

---

**AIMMS Language Reference - Additional Separation Procedures for  
Benders Decomposition**

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](http://www.aimms.com).

Copyright © 1993–2018 by AIMMS B.V. All rights reserved.

AIMMS B.V.  
Diakenhuisweg 29-35  
2033 AP Haarlem  
The Netherlands  
Tel.: +31 23 5511512

AIMMS Inc.  
11711 SE 8th Street  
Suite 303  
Bellevue, WA 98005  
USA  
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.  
55 Market Street #10-00  
Singapore 048941  
Tel.: +65 6521 2827

AIMMS  
SOHO Fuxing Plaza No.388  
Building D-71, Level 3  
Madang Road, Huangpu District  
Shanghai 200025  
China  
Tel.: ++86 21 5309 8733

Email: [info@aimms.com](mailto:info@aimms.com)  
WWW: [www.aimms.com](http://www.aimms.com)

AIMMS is a registered trademark of AIMMS B.V. IBM ILOG CPLEX and CPLEX is a registered trademark of IBM Corporation. GUROBI is a registered trademark of Gurobi Optimization, Inc. KNITRO is a registered trademark of Artelys. WINDOWS and EXCEL are registered trademarks of Microsoft Corporation.  $\TeX$ ,  $\LaTeX$ , and  $\AMS-\LaTeX$  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 AIMMS 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 AIMMS B.V.

**AIMMS 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 AIMMS 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, AIMMS 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, AIMMS 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 AIMMS B.V. using  $\LaTeX$  and the LUCIDA font family.

## Chapter 2

# Additional Separation Procedures for Benders' Decomposition

In Section 21.4 we showed the implementation of the procedure `Separation-OptimalityAndFeasibilityDual` as used by the textbook algorithm. The Benders' module implements also three other separation procedure that can be used by the Benders' decomposition algorithm depending on the setting of the control parameters `UseDual` and `FeasibilityOnly`. In this chapter we explain the implementation of these procedures.

*This chapter*

The procedure `SeparationFeasibilityOnly` is called by the Benders' decomposition algorithm in case the primal of the Benders subproblem is used (parameter `UseDual` equals 0) and if only feasibility cuts can be generated by the algorithm (parameter `FeasibilityOnly` equals 1). This procedure creates the feasibility problem for the (primal) subproblem if it does not exist yet. The feasibility problem is updated and solved. If its optimal objective value equals 0 (or is negative) then we have found an optimal solution for the original problem and the algorithm terminates. If the optimal objective value is larger than 0, indicating that the subproblem would have been infeasible, we add a feasibility cut to the master problem. The feasibility cut is created using the dual solution of the feasibility problem. By the dual solution we mean the shadow prices of the constraints and the reduced costs of the variables in the feasibility subproblem.

*The procedure  
Separation-  
FeasibilityOnly*

```
return when ( BendersAlgorithmFinished );

! Create feasibility problem corresponding to Subproblem (if it does not exist yet).
if ( not FeasibilityProblemCreated ) then
    gmpF := GMP::Instance::CreateFeasibility( gmpS, "FeasProb",
                                             useMinMax : UseMinMaxForFeasibilityProblem );
    solvesF := GMP::Instance::CreateSolverSession( gmpF );
    FeasibilityProblemCreated := 1;
endif;

! Update feasibility problem corresponding to Subproblem and solve it.
GMP::Benders::UpdateSubProblem( gmpF, gmpM, 1, round : 1 );

GMP::SolverSession::Execute( solvesF );
GMP::Solution::RetrieveFromSolverSession( solvesF, 1 );
```

```

! Check whether objective is 0 in which case optimality condition is satisfied.
ObjectiveFeasProblem := GMP::SolverSession::GetObjective( solsesF );

if ( ObjectiveFeasProblem <= BendersOptimalityTolerance ) then
  if ( MasterHasBeenSolved ) then
    return AlgorithmTerminate( 'Optimal' );
  endif;
endif;

! Add feasibility cut to the Master problem.
NumberOfFeasibilityCuts += 1;
GMP::Benders::AddFeasibilityCut( gmpM, gmpF, 1, NumberOfFeasibilityCuts );

```

The procedure `SeparationOptimalityAndFeasibility` is called by the Benders' decomposition algorithm in case the primal of the Benders subproblem is used (parameter `UseDual` equals 0) and if both optimality and feasibility cuts can be generated by the algorithm (parameter `FeasibilityOnly` equals 0). This procedure updates the primal subproblem and solves it. If the primal subproblem is infeasible then this procedure creates a feasibility problem for the subproblem if it does not exist yet. The feasibility problem is updated and solved, and its dual solution is used to create a feasibility cut which is added to the master problem. If the primal subproblem is bounded and optimal then the objective value of the subproblem is compared to the objective value of the master problem to check whether the algorithm has found an optimal solution for the original problem. If the solution is not optimal yet then an optimality cut is added to the master problem, using the dual solution of the primal subproblem.

*The procedure  
Separation-  
OptimalityAnd-  
Feasibility*

```

return when ( BendersAlgorithmFinished );

! Update Subproblem and solve it.
GMP::Benders::UpdateSubProblem( gmpS, gmpM, 1, round : 1 );

GMP::SolverSession::Execute( solsesS );
GMP::Solution::RetrieveFromSolverSession( solsesS, 1 );

ProgramStatus := GMP::Solution::GetProgramStatus( gmpS, 1 );

if ( ProgramStatus = 'Unbounded' ) then
  return AlgorithmTerminate( 'Unbounded' );
endif;

if ( ProgramStatus = 'Infeasible' ) then

  ! Create (if it does not exist yet) and update feasibility problem corresponding to
  ! Subproblem, and solve it to create feasibility cut for the Master problem.
  if ( not FeasibilityProblemCreated ) then
    gmpF := GMP::Instance::CreateFeasibility( gmpS, "FeasProb",
      useMinMax : UseMinMaxForFeasibilityProblem );
    solsesF := GMP::Instance::CreateSolverSession( gmpF );
    FeasibilityProblemCreated := 1;
  endif;

```

```

GMP::Benders::UpdateSubProblem( gmpF, gmpM, 1, round : 1 );

GMP::SolverSession::Execute( solvesF );
GMP::Solution::RetrieveFromSolverSession( solvesF, 1 );

! Add feasibility cut to the Master problem.
NumberOfFeasibilityCuts += 1;
GMP::Benders::AddFeasibilityCut( gmpM, gmpF, 1, NumberOfFeasibilityCuts );

else

! Check whether optimality condition is satisfied.
ObjectiveSubProblem := GMP::SolverSession::GetObjective( solvesS );

if ( SolutionImprovement( ObjectiveSubProblem, BestObjective ) ) then
    BestObjective := ObjectiveSubProblem;
endif;

if ( SolutionIsOptimal( ObjectiveSubProblem, ObjectiveMaster ) ) then
    return AlgorithmTerminate( 'Optimal' );
endif;

! Add optimality cut to the Master problem.
NumberOfOptimalityCuts += 1;
GMP::Benders::AddOptimalityCut( gmpM, gmpS, 1, NumberOfOptimalityCuts );

endif;

```

The procedure `SeparationFeasibilityOnlyDual` is called by the Benders' decomposition algorithm in case the dual of the Benders subproblem is used (parameter `UseDual` equals 1) and if only feasibility cuts can be generated by the algorithm (parameter `FeasibilityOnly` equals 1). This procedure updates the dual subproblem and solves it. If its optimal objective value equals 0 (or is negative) then we have found an optimal solution for the original problem and the algorithm terminates. If the optimal objective value is larger than 0 then we create a feasibility cut using the level values of the variables in the solution of the dual subproblem. This feasibility cut is added to the master problem.

*The procedure  
Separation-  
Feasibility-  
OnlyDual*

```

return when ( BendersAlgorithmFinished );

! Update Subproblem and solve it.
GMP::Benders::UpdateSubProblem( gmpS, gmpM, 1, round : 1 );

GMP::SolverSession::Execute( solvesS );
GMP::Solution::RetrieveFromSolverSession( solvesS, 1 );

! Check whether objective is 0 in which case optimality condition is satisfied.
ObjectiveSubproblem := GMP::SolverSession::GetObjective( solvesS );

if ( ObjectiveSubproblem <= BendersOptimalityTolerance ) then
    if ( MasterHasBeenSolved ) then
        return AlgorithmTerminate( 'Optimal' );
    endif;
else
    ! Add feasibility cut to the Master problem.

```

```
NumberOfFeasibilityCuts += 1;  
GMP::Benders::AddFeasibilityCut( gmpM, gmpS, 1, NumberOfFeasibilityCuts );  
endif;
```