit	Cutting plane strategy
ECPbeta	Updating multiplier if MIP is infeasible
ECPdumpsol	Write encountered solutions to.gdx files
ECPgamma	Updating multiplier when verifying solution
ECPiterlim	Maximum number of AlphaECP iterations
ECPloglevel	Level of AlphaECP output to statusfile
ECPpcostrategy	Pseudo-convex objective function strategy
ECPretsol	Return solution (1.MIP/2.NLP/3.QUALITY/4.PERFORMANCE)
ECPstart	User specified start-point
ECPstrategy	AlphaECP strategy
solvelink	Solverlink for NLP and MIP sub-solver
TOLepsf	Pseudo-convex objective function termination tolerance
TOLepsg	Constraint tolerance
TOLepsz	Distance tolerance for a new linearization
TOLgrad	Gradient tolerance
TOLinfbd	Infinity bound (MIP variable bound)

4.3 MIP Solver related options

MIPloglevel	Level of MIP solver output
MIPoptcr	Relative MIP gap in intermediate sub-problems
MIPoptcrlim	Initial MIPoptcr interval before MIPoptcr reduction
MIPoptfile	Option file for MIP sub-solver

4.4 NLP Solver related options

NLPcall	NLP strategy
NLPcalliter	NLP solver call at next (incremental) iteration
NLPplimsameint	NLP call after a number of recurring integer solutions
NLPloglevel	Level of NLP solver output
NLPreslim	NLP time limit per call

5 Detailed Descriptions of AlphaECP Options

CUTdelcrit (*integer*) Cutting plane strategy

(*default = 3*)

- 0 Do not remove any valid cuts.
- 1 As 0 and allow temporary cuts at semirandom points if normal cuts can not be made.
- 2 Allow temporary cuts and cut reselection, and use memory to save points and cuts.
- 3 As 2 and call the reselection heuristic before termination to improve the solution.

CUTnrcuts (*real*) Cut generation pace

The number of linearizations that are generated during an iteration can be chosen by AlphaECP, proportional to the number of violating constraints or be determined by a fixed amount. Furthermore, the cut reselection $CUTdelcrit \geq 2$ adds cuts to the problem so that the requested cut generation pace is taken in consideration.

(*default = 0*)

- 0 Let AlphaECP decide.
- $0 < n < 1$ Number of linearizations = $n \times$ the number of linearizations that is possible to generate.
- > 1 Specifies the number of linearizations to generate.

ECPbeta (*real*) Updating multiplier if MIP is infeasible

In case of an infeasible MIP solution, the invalid cuts are updated with the *ECPbeta* multiplier.

(*default = 1.3*)

ECPdumpsol (*integer*) Write encountered solutions to gdx files

(*default = 0*)

- 0 No.
- 1 Solutions that the NLP solver found.
- 2 Solutions that the NLP or MIP solver found.

ECPgamma (*real*) Updating multiplier when verifying solution

If a MINLP solution is obtained but some cuts are not valid underestimators, then they are updated with the *ECPgamma* multiplier in order to make them into valid underestimators.

(*default = 2.0*)

ECPiterlim (*integer*) Maximum number of AlphaECP iterations

This is the maximum number of iterations given to AlphaECP to perform the optimization. Value -1 deactivates the AlphaECP iteration limit.

(*default = -1*)

- 1 No limit.
- ≥ 0 Specifies an iteration limit.

ECPloglevel (*integer*) Level of AlphaECP output to statusfile

(*default = 0*)

- 0 No additional output to statusfile.
- 1 Report solutions. Report all encountered solutions with their corresponding variable levels.
- 2 Report main actions at iteration level (available for minimization problems).
- 3 Report main actions at linearization level (available for minimization problems).

- 4 Full reporting. Report the main actions taken, the linearizations, function values, and solution points for every iteration and line search details (available for minimization problems).

ECPpcostrategy (*integer*) Pseudo-convex objective function strategy

(default = 3)

- 1 Remove support. Remove old support planes when a new pseudo-convex problem is formed.
- 2 Replace support. Replace old support planes with linearizations of the reduction constraint when a new pseudo-convex problem is formed.
- 3 Remove support and line search. Remove old support planes when a new pseudo-convex problem is formed and perform a line search when it is possible.
- 4 Replace support and line search. Replace old support planes with linearizations of the reduction constraint when a new pseudo-convex problem is formed and perform a line search when it is possible.

ECPretsol (*integer*) Return solution (1.MIP/2.NLP/3.QUALITY/4.PERFORMANCE)

The reported solution can be extracted from either the MIP or NLP solver result. If the MIP solution is returned only the primal values are available.

(default = 2)

- 1 Choose MIP solution if it is available.
- 2 Choose NLP solution if it is available.
- 3 Choose the solution with the best tolerance.
- 4 Choose the solution with the best objective value.

ECPstart (*integer*) User specified start-point

Define which variable levels are used when the optimization is started.

(default = 3)

- 0 Do not use a start-point; start the algorithm by solving the linear part (MIP) of the problem.
- 1 Use the user specified startpoint, but the variable levels are adjusted with a small value.
- 2 Use the exact start-point set by the user.
- 3 Use the exact start-point if linearly feasible; else adjust variable levels with a small value.

ECPstrategy (*integer*) AlphaECP strategy

(default = 2)

- 1 Convex strategy. Ensures global optimality for problems with convex objective function and convex constraints.
- 2 Pseudo-convex constraints. Ensures global optimality for problems with convex objective function and pseudo-convex constraints.
- 3 Pseudo-convex objective. Ensures global optimality for problems with pseudo-convex objective function and pseudo-convex constraints. The reformulation of a non-linear objective function into a constraint must be done in a specific way. The requirement is that the objective variable must be in a linear part of the non-linear function. The reformulation can be done, assuming that the minimized or maximized variable is called objvar, as follows: (objective function expression) - objvar =E= 0. Furthermore, this strategy can effectively use a feasible start-point.
- 4 Pseudo-convex objective, but first complete with ECPstrategy 2. (Only the necessary linearizations are removed when the *ECPstrategy* is changed.)
- 5 Pseudo-convex objective, but find the first solution with ECPstrategy 2. (Only the necessary linearizations are removed when the *ECPstrategy* is changed.)

MIPloglevel (*integer*) Level of MIP solver output

By default the detailed log of the MIP solver is suppressed in the AlphaECP log stream. If this option is turned on and the GAMS `LogOption` is set to 1 or 3, the MIP log will be merged into the AlphaECP log.

(default = 0)

0 No output.

1 MIP solver log goes to GAMS log.

MIPnrsols (*integer*) Upper limit of considered MIP solutions per MIP call

When the MIP solver returns several solutions then the most suitable solution is chosen. The solutions from the MIP solver are many times similar and a larger number might help to find a feasible MINLP solution if the constraints are almost satisfied. See *MIPsolstrat* to change the solution collection strategy.

(default = 50)

MIPoptcr (*real*) Relative MIP gap in intermediate sub-problems

The relative stopping tolerance sent to the MIP solver for intermediate MIP problems. Note that the *MIPoptcr* value is decreased automatically to zero during the optimization.

(default = 1.0)

MIPoptcrlim (*integer*) Initial MIPoptcr interval before MIPoptcr reduction

The *MIPoptcr* parameter is reduced in steps: From 1 to 0.5 to 0.3 to 0.2 to 0.1 to 0.0. The first reduction is at iteration *MIPoptcrlim*. *MIPoptcrlim* defines a step reduction at specific iterations (next reduction at iteration = the iteration number for this reduction multiplied by two). Note that a step reduction can also be caused by other reasons. If *MIPoptcrlim* is 200 then *MIPoptcr* is reduced at the following iterations: 200, 400, 800, etc.

(default = 200)

MIPoptfile (*integer*) Option file for MIP sub-solver

By default the MIP sub-solver is called without an option file. This option allows the user to specify an option number and therefore an option file to be used for the MIP sub-solver runs.

(default = 0)

MIPsolstrat (*integer*) MIP solution collection strategy

(default = 1)

0 Instruct MIP solver to return only one solution.

1 Instruct MIP solver to return any solutions encountered during MIP procedure.

2 Instruct MIP solver to search for solutions to obtain requested number MIPnrsols solutions.

3 As 2, but furthermore require the solutions to fulfill MIPoptcr ≥ 0.2 .

4 Let AlphaECP decide.

MIPsolver (*string*) MIP solver for sub-problems

This option allows the user to specify a GAMS MIP sub-solver, for example, CPLEX, GUROBI, XPRESS, etc. If no option is supplied the current active default MIP solver is selected.

(default = GAMS MIP solver)

NLPcall (*integer*) NLP strategy

Determine when the NLP solver is called.

(default = 5)

0 No output.

1 Call the NLP solver at end of AlphaECP algorithm.

2 As 1 and when a better solution is found.

3 As 2 and when the same integer solution is encountered *NLPlimsameint* times.

4 Let AlphaECP decide.

5 Let AlphaECP decide and add noise to the variable levels before call.

NLPcalliter (*integer*) NLP solver call at next (incremental) iteration

Specify an iteration interval for the NLP solver calls.

(default = 0)

NLPlimsameint (*integer*) NLP call after a number of recurring integer solutions

If the same integer solution is encountered *NLPlimsameint* times in a row then the NLP solver is called. The counter is reset after the NLP solver is called.

(default = 5)

NLPloglevel (*integer*) Level of NLP solver output

By default the detailed log of the NLP solver is suppressed in the AlphaECP log stream. If this option is turned on and the GAMS `LogOption` is set to 1 or 3, the NLP log will be merged into the AlphaECP log.

(default = 0)

0 No output.

1 NLP solver log goes to GAMS log.

NLPreslim (*real*) NLP time limit per call

The time limit (in seconds) given to the chosen NLP solver at each NLP solver call. Setting this option to 0 calculates a time limit which is relative to the problem size.

(default = 0)

NLPsolver (*string*) NLP solver for sub-problems

`solver[.n]` Solver is the name of the GAMS NLP solver that should be used in the root node, and `n` is the integer corresponding to `optfile`. If `.n` is missing, the `optfile` is treated as zero i.e. the NLP solver will not look for an options file. This option can be used to overwrite the default that uses the NLP solver specified with an `Option NLP = solver;` statement or the default GAMS solver for NLP.

(default = GAMS NLP solver)

reslim (*real*) Time limit for AlphaECP (in seconds)

(default = GAMS *reslim*)

solvelink (*integer*) Solvelink for NLP and MIP sub-solver

(default = 5)

1 Call NLP and MIP solver via script.

2 Call NLP and MIP solver via module.

5 Call NLP and MIP solver in memory.

TOLepsf (*real*) Pseudo-convex objective function termination tolerance

Maximum allowed absolute difference between the nonlinear and the MIP objective function value (used only in *ECPstrategy 3*).

(default = 1e-3)

TOLepsg (*real*) Constraint tolerance

The nonlinear constraint tolerance defines the maximum value that a nonlinear constraint may violate. For example, a constraint required to be zero may hold a value +/- *TOLepsg* at a solution.

(default = 1e-3)

TOLepsz (real) Distance tolerance for a new linearization

The maximum perpendicular distance between a valid cutting plane and its generation point (MIP solution).
(default = 1e-1)

TOLgrad (real) Gradient tolerance

The absolute value of a gradient's partial derivative must be above *TOLgrad* value in order for it to be considered nonzero.
(default = 1e-6)

TOLinfbnd (real) Infinity bound (MIP variable bound)

All variables must have a positive and a negative finite bound in order to ensure a bounded MIP problem. The finite bound value, *TOLinfbnd*, will be applied to single or double unbounded variables.
(default = 1e10)

6 FAQ

What are good settings to solve a convex problem?

Use *ECPstrategy 1* and *CUTdelcrit 1*.

What are good settings if the solution speed is essential?

Try *ECPstrategy 1* and *CUTdelcrit 1* and try if using multiple threads for the MIP solver improves the solution speed. However, the chance of not finding a feasible solution for a non-convex problem with nonlinear equality constraints is considerable.

What are good settings when the solution quality is essential?

Use *NLPcalliter 1* and *MIPsolstrat 4* or *3* and, furthermore, try different values for *CUTnrcuts* option, for example, *0.1*.

The objective function is non-linear, should the default *ECPstrategy* be used?

If the objective function constraint can be written in the required form of *ECPstrategy 3* then this strategy may find a better solution. If the constraints and the objective function are pseudo-convex the global optimal solution will be found.

7 AlphaECP References

Kelley J. E. (1960). The Cutting Plane Method for Solving Convex Programs. Journal of SIAM, Vol. VIII, No. 4, 703-712.

Pörn R. and Westerlund T. (2000). A Cutting Plane method for Minimizing Pseudo-convex Functions in the Mixed Integer Case. Computers Chem. Engng, 24, 2655-2665.

Still C. and Westerlund T. (2001). Extended Cutting Plane Algorithm. Encyclopedia of Optimization, Floudas and Pardalos (eds.), Kluwer Academic Publishers.

Westerlund T. and Pettersson F. (1995). An Extended Cutting Plane Method for Solving Convex MINLP Problems. Computers Chem. Engng Sup., 19, 131-136.

Westerlund T., Skrifvars H., Harjunkoski I. and Pörn R. (1998). An Extended Cutting Plane Method for Solving a Class of Non-Convex MINLP Problems. Computers Chem. Engng, 22, 357-365.

Westerlund T. and Pörn R. (2002). Solving Pseudo-Convex Mixed Integer Optimization Problems by Cutting Plane Techniques. Optimization and Engineering, 3, 253-280.