
**AIMMS Language Reference - AIMMS Outer Approximation Algorithm
for MINLP**

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com or order your hard-copy at www.lulu.com/aimms.

Copyright © 1993–2011 by Paragon Decision Technology B.V. All rights reserved.

Paragon Decision Technology B.V.	Paragon Decision Technology Inc.	Paragon Decision Technology Pte.
Schipholweg 1	500 108th Avenue NE	Ltd.
2034 LS Haarlem	Ste. # 1085	80 Raffles Place
The Netherlands	Bellevue, WA 98004	UOB Plaza 1, Level 36-01
Tel.: +31 23 5511512	USA	Singapore 048624
Fax: +31 23 5511517	Tel.: +1 425 458 4024	Tel.: +65 9640 4182
	Fax: +1 425 458 4025	

Email: info@aimms.com
WWW: www.aimms.com

AIMMS is a registered trademark of Paragon Decision Technology B.V. IBM ILOG CPLEX and sc CPLEX is a registered trademark of IBM Corporation. GUROBI is a registered trademark of Gurobi Optimization, Inc. KNITRO is a registered trademark of Ziena Optimization, Inc. XPRESS-MP is a registered trademark of FICO Fair Isaac Corporation. MOSEK is a registered trademark of Mosek ApS. WINDOWS and EXCEL are registered trademarks of Microsoft Corporation. $\text{T}_{\text{E}}\text{X}$, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, and $\text{A}_{\text{M}}\text{S}_{\text{L}}\text{A}_{\text{T}}\text{E}_{\text{X}}$ are trademarks of the American Mathematical Society. LUCIDA is a registered trademark of Bigelow & Holmes Inc. ACROBAT is a registered trademark of Adobe Systems Inc. Other brands and their products are trademarks of their respective holders.

Information in this document is subject to change without notice and does not represent a commitment on the part of Paragon Decision Technology B.V. The software described in this document is furnished under a license agreement and may only be used and copied in accordance with the terms of the agreement. The documentation may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Paragon Decision Technology B.V.

Paragon Decision Technology B.V. makes no representation or warranty with respect to the adequacy of this documentation or the programs which it describes for any particular purpose or with respect to its adequacy to produce any particular result. In no event shall Paragon Decision Technology B.V., its employees, its contractors or the authors of this documentation be liable for special, direct, indirect or consequential damages, losses, costs, charges, claims, demands, or claims for lost profits, fees or expenses of any nature or kind.

In addition to the foregoing, users should recognize that all complex software systems and their documentation contain errors and omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors shall not be responsible under any circumstances for providing information or corrections to errors and omissions discovered at any time in this book or the software it describes, whether or not they are aware of the errors or omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors do not recommend the use of the software described in this book for applications in which errors or omissions could threaten life, injury or significant loss.

This documentation was typeset by Paragon Decision Technology B.V. using $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ and the LUCIDA font family.

Chapter 22

AIMMS Outer Approximation Algorithm for MINLP

Outer approximation is a basic approach for solving Mixed-Integer NonLinear Programming (MINLP) models. The underlying algorithm is an interplay between two solvers, one for solving mixed-integer linear models and one for solving nonlinear models. Even though the standard outer approximation algorithm is provided with AIMMS, you as an algorithmic developer may want to customize the individual steps in order to obtain better performance and/or a better solution for your particular model.

*Open solver
approach*

The outer approximation algorithm in AIMMS is, therefore, provided as a customizable procedure written in the AIMMS language itself using functions and procedures provided by the GMP library (a white box solver), whereas most other outer approximation solvers are provided as a closed implementation (a black box solver). In the remainder of this chapter, we will refer to the open framework, with which an outer approximation algorithm can be implemented as the AIMMS Outer Approximation (AOA) solver.

*The AIMMS
Outer Approx-
imation solver*

In this chapter you find the description of, and the motivation behind, the open approach to solving MINLP models based on outer approximation. Following a brief introduction to the problem statement and the basic algorithm, an initial implementation of the basic solution algorithm using procedures in the AIMMS language is described. These procedures use functions that are especially designed to support the open approach. The chapter concludes with suggestions on additional ways to vary the individual steps of the overall algorithm in order to obtain customized versions of the outer approximation algorithm.

This chapter

22.1 Problem statement

The mixed-integer nonlinear programming models to be solved can be expressed as follows.

MINLP

Minimize:

$$f(x, y)$$

Subject to:

$$\begin{aligned} h(x, y) &= 0 \\ g(x, y) &\leq 0 \\ Cy + Dx &\leq d \\ x \in \mathcal{X} &= \{x \in \mathbb{R}^n \mid x^L \leq x \leq x^U\} \\ y \in \mathcal{Y} &= \mathbb{Z}^m \end{aligned}$$

The usual assumption is that the nonlinear submodel (i.e. the model in which all integer variables are fixed) is convex. This assumption is to guarantee that each locally optimal solution of the nonlinear submodel is also a globally optimal solution. In practice this assumption does not always hold, but the algorithm can still be applied. Convergence to a global optimum of the MINLP using the outer approximation algorithm is then no longer guaranteed.

*Usual
assumption*

22.2 Basic algorithm

The algorithm solves an alternating sequence of mixed-integer linear models and nonlinear models.

*Algorithm in
words*

1. First, the entire model is solved as a nonlinear program with all the integer variables relaxed as continuous variables between their bounds.
2. Then a linearization is carried out around the optimal solution, and the resulting constraints are added to the linear constraints that are already present. This new linear model is referred to as the master MIP model.
3. The master MIP model is solved as a mixed-integer linear program.
4. The integer part of the resulting optimal solution is then temporarily fixed, and the original MINLP model with fixed integer variables is solved as a nonlinear submodel.
5. Again, a linearization around the optimal solution is constructed and the new linear constraints are added to the master MIP model. To prevent cycling, one or more constraints are added to cut off the previously-found integer solution of the master model.
6. Steps 3-5 are repeated until one of the termination criteria is satisfied.

A more detailed description of the general Outer Approximation algorithm can be found in [Du86].

As linearizations are added to the master MIP model, the model becomes an improved approximation of the original MINLP model. Using the usual convexity assumption regarding the nonlinear submodel, convergence to a global optimum occurs when the objective function value of the master MIP model is worse than the value associated with the NLP submodel.

Convexity and convergence

Several termination criteria are used in practice. These criteria can be used in isolation or in some logical combination. Three of them are discussed in the following paragraphs.

Termination ...

Perhaps the most frequently-used criterion is the iteration limit. One reason is that a good solution is usually found during the first few iterations. Another reason for using an iteration limit is that the size of the underlying master MIP model tends to grow significantly each time linearization constraints are added, causing an increase in computation time.

... iteration limit

A second criterion is the worsening of the objective function value of two successive nonlinear submodels. This worsening occurs quite frequently, even if the NLP submodel is convex. The underlying reason is that the master MIP model will not always select binary solutions that lead to successively improving NLPs. This criterion seems appropriate when the worsening is persistent over several iterations.

... objective worsening

A third termination criterion is insufficient improvement in the objective function value of the master MIP model in relation to the objective function value of the previously solved NLP submodel. The difference between these two values represents the optimality gap, since the master MIP model represents an outer approximation (thus a relaxation) of the original MINLP model. When the gap is closed at crossover, the optimal solution has been found provided the NLP submodel is convex.

... crossover

Upon termination of the algorithm, the known best solution (also referred to as the incumbent solution) is declared as the final solution. In many practical applications, this solution is not necessarily optimal due to termination based on an iteration limit. In addition, it is often not possible to verify that the NLP submodel is convex.

Final solution

The term 'outer approximation' refers to the linear approximation of the convex nonlinear constraints at selected points along the boundary of the convex solution region. The accumulation of such inequality constraints forms an outer approximation of the solution region, and this approximation can be used in the optimization rather than the nonlinear constraints from which it was derived. The formula for the linearization of a scalar nonlinear inequality $g(x, y) \leq 0$ around the point $(x, y) = (x^0, y^0)$ is as follows.

Linearizations

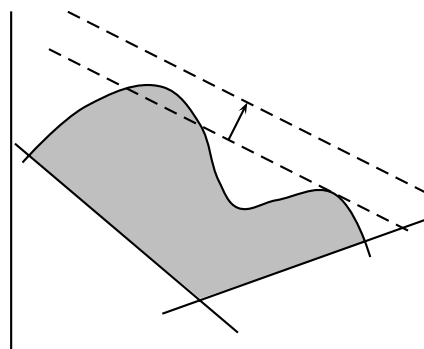


Figure 22.1: Effect of loosening a linearization

$$g(x^0, y^0) + \nabla g(x^0, y^0)^T \begin{bmatrix} x - x^0 \\ y - y^0 \end{bmatrix} \leq 0$$

The linear approximation ceases to be an outer approximation if the solution region is not convex. In this situation there is the possibility that portions of the solution region are cut off as illustrated in Figure 22.1.

The nonconvex case

In practical implementations of the outer approximation algorithm, the linearizations are allowed to move away from the feasible region. Such heuristic flexibility allows solutions to be found that would otherwise have been cut off. The implementation allows deviations through the use of artificial nonnegative variables and then penalizing them while solving the master model.

Loosening inequalities

22.3 Open solver approach

The basic outer approximation algorithm that is part of the AOA solver can be completely implemented using functionality provided by the GMP library.

The AOA solver explained

- From the math program instance representing the original MINLP model, a new math program instance representing the initial master MIP problem can be created using the function `GMP::Instance::CreateMasterMIP`.
- The functions from the `GMP::Linearization` namespace can be used to add linearizations of the nonlinear constraints of the original MINLP problem to the master MIP, in a customizable manner.
- Using the `GMP::Instance::FixColumns` procedure, the integer columns of the nonlinear sub-problem can be fixed to the current integer solution of the master MIP.
- Using the `GMP::Instance::AddIntegerEliminationRows` procedure, prior integer solutions of the master MIP are excluded from subsequent solves.

The basis GMP implementation of the AIMMS Outer Approximation (AOA) algorithm can be found in a single AIMMS module, called GMP Outer Approximation, that is provided as part of the AIMMS system. You can install this module using the **Install System Module** command in the AIMMS **Settings** menu.

AOA module ...

Because the AIMMS Outer Approximation algorithm is completely implemented using functionality provided the GMP library, you have the complete freedom to modify the math program instances generated by the basic AOA solver using the matrix manipulation routines discussed in Section 21.3. Such problem-specific modifications to the basic algorithm may help you to find a better overall solution to your MINLP problem, or to find a good solution faster.

Modifying the algorithm

22.4 A first and basic implementation

To call the AOA solver, the GMP library is used to generate a number of math program instances, and associated solver sessions, where SymbolicMP is the symbolic mathematical program containing the MINLP model.

Calling the AOA solver

```
! Create the MINLP model
GMINLP := GMP::Instance::Generate(SymbolicMP, FormatString("%e", SymbolicMP)) ;

! Create NLP model.
GNLP := GMP::Instance::Copy( GMINLP, 'OA_NLP' ) ;
GMP::Instance::SetMathematicalProgrammingType( GNLP, 'RMINLP' ) ;
ssNLP := GMP::Instance::CreateSolverSession( GNLP ) ;

! Create Master MIP model.
GMIP := GMP::Instance::CreateMasterMip( GMINLP, 'OA_MasterMIP' ) ;
ssMIP := GMP::Instance::CreateSolverSession( GMIP ) ;

BasicAlgorithm;
```

The basic algorithm outlined above is available in the GMP Outer Approximation module as the procedure DoOuterApproximation.

The basic algorithm is straightforward, and makes a call to five other procedures that execute the various algorithm steps. The naming convention is self-explanatory, and the following lines make up this first example of a main procedure. For the sake of brevity and clarity, the parts of the code used to create a status file and to customize the contents of the progress window have been left out. They can be found in the basic implementation of the AOA algorithm in the AOA module.

The basic algorithm

```
InitializeAlgorithm;
SolveRelaxedMINLP;

while ( not MINLPAlgorithmHasFinished ) do
  AddLinearizationsAndSolveMasterMIP;
  FixIntegerVariablesAndSolveNLP;
  TerminateOrPrepareForNextIteration;
endwhile;
```

Note that the scalar parameter `MINLPAlgorithmHasFinished` must be initially set to zero, and should only get a nonzero value when the algorithm is ready to terminate.

The following procedure is used to set all algorithmic parameters and options, and to prepare the status file and progress window output.

Initialize-Algorithm

```

IterationCount           := 0 ;
LinearizationCount       := 1 ;
EliminationCount         := 1 ;
IncumbentSolutionHasBeenFound := 0 ;
MINLPAlgorithmHasFinished := 0 ;

if ( NLPUseInitialValues ) then
  GMP::Solution::RetrieveFromModel( GMLP, SolNumbInitialValues ) ;
endif;

if ( GMP::Instance::GetDirection( GMINLP ) = 'maximize' ) then
  MINLPOptimizationDirection := 1;
else
  MINLPOptimizationDirection := -1;
endif;

GMP::Solution::SetProgramStatus( GMINLP, SolNumb, 'ProgramNotSolved' ) ;
GMP::Solution::SetSolverStatus( GMINLP, SolNumb, 'Unknown' ) ;

! The marginals of the NLP solver are needed.
option always_store_marginals := 'On';

```

The algorithmic parameters are initially set such that the AOA solver will always select the original initial values (i.e. the values of the variables prior to starting the AOA solver) as the starting values for each NLP submodel to be solved. This setting has found to work quite well in extensive tests performed using this algorithm.

The following termination procedure is used in several of the procedures that are described later.

MINLPTerminate

```

if ( IncumbentSolutionHasBeenFound ) then
  GMP::Solution::SetProgramStatus( GMINLP, SolNumb, 'LocallyOptimal' ) ;
  GMP::Solution::SetSolverStatus( GMINLP, SolNumb, 'NormalCompletion' ) ;
else
  GMP::Solution::SetProgramStatus( GMINLP, SolNumb, 'LocallyInfeasible' ) ;
  GMP::Solution::SetSolverStatus( GMINLP, SolNumb, 'NormalCompletion' ) ;
endif;

GMP::Solution::SendToModel( GMINLP, SolNumb ) ;
MINLPAlgorithmHasFinished := 1 ;

```

The parameter `IncumbentSolutionHasBeenFound` contains a value of one or zero depending on whether the AOA solver has received an incumbent solution to the original MINLP model. Such a solution may be found when solving the NLP submodel, and this must then be communicated to the AOA solver. Note that you also need to set the program status and indicate when the MINLP algorithm has finished.

The first model that is solved during the algorithm is the relaxed MINLP model. All integer variables are relaxed to continuous variables. The following procedure implements this first solution step of the outer approximation algorithm.

*SolveRelaxed-
MINLP*

```

SolveNLPSubProblem( 1 );
ProgramStatus := GMP::Solution::GetProgramStatus( GNL, So1Numb );

if ( ProgramStatus in NLPoptimalityStatus ) then
  ! Save NLP solution as MINLP solution if an integer solution has been found.

  if ( GMP::Solution::IsInteger( GNL, So1Numb ) ) then

    ! Set incumbent solution for MINLP.
    GMP::Solution::RetrieveFromModel( GMINLP, So1Numb );
    IncumbentSolutionHasBeenFound := 1 ;

    if ( TerminateAfterFirstNLPisInteger ) then
      ! Terminate if an integer solution has been found.

      MINLPterminate;
    endif;
  endif;
else
  ! Terminate if no linearization point has been found.

  SolverStatus := GMP::Solution::GetSolverStatus( GNL, So1Numb );

  if not ( SolverStatus in NLPContinuationStatus ) then
    MINLPterminate;
  endif;
endif ;

IterationCount += 1 ;
GMP::Solution::SetIterationCount( GMINLP, So1Numb, IterationCount );

```

When the procedure `SolveNLPSubProblem` has terminated, the AOA solver has typically found a point for the linearization step. The exception being when the NLP solver does not produce a solution at all (either feasible or infeasible). In such a situation the outer approximating algorithm should be terminated. Note that in the special event that the solution is feasible and has integer values for the integer variables, a locally optimal solution has been found and the AOA solver is instructed accordingly. Otherwise, the next step of the outer approximation algorithm can be executed.

If a termination flag has not been set, the following procedure adds linearizations to the master MIP model prior to solving it. If this model becomes infeasible, the outer approximation algorithm will be terminated.

*Add-
Linearizations-
AndSolve-
MasterMIP*

```

return when ( MINLPAlgorithmHasFinished );

GMP::Linearization::Add( GMIP, GNL, So1Numb, AllNonLinearConstraints,
  DeviationsPermitted, PenaltyMultiplier,
  LinearizationCount, JacobianTolerance );

LinearizationCount += 1 ;

```

```

GMP::SolverSession::Execute( ssMIP ) ;

GMP::Solution::RetrieveFromSolverSession( ssMIP, So1Numb ) ;
GMP::Solution::SendToModel( GMIP, So1Numb ) ;

ProgramStatus := GMP::Solution::GetProgramStatus( GMIP, So1Numb ) ;

if not ( ProgramStatus in MIPOptimalityStatus ) then
    MINLPTerminate;
endif ;

```

The AIMMS parameters `DeviationsPermitted` and `PenaltyMultiplier` are part of the AOA module. By default, deviations are allowed and are penalized with the value 1000 in the objective function of the master MIP.

The following procedure implements the next major step of the outer approximation algorithm. First, the NLP submodel is solved after fixing all the integer variables in the MINLP model using the values found from solving the previous master MIP model. Then, if the combination of integer values and feasible NLP solution values improves the current MINLP incumbent solution, a new incumbent solution is set. When the NLP submodel does not produce a solution (either feasible or infeasible), the outer approximation algorithm will be terminated.

*FixInteger-
VariablesAnd-
SolveNLP*

```

return when ( MINLPAlgorithmHasFinished );

SolveNLPSubProblem( 0 );
ProgramStatus := GMP::Solution::GetProgramStatus( GNLP, So1Numb ) ;

if ( ProgramStatus in NLPoptimalityStatus ) then
    ! Save NLP solution as MINLP solution if no incumbent solution
    ! has been found yet, or if the NLP solution is better than
    ! the current incumbent.

    if ( not IncumbentSolutionHasBeenFound ) then
        ! Set incumbent solution for MINLP.

        GMP::Solution::RetrieveFromModel( GMINLP, So1Numb ) ;
        IncumbentSolutionHasBeenFound := 1 ;
    else

        NLPobjectiveValue := GMP::Solution::GetObjectiv( GNLP , So1Numb ) ;
        MINLPIncumbentValue := GMP::Solution::GetObjectiv( GMINLP, So1Numb ) ;

        if ( MINLPSolutionImprovement( NLPobjectiveValue, MINLPIncumbentValue ) ) then
            ! Set incumbent solution for MINLP.

            GMP::Solution::RetrieveFromModel( GMINLP, So1Numb ) ;
            IncumbentSolutionHasBeenFound := 1 ;
        endif;
    endif ;
else
    ! Terminate if no linearization point has been found.

    SolverStatus := GMP::Solution::GetSolverStatus( GNLP, So1Numb ) ;

    if not ( SolverStatus in NLPContinuationStatus ) then

```

```

        MINLPTerminate;
    endif;
endif ;

```

The AOA solver maintains the MINLP model, the master MIP model, the NLP submodel, and the incumbent solution of the MINLP. As a result, direct access to the corresponding objective function values is available.

The procedure `SolveNLPSubProblem` solves the NLP sub-problem using various routines from the GMP library. The procedure has a single argument `initialSolve` which indicates whether this is the solve of the initial relaxed MINLP problem. In that case some steps in the procedure are not necessary.

*SolveNLPSub-
Problem*

```

    if ( NLPUseInitialValues ) then
        GMP::Solution::SendToModel( GMLP, So1NumbInitialValues ) ;
    elseif ( not initialSolve ) then
        GMP::Solution::SendToModel( GMIP, So1Numb ) ;
    endif;

    GMP::Solution::RetrieveFromModel( GMLP, So1Numb ) ;
    GMP::Solution::SendToSolverSession( ssNLP, So1Numb ) ;

    if ( not initialSolve ) then
        GMP::Instance::FixColumns( GMLP, GMIP, So1Numb, AllIntegerVariables ) ;
    endif;

    GMP::SolverSession::Execute( ssNLP ) ;

    GMP::Solution::RetrieveFromSolverSession( ssNLP, So1Numb ) ;
    GMP::Solution::SendToModel( GMLP, So1Numb ) ;

```

The following procedure implements the final major step of the outer approximation algorithm. If a termination flag has not been set previously, and the maximum number of iterations has not yet been reached, then the previously found integer solution of the master MIP model will be eliminated by adding the appropriate cuts. This will ensure that the next master MIP will have a new integer solution (or none at all).

*TerminateOr-
PrepareForNext-
Iteration*

```

    return when ( MINLPAAlgorithmHasFinished );

    if ( IterationCount = IterationMax ) then
        MINLPTerminate;
    else
        ! Prepare for next iteration

        IterationCount += 1 ;
        GMP::Solution::SetIterationCount( GMINLP, So1Numb, IterationCount ) ;
        GMP::Instance::AddIntegerEliminationRows( GMIP, So1Numb, EliminationCount ) ;
        EliminationCount += 1 ;
    endif ;

```

Note that you are responsible for determining the appropriate iteration count for the overall outer approximation algorithm. As you are free to develop a solution algorithm in any way you desire, it is not always possible for the AOA solver to determine the correct setting of the MINLP iteration count.


```

or (not MINLPSolutionImprovement( GMP::Solution::GetObjective(GMINLP, SolNumb),
                                  NLPCurrentObjectiveValue ))
or ( IterationCounter = IterationMax ) then

  MINLPTerminate;

else
  ! Prepare for next iteration

  IterationCount += 1 ;
  GMP::Solution::SetIterationCount( GMINLP, SolNumb, IterationCount ) ;
  GMP::Instance::AddIntegerEliminationRows( GMIP, SolNumb, EliminationCount ) ;
  EliminationCount += 1 ;
endif ;

```

The above paragraphs indicate just a few of the ways in which you can alter the basic implementation of the outer approximation algorithm in AIMMS. Of course, it is not necessary to develop your own variant. Whenever you need to solve a MINLP model using the AOA solver, you can simply call the basic implementation described in the previous section. As soon as you can see improved ways to solve a particular model, you can apply your own ideas by modifying the procedures as you see fit.

Conclusion

Bibliography

- [Du86] M.A. Duran and I.E. Grossmann, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, *Mathematical Programming* **36** (1986), 307-339.