DICOPT

Ignacio E. Grossmann, Jagadisan Viswanathan, Aldo Vecchietti; Engineering Research Design Center, Carnegie Mellon University, Pittsburgh, PA

Ramesh Raman, Erwin Kalvelagen; GAMS Development Corporation, Washington D.C.

Contents

1 Introduction **1**
2 Requirements **2**
3 How to Run a Model with GAMS/DICOPT **2**
4 Overview of DICOPT **2**
5 The Algorithm **3**
6 Modeling **5**
6.1 Relaxed Model 5
6.2 OPTCR and OPTCA 6
6.3 Integer Formulations 6
6.4 Non-smooth Functions 6
7 GAMS Options **7**
7.1 The OPTION Statement 7
7.2 The Model Suffix 8
8 DICOPT Options **9**
9 DICOPT Output **16**
10 Special Notes **17**
10.1 Stopping Rule 17
10.2 Solving the NLP Problems 19
10.3 Solving the MIP Master Problems 19
10.4 Model Debugging 19

1 Introduction

DICOPT is a program for solving mixed-integer nonlinear programming (MINLP) problems that involve linear binary or integer variables and linear and nonlinear continuous variables. While the modeling and solution of these MINLP optimization problems has not yet reached the stage of maturity and reliability as linear, integer or non-linear programming modeling, these problems have a rich area of applications. For example, they often arise in engineering design, management sciences, and finance. DICOPT (DIscrete and Continuous OPTimizer) was developed by J. Viswanathan and Ignacio E. Grossmann at the Engineering Design Research Center (EDRC) at Carnegie Mellon University. The program is based on the extensions of the outer-approximation algorithm for the equality relaxation strategy. The MINLP algorithm inside DICOPT solves a series of NLP and MIP sub-problems. These sub-problems can be solved using any NLP (Nonlinear Programming) or MIP (Mixed-Integer Programming) solver that runs under GAMS.

Although the algorithm has provisions to handle non-convexities, it does not necessarily obtain the global optimum.

The GAMS/DICOPT system has been designed with two main goals in mind:

- to build on existing modeling concepts and to introduce a minimum of extensions to the existing modeling language and provide upward compatibility to ensure easy transition from existing modeling applications to nonlinear mixed-integer formulations
- to use existing optimizers to solve the DICOPT sub-problems. This allows one to match the best algorithms to the problem at hand and guarantees that any new development and enhancements in the NLP and MIP solvers become automatically and immediate available to DICOPT.

2 Requirements

In order to use DICOPT you will need to have access to a licensed GAMS BASE system as well as at least one licensed MIP solver and one licensed NLP solver. For difficult models it is advised to have access to multiple solvers. Free student/demo systems are available from GAMS Development Corporation. These systems are restricted in the size of models that can be solved.

3 How to Run a Model with GAMS/DICOPT

DICOPT is capable of solving only MINLP models. If you did not specify DICOPT as the default solver, then you can use the following statement in your GAMS model:

```
option minlp = dicopt;
```

It should appear before the solve statement. DICOPT automatically uses the default MIP and NLP solver to solve its sub-problems. One can override this with the GAMS statements like:

```
option nlp = conopt;  { or any other nlp solver }
option mip = cplex;   { or any other mip solver }
```

These options can also be specified on the command line, like:

```
> gams mymodel minlp=dicopt nlp=conopt mip=cplex
```

In the IDE (Integrated Development Enviroment) the command line option can be specified in the edit line in the right upper corner of the main window.

Possible NLP solvers include `minos5`, `minos`, `conopt`, `conopt3`, and `snopt`. Possible MIP solvers are `cplex`, `osl`, `osl2`, `osl3`, `xpress`, and `xa`.

With an option file it is even possible to use alternate solvers in different cycles. Section 8 explains this is in detail.

4 Overview of DICOPT

DICOPT solves models of the form:

MINLP	min or max	$f(x,y)$
	subject to	$g(x,y) \sim b$
		$\ell_x \leq x \leq u_x$
		$y \in \lceil \ell_y \rceil, ..., \lfloor u_y \rfloor$

where x are the continuous variables and y are the discrete variables. The symbol $\sim$ is used to denote a vector of relational operators $\{\leq, =, \geq\}$. The constraints can be either linear or non-linear. Bounds ℓ and u on the variables are handled directly. $\lceil x \rceil$ indicates the smallest integer, greater than or equal to x. Similarly, $\lfloor x \rfloor$ indicates the largest integer, less than or equal to x. The discrete variables can be either integer variables or binary variables.

5 The Algorithm

The algorithm in DICOPT is based on three key ideas:

- Outer Approximation
- Equality Relaxation
- Augmented Penalty

Outer Approximation refers to the fact that the surface described by a convex function lies above the tangent hyper-plane at any interior point of the surface. (In 1-dimension, the analogous geometrical result is that the tangent to a convex function at an interior point lies below the curve). In the algorithm outer-approximations are attained by generating linearizations at each iterations and accumulating them in order to provide successively improved linear approximations of nonlinear convex functions that underestimate the objective function and overestimate the feasible region.

Equality Relaxation is based on the following result from non-linear programming. Suppose the MINLP problem is formulated in the form:

$$\begin{aligned} \text{minimize or maximize } & f(x) + c^T y \\ \text{subject to } & G(x) + Hy \sim b \\ & \ell \leq x \leq u \\ & y \in \{0, 1\} \end{aligned} \tag{1.1}$$

i.e. the discrete variables are binary variables and they appear linearly in the model.

If we reorder the equations into equality and inequality equations, and convert the problem into a minimization problem, we can write:

$$\begin{aligned} \text{minimize } & c^T y + f(x) \\ \text{subject to } & Ay + h(x) = 0 \\ & By + g(x) \leq 0 \\ & \ell \leq x \leq u \\ & y \in \{0, 1\} \end{aligned} \tag{1.2}$$

Let $y^{(0)}$ be any fixed binary vector and let $x^{(0)}$ be the solution of the corresponding NLP subproblem:

$$\begin{aligned} \text{minimize } & c^T y^{(0)} + f(x) \\ \text{subject to } & Ay^{(0)} + h(x) = 0 \\ & By^{(0)} + g(x) \leq 0 \\ & \ell \leq x \leq u \end{aligned} \tag{1.3}$$

Further let

$$\begin{aligned} T^{(0)} &= \text{diag}(t_{i,i}) \\ t_{i,i} &= \text{sign}(\lambda_i) \end{aligned} \tag{1.4}$$

where λ_i is the Lagrange multiplier of the i-th equality constraint.

If f is pseudo-convex, h is quasi-convex, and g is quasi-convex, then x^0 is also the solution of the following NLP:

$$\begin{aligned} \text{minimize } & c^T y^{(0)} + f(x) \\ \text{subject to } & T^{(0)}(Ay^{(0)} + h(x)) \leq 0 \\ & By^{(0)} + g(x) \leq 0 \\ & \ell \leq x \leq u \end{aligned} \tag{1.5}$$

In colloquial terms, under certain assumptions concerning the convexity of the nonlinear functions, an equality constraint can be "relaxed" to be an inequality constraint. This property is used in the MIP master problem to accumulate linear approximations.

Augmented Penalty refers to the introduction of (non-negative) slack variables on the right hand sides of the just described inequality constraints and the modification of the objective function when assumptions concerning convexity do not hold.

The algorithm underlying DICOPT starts by solving the NLP in which the 0-1 conditions on the binary variables are relaxed. If the solution to this problem yields an integer solution the search stops. Otherwise, it continues with an alternating sequence of nonlinear programs (NLP) called subproblems and mixed-integer linear programs (MIP) called master problems. The NLP subproblems are solved for fixed 0-1 variables that are predicted by the MIP master problem at each (major) iteration. For the case of convex problems, the master problem also provides a lower bound on the objective function. This lower bound (in the case of minimization) increases monotonically as iterations proceed due to the accumulation of linear approximations. Note that in the case of maximization this bound is an upper bound. This bound can be used as a stopping criterion through a DICOPT option `stop 1` (see section 8). Another stopping criterion that tends to work very well in practice for non-convex problems (and even on convex problems) is based on the heuristic: stop as soon as the NLP subproblems start worsening (i.e. the current NLP subproblem has an optimal objective function that is worse than the previous NLP subproblem). This stopping criterion relies on the use of the augmented penatly and is used in the description of the algorithm below. This is also the default stopping criterion in the implementation of DICOPT. The algorithm can be stated briefly as follows:

1. Solve the NLP relaxation of the MINLP program. If $y^{(0)} = y$ is integer, stop("integer optimum found"). Else continue with step 2.
2. Find an integer point $y^{(1)}$ with an MIP master problem that features an augmented penalty function to find the minimum over the convex hull determined by the half-spaces at the solution $(x^{(0)}, y^{(0)})$.
3. Fix the binary variables $y = y^{(1)}$ and solve the resulting NLP. Let $(x^{(1)}, y^{(1)})$ be the corresponding solution.
4. Find an integer solution $y^{(2)}$ with a MIP master problem that corresponds to the minimization over the intersection of the convex hulls described by the half-spaces of the KKT points at $y^{(0)}$ and $y^{(1)}$.
5. Repeat steps 3 and 4 until there is an increase in the value of the NLP objective function. (Repeating step 4 means augmenting the set over which the minimization is performed with additional linearizations - i.e. half-spaces - at the new KKT point).

In the MIP problems integer cuts are added to the model to exclude previously determined integer vectors $y^{(1)}, y^{(2)}, ..., y^{(K)}$.

For a detailed description of the theory and references to earlier work, see [5, 3, 1].

The algorithm has been extended to handle general integer variables and integer variables appearing nonlinearly in the model.

6 Modeling

6.1 Relaxed Model

Before solving a model with DICOPT, it is strongly advised to experiment with the relaxed model where the integer restrictions are ignored. This is the `RMINLP` model. As the DICOPT will start solving the relaxed problem and can use an existing relaxed optimal solution, it is a good idea to solve the `RMINLP` always before attempting to solve the `MINLP` model. I.e. the following fragment is not detrimental with respect to performance:

```
  model m /all/;
  option nlp=conopt;
  option mip=cplex;
  option rminlp=conopt;
  option minlp=dicopt;
*
* solve relaxed model
*
  solve m using rminlp minimizing z;
  abort$(m.modelstat > 2.5) "Relaxed model could not be solved";

*
* solve minlp model
*
  solve m using minlp minimizing z;
```

The second `SOLVE` statement will only be executed if the first `SOLVE` was succesful, i.e. if the model status was one (optimal) or two (locally optimal).

In general it is not a good idea to try to solve an MINLP model if the relaxed model can not be solved reliably. As the RMINLP model is a normal NLP model, some obvious points of attention are:

- Scaling. If a model is poorly scaled, an NLP solver may not be able find the optimal or even a feasible solution. Some NLP solvers have automatic scaling algorithms, but often it is better to attack this problem on the modeling level. The GAMS scaling facility can help in this respect.
- Starting point. If a poor starting point is used, the NLP solver may not be able to find a feasible or optimal solution. A starting point can be set by setting level values, e.g. `X.L = 1;`. The GAMS default levels are zero, with is often not a good choice.
- Adding bounds. Add bounds so that all functions can be properly evaluated. If you have a function $\sqrt{x}$ or $\log(x)$ in the model, you may want to add a bound `X.LO=0.001;`. If a function like $\log(f(x))$ is used, you may want to introduce an auxiliary variable and equation $y = f(x)$ with an appropriate bound `Y.LO=0.001;`.

In some cases the relaxed problem is the most difficult model. If you have more than one NLP solver available, you may want to try a sequence of them:

```
  model m /all/;
  option nlp=conopt;
  option mip=cplex;
  option rminlp=conopt;
  option minlp=dicopt;
*
* solve relaxed model
*
  solve m using rminlp minimizing z;
  if (m.modelstat > 2.5,
```

```
     option rminlp=minos;
     solve m using rminlp minimizing z;
  );
  if (m.modelstat > 2.5,
     option rminlp=snopt;
     solve m using rminlp minimizing z;
  );

*
* solve minlp model
*
  solve m using minlp minimizing z;
```

In this fragment, we first try to solve the relaxed model using `CONOPT`. If that fails we try `MINOS`, and if that solve also fails, we try `SNOPT`.

It is worthwhile to spend some time in getting the relaxed model to solve reliably and speedily. In most cases, modeling improvements in the relaxed model, such as scaling, will also benefit the subsequent NLP sub-problems. In general these modeling improvements turn out to be rather solver independent: changes that improve the performance with CONOPT will also help solving the model with MINOS.

6.2 OPTCR and OPTCA

The DICOPT algorithm assumes that the integer sub-problems are solved to optimality. The GAMS options for `OPTCR` and `OPTCA` are therefore ignored: subproblems are solved with both tolerances set to zero. If you really want to solve a MIP sub-problem with an optimality tolerance, you can use the DICOPT option file to set `OPTCR` or `OPTCA` in there. For more information see section 8.

For models with many discrete variables, it may be necessary to introduce an `OPTCR` or `OPTCA` option in order to solve the model in acceptable time. For models with a limited number of integer variables the default to solve MIP sub-models to optimality may be acceptable.

6.3 Integer Formulations

A number of MIP formulations are not very obvious and pose a demand on the modeler with respect to knowledge and experience. A good overview of integer programming modeling is given in [6].

Many integer formulations use a so-called big-M construct. It is important to choose small values for those big-M numbers. As an example consider the fixed charge problem where $y_i \in \{0,1\}$ indicate if facility i is open or closed, and where x_i is the production at facility i. Then the cost function can be modeled as:

$$\begin{aligned} C_i &= f_i y_i + v_i x_i \\ x_i &\le M_i y_i \\ y_i &\in \{0,1\} \\ 0 &\le x_i \le cap_i \end{aligned} \tag{1.6}$$

where f_i is the fixed cost and v_i the variables cost of operating facility i. In this case M_i should be chosen large enough that x_i is not restricted if $y_i = 1$. On the other hand, we want it as small as possible. This leads to the choice to have M_i equal to the (tight) upperbound of variable x_i (i.e. the capacity cap_i of facility i).

6.4 Non-smooth Functions

NLP modelers are alerted by GAMS against the use of non-smooth functions such as `min()`, `max()`, `smin()`, `smax()` and `abs()`. In order to use these functions, a non-linear program needs to be declared as a DNLP model instead of a regular NLP model:

```
option dnlp=conopt;
model m /all/;
solve m minimizing z using dnlp;
```

This construct is to warn the user that problems may arise due to the use of non-smooth functions.

A possible solution is to use a smooth approximation. For instance, the function $f(x) = |x|$ can be approximated by $g(x) = \sqrt{x^2 + \varepsilon}$ for some $\varepsilon > 0$. This approximation does not contain the point $(0, 0)$. An alternative approximation can be devised that has this property:

$$f(x) \approx \frac{2x}{1 + e^{-x/h}} - x \tag{1.7}$$

For more information see [2].

For MINLP models, there is not such a protection against non-smooth functions. However, the use of such functions is just as problematic here. However, with MINLP models we have the possibility to use discrete variables, in order to model if-then-else situations. For the case of the absolute value for instance we can replace x by $x^+ - x^-$ and $|x|$ by $x^+ + x^-$ by using:

$$\begin{aligned} x &= x^+ - x^- \\ |x| &= x^+ + x^- \\ x^+ &\leq \delta M \\ x^- &\leq (1 - \delta) M \\ x^+, x^- &\geq 0 \\ \delta &\in \{0, 1\} \end{aligned} \tag{1.8}$$

where δ is a binary variable.

7 GAMS Options

GAMS options are specified in the GAMS model source, either using the `option` statement or using a model suffix.

7.1 The OPTION Statement

An option statement sets a global parameter. An option statement should appear *before* the `solve` statement, as in:

```
model m /all/;
option iterlim=100;
solve m using minlp minimizing z;
```

Here follows a list of option statements that affect the behavior of DICOPT:

option domlim = n;

This option sets a limit on the total accumulated number of non-linear function evaluation errors that are allowed while solving the NLP subproblems or inside DICOPT itself. An example of a function evaluation error or domain error is taking the square root of a negative number. This situations can be prevented by adding proper bounds. The default is zero, i.e. no function evaluation errors are allowed.

In case a domain error occurs, the listing file will contain an appropriate message, including the equation that is causing the problem, for instance:

```
**** ERRORS(S) IN EQUATION loss(cc,sw)
     2 instance(s) of - UNDEFINED REAL POWER (RETURNED  0.0E+00)
```

If such errors appear you can increase the `DOMLIM` limit, but often it is better to prevent the errors to occur. In many cases this can be accomplished by adding appropriate bounds. Sometimes you will need to add extra variables and equations to accomplish this. For instance with an expression like $\log(x - y)$, you may want to introduce a variable $z > \varepsilon$ and an equation $z = x - y$, so that the expression can be rewritten as $\log(z)$.

option iterlim = n;
This option sets a limit on the total accumulated (minor) iterations performed in the MIP and NLP sub-problems. The default is 1000.

option minlp = dicopt;
Selects DICOPT to solve MINLP problems.

option mip = s;
This option sets the MIP solver to be used for the MIP master problems. Note that changing from one MIP solver to another can lead to different results, and may cause DICOPT to follow a different path.

option nlp = s;
This option sets the NLP solver to be used for the NLP sub-problems. Note that changing from one NLP solver to another can lead to different results, and may cause DICOPT to follow a different path.

option optca = x;
This option is ignored. MIP master problems are solved to optimality unless specified differently in the DICOPT option file.

option optcr = x;
This option is ignored. MIP master problems are solved to optimality unless specified differently in the DICOPT option file.

option reslim = x;
This option sets a limit on the total accumulated time (in seconds) spent inside DICOPT and the subsolvers. The default is 1000 seconds.

option sysout = on;
This option will print extra information to the listing file.

In the list above (and in the following) n indicates an integer number. GAMS will also accept fractional values: they will be rounded. Options marked with an x parameter expect a real number. Options with an s parameter, expect a string argument.

7.2 The Model Suffix

Some options are set by assigning a value to a model suffix, as in:

```
model m /all/;
m.optfile=1;
solve m using minlp minimizing z;
```

Here follows a list of model suffices that affect the behavoir of DICOPT:

***m*.dictfile = 1;**
This option tells GAMS to write a dictionary file containing information about GAMS identifiers (equation and variables names). This information is needed when the DICOPT option `nlptracelevel` is used. Otherwise this option can be ignored.

***m*.iterlim = *n*;**
Sets the total accumulated (minor) iteration limit. This option overrides the global iteration limit set by an option statement. E.g.,

```
model m /all/;
m.iterlim = 100;
option iterlim = 1000;
solve m using minlp minimizing z;
```

will cause DICOPT to use an iteration limit of 100.

***m*.optfile = 1;**
This option instructs DICOPT to read an option file `dicopt.opt`. This file should be located in the current directory (or the project directory when using the GAMS IDE). The contents of the option file will be echoed to the listing file and to the screen (the log file):

```
--- DICOPT: Reading option file D:\MODELS\SUPPORT\DICOPT.OPT
> maxcycles 10
--- DICOPT: Starting major iteration 1
```

If the option file does not exist, the algorithm will proceed using its default settings. An appropriate message will be displayed in the listing file and in the log file:

```
--- DICOPT: Reading option file D:\MODELS\SUPPORT\DICOPT.OPT
--- DICOPT: File does not exist, using defaults...
--- DICOPT: Starting major iteration 1
```

***m*.optfile = *n*;**
If $n > 1$ then the option file that is read is called `dicopt.op`n (for $n = 2, ..., 9$) or `dicopt.o`n (for $n = 10, ..., 99$). E.g. `m.optfile=2;` will cause DICOPT to read `dicop.op2`.

***m*.prioropt = 1;**
This option will turn on the use of priorities on the discrete variables. Priorities influence the branching order chosen by the MIP solver during solution of the MIP master problems. The use of priorities can greatly impact the performance of the MIP solver. The priorities themselves have to be specified using the `.prior` variables suffix, e.g. `x.prior(i,j) = ord(i);`. Contrary to intuition, variables with a lower value for their priority are branched on before variables with a higher priority. I.e. the most important variables should get lower priority values.

***m*.reslim = *x*;**
Sets the total accumulated time limit. This option overrides the global time limit set by an option statement.

8 DICOPT Options

This sections describes the options that can be specified in the DICOPT option file. This file is usually called `dicopt.opt`. In order to tell DICOPT to read this file, you will need to set the `optfile` model suffix, as in:

```
model m /all/;
m.optfile=1;
solve m using minlp minimizing z;
```

The option file is searched for in the current directory, or in case the IDE (Integrated Development Environment) is used, in the project directory.

The option file is a standard text file, with a single option on each line. All options are case-insensitive. A line is a comment line if it starts with an asterisk, `*`, in column one. A valid option file can look like:

```
* stop only on infeasible MIP or hitting a limit
stop 0
* use minos to solve first NLP sub problem
* and conopt for all subsequent ones
nlpsolver minos conopt
```

A convenient way to write the option file from within a GAMS model is to use the following construct:

```
$onecho > dicopt.opt
stop 0
nlpsolver minos conopt
$offecho
```

This will make the model self-contained. Notice however that this overwrites an existing file `dicopt.opt`.

Here follows a list of available DICOPT options:

continue n

This option can be used to let DICOPT continue in case of NLP solver failures. The preferred approach is to fix the model, such that NLP subproblems solve without problems. However, in some cases we can ignore (partial) failures of an NLP solver in solving the NLP subproblems as DICOPT may recover later on. During model debugging, you may therefore add the option `continue 0`, in order for DICOPT to function in a more finicky way.

continue 0

Stop on solver failure. DICOPT will terminate when an NLP subproblem can not be solved to optimality. Some NLP solvers terminate with a status other than optimal if not all of the termination criteria are met. For instance, the change in the objective function is negligable (indicating convergence) but the reduced gradients are not within the required tolerance. Such a solution may or may not be close the (local) optimum. Using `continue 0` will cause DICOPT not to accept such a solution.

continue 1

NLP subproblem failures resulting in a non-optimal but feasible solutions are accepted. Sometimes an NLP solver can not make further progress towards meeting all optimality conditions, although the current solution is feasible. Such a solution can be accepted by this option.

continue 2

NLP subproblem failures resulting in a non-optimal but feasible solution are accepted (as in option `continue 1`). NLP subproblem failures resulting in an infeasible solution are ignored. The corresponding configuration of discrete variables is forbidden to be used again. An integer cut to accomplish this, is added to subsequent MIP master problems. Note that the relaxed NLP solution should be feasible. This setting is the default.

domlim $i_1\ i_2\ \ldots\ i_n$

Sets a limit of the number of function and derivative evaluation errors for a particular cycle. A number of -1 means that the global GAMS option `domlim` is used. The last number i_n sets a domain error limit for all cycles $n, n+1, \ldots$.

Example: `domlim 0 100 0`

The NLP solver in the second cycle is allowed to make up to 100 evaluation errors, while all other cycles must be solved without evaluation errors.

The default is to use the global GAMS `domlim` option.

epsmip x

This option can be used to relax the test on MIP objective functions. The objective function values of the MIP master problems should form a monotonic worsening curve. This is not the case if the MIP master problems are not solved to optimality. Thus, if the options `OPTCR` or `OPTCA` are set to a nonzero value, this test is bypassed. If the test fails, DICOPT will fail with a message:

```
The MIP solution became better after adding integer cuts.
Something is wrong. Please check if your model is properly
scaled. Also check your big M formulations -- the value
of M should be relatively small.

This error can also occur if you used a MIP solver option
file with a nonzero OPTCR or OPTCA setting. In that case
you may want to increase the EPSMIP setting using a
DICOPT option file.
```

The value of

$$\frac{\text{PreviousObj} - \text{CurrentObj}}{1 + |\text{PreviousObj}|} \tag{1.9}$$

is compared against `epsmip`. In case the test fails, but you want DICOPT to continue anyway, you may want to increase the value of `epsmip`. The current values used in the test (previous and current MIP objective, `epsmip`) are printed along with the message above, so you will have information about how much you should increase `epsmip` to pass the test. Normally, you should not have to change this value. The default is $x = 1.0e - 6$.

epsx x
This tolerance is used to distinguish integer variables that are set to an integer value by the user, or integer variables that are fractional. See the option `relaxed`. Default: $x = 1.0e - 3$.

infeasder n
This option is to determine whether linearizations of infeasible NLP subproblems are added or not to the MIP master problem.

infeasder 0
This is the default option in which no linearizations are added in the infeasible NLP subproblems. In this case a simple integer cut is added to remove from consideration the 0-1 vector that gave rise to the infeasible NLP. Since this may slow the convergence, it is recommended to reformulate the MINLP with "elastic" constraints (i.e. adding slacks to infeasible constraints and adding a penalty for them in the objective) so as to ensure that the NLP subproblems are mathematically feasible.

infeasder 1
This will add linearizations derived from the infeasible NLP subproblem to the master problem. This option is recommended to speed up convergence when the MINLP is known to be convex (i.e. its continuous relaxation is convex). If used for a nonconvex MINLP possibility of cutting-off the global optimum is increased.

The default is $n = 0$.

maxcycles n
The maximum number of cycles or major iterations performed by DICOPT. The default is $n = 20$.

mipiterlim $i_1\ i_2\ \ldots\ i_n$
Sets an iteration limit on individual MIP master problems. The last number i_n is valid for all subsequent cycles $n, n+1, \ldots$. A number of -1 indicates that there is no (individual) limit on the corresponding MIP master problem. A global iteration limit is maintained through the GAMS option `iterlim`.

Example: `mipiterlim 10000 -1`
The first MIP master problem can not use more than 10000 iterations, while subsequent MIP master problems are not individually restricted.

Example: `mipiterlim 10000`
Sets an iteration limit of 10000 on all MIP master problems.

When this option is used it is advised to have the option `continue` set to its default of 2. The default for this option is not to restrict iteration counts on individual solves of MIP master problems.

mipoptfile s_1 s_2 ... s_n

Specifies the option file to be used for the MIP master problems. Several option files can be specified, separated by a blank. If a digit `1` is entered, the default option file for the MIP solver in question is being used. The digit `0` indicates: no option file is to be used. The last option file is also used for subsequent MIP master problems.

Example: `mipoptfile mip.opt mip2.opt 0`

This option will cause the first MIP master problem solver to read the option file `mip.opt`, the second one to read the option file `mip2.opt` and subsequent MIP master problem solvers will not use any option file.

Example: `mipoptfile 1`

This will cause the MIP solver for all MIP subproblems to read a default option file (e.g. `cplex.opt`, `xpress.opt`, `osl2.opt` etc.).

Option files are located in the current directory (or the project directory when using the IDE). The default is not to use an option file.

mipreslim x_1 x_2 ... x_n

Sets a resource (time) limit on individual MIP master problems. The last number x_n is valid for all subsequent cycles $n, n+1, \ldots$. A number -1.0 means that the corresponding MIP master problem is not individually time restricted. A global time limit is maintained through the GAMS option `reslim`.

Example: `mipreslim -1 10000 -1`

The MIP master problem in cycle 2 can not use more than 100 seconds, while subsequent MIP master problems are not individually restricted.

Example: `mipreslim 1000`

Sets a time limit on all MIP master problems of 1000 seconds.

When this option is used it is advised to have the option `continue` set to its default of 2. The default for this option is not to restrict individually the time a solver can spent on the MIP master problem.

mipsolver s_1 s_2 ... s_n

This option specifies with MIP solver to use for the MIP master problems.

Example: `mipsolver cplex osl2`

This instructs DICOPT to use Cplex for the first MIP and OSL2 for the second and subsequent MIP problems. The last entry may be used for more than one problem.

The names to be used for the solvers are the same as one uses in the GAMS statement `OPTION MIP=....;`. The default is to use the default MIP solver.
Note that changing from one MIP solver to another can lead to different results, and may cause DICOPT to follow a different path.

nlpiterlim i_1 i_2 ... i_n

Sets an iteration limit on individual NLP subproblems. The last number i_n is valid for all subsequent cycles $n, n+1, \ldots$. A number of -1 indicates that there is no (individual) limit on the corresponding NLP subproblem. A global iteration limit is maintained through the GAMS option `iterlim`.

Example: `nlpiterlim 1000 -1`

The first (relaxed) NLP subproblem can not use more than 1000 iterations, while subsequent NLP subproblems are not individually restricted.

Example: `nlpiterlim 1000`

Sets an iteration limit of 1000 on all NLP subproblems.

When this option is used it is advised to have the option `continue` set to its default of 2. The default is not to restrict the amount of iterations an NLP solver can spend on an NLP subproblem, other than the global iteration limit.

nlpoptfile s_1 s_2 ... s_n

Specifies the option file to be used for the NLP subproblems. Several option files can be specified, separated by a blank. If a digit `1` is entered, the default option file for the NLP solver in question is being used. The digit `0` indicates: no option file is to be used. The last option file is also used for subsequent NLP subproblems.

Example: `nlpoptfile nlp.opt nlp2.opt 0`
This option will cause the first NLP subproblem solver to read the option file `nlp.opt`, the second one to read the option file `nlp2.opt` and subsequent NLP subproblem solvers will not use any option file.

Example: `nlpoptfile 1`
This will cause the NLP solver for all NLP subproblems to read a default option file (e.g. `conopt.opt`, `minos.opt`, `snopt.opt` etc.).

Option files are located in the current directory (or the project directory when using the IDE). The default is not to use an option file.

nlpreslim x_1 x_2 ... x_n

Sets a resource (time) limit on individual NLP subproblems. The last number x_n is valid for all subsequent cycles $n, n+1, \ldots$. A number -1.0 means that the corresponding NLP subproblem is not individually time restricted. A global time limit is maintained through the GAMS option `reslim`.

Example: `nlpreslim 100 -1`
The first (relaxed) NLP subproblem can not use more than 100 seconds, while subsequent NLP subproblems are not individually restricted.

Example: `nlpreslim 1000`
Sets a time limit of 1000 seconds on all NLP subproblems.

When this option is used it is advised to have the option `continue` set to its default of 2. The default for this option is not to restrict individually the time an NLP solver can spend on an NLP subproblem (other than the global resource limit).

nlpsolver s_1 s_2 ... s_n

This option specifies which NLP solver to use for the NLP subproblems.

Example: `nlpsolver conopt minos snopt`
tells DICOPT to use CONOPT for the relaxed NLP, MINOS for the second NLP subproblem and SNOPT for the third and subsequent ones. The last entry is used for more than one subproblem: for all subsequent ones DICOPT will use the last specified solver.

The names to be used for the solvers are the same as one uses in the GAMS statement `OPTION NLP=....;`. The default is to use the default NLP solver. Note that changing from one NLP solver to another can lead to different results, and may cause DICOPT to follow a different path.

nlptracefile s

Name of the files written if the option `nlptracelevel` is set. Only the stem is needed: if the name is specified as `nlptracefile nlptrace`, then files of the form `nlptrace.001`, `nlptrace.002`, etc. are written. These files contain the settings of the integer variables so that NLP subproblems can be investigated independently of DICOPT. Default: `nlptrace`.

nlptracelevel n

This sets the level for NLP tracing, which writes a file for each NLP sub-problem, so that NLP sub-problems can be investigated outside the DICOPT environment. See also the option `nlptracefile`.

nlptracelevel 0
No trace files are written. This is the default.

nlptracelevel 1
A GAMS file for each NLP subproblem is written which fixes the discrete variables.

nlptracelevel 2
As `nlptracelevel 1`, but in addition level values of the continuous variables are written.

nlptracelevel 3
As `nlptracelevel 2`, but in addition marginal values for the equations and variables are written.

By including a trace file to your original problem, and changing it into an MINLP problem, the subproblem will be solved directly by an NLP solver. This option only works if the names in the model (names of variables and equations) are exported by GAMS. This can be accomplished by using the *m*.`dictfile` model suffix, as in `m.dictfile=1;`. In general it is more convenient to use the `CONVERT` solver to generate isolated NLP models (see section 10.4).

optca x_1 x_2 ... x_n
The absolute optimality criterion for the MIP master problems. The GAMS option `optca` is ignored, as by default DICOPT wants to solve MIP master problems to optimality. To allow to solve large problem, it is possible to stop the MIP solver earlier, by specifying a value for `optca` or `optcr` in a DICOPT option file. With setting a value for `optca`, the MIP solver is instructed to stop as soon as the gap between the best possible integer solution and the best found integer solution is less than x, i.e. stop as soon as

$$|\text{BestFound} - \text{BestPossible}| \leq x \tag{1.10}$$

It is possible to specify a different `optca` value for each cycle. The last number x_n is valid for all subsequent cycles $n, n+1, \ldots$.

Example: `optca 10`
Stop the search in all MIP problems as soon as the absolute gap is less than 10.

Example: `optca 0 10 0`
Sets a nonzero `optca` value of 10 for cycle 2, while all other MIP master problems are solved to optimality.

The default is zero.

optcr x_1 x_2 ... x_n
The relative optimality criterion for the MIP master problems. The GAMS option `optca` is ignored, as by default DICOPT wants to solve MIP master problems to optimality. To allow to solve large problem, it is possible to stop the MIP solver earlier, by specifying a value for `optca` or `optcr` in a DICOPT option file. With setting a value for `optcr`, the MIP solver is instructed to stop as soon as the relative gap between the best possible integer solution and the best found integer solution is less than x, i.e. stop as soon as

$$\frac{|\text{BestFound} - \text{BestPossible}|}{|\text{BestPossible}|} \leq x \tag{1.11}$$

Note that the relative gap can not be evaluated if the best possible integer solution is zero. In those cases the absolute optimality criterion `optca` can be used. It is possible to specify a different `optcr` value for each cycle. The last number x_n is valid for all subsequent cycles $n, n+1, \ldots$.

Example: `optcr 0.1`
Stop the search in all the MIP problems as soon as the relative gap is smaller than 10%.

Example: `optcr 0 0.01 0`
Sets a nonzero `optcr` value of 1% for cycle 2, while all other MIP master problems are solved to optimality.

The default is zero.

relaxed n
In some cases it may be possible to use a known configuration of the discrete variables. Some users have very difficult problems, where the relaxed problem can not be solved, but where NLP sub-problems with the integer variables fixed are much easier. In such a case, if a reasonable integer configuration is known in advance, we can bypass the relaxed NLP and tell DICOPT to directly start with this integer configuration. The integer variables need to be specified by the user before the solve statement by assigning values to the levels, as in `Y.L(I) = INITVAL(I);`.

relaxed 0
The first NLP sub-problem will be executed with all integer variables fixed to the values specified by the user. If you don't assign a value to an integer variable, it will retain it's current value, which is zero by default.

relaxed 1
The first NLP problem is the relaxed NLP problem: all integer variables are relaxed between their bounds. This is the default.

relaxed 2
The first NLP subproblem will be executed with some variables fixed and some relaxed. The program distinguishes the fixed from the relaxed variables by comparing the initial values against the bounds and the tolerance allowed `EPSX`. `EPSX` has a default value of 1.e-3. This can be changed in through the option file.

solvelink n
This option defines the solvelink used for the NLP and MIP subsolver

solvelink 1
Call NLP and MIP solver via script.

solvelink 2
Call NLP and MIP solver via module.

solvelink 5
Call NLP and MIP solver in memory. This is the default.

stop n
This option defines the stopping criterion to be used. The search is always stopped when the (minor) iteration limit (the `iterlim` option), the resource limit (the `reslim` option), or the major iteration limit (see *maxcycles*) is hit or when the MIP master problem becomes infeasible.

stop 0
Do not stop unless an iteration limit, resource limit, or major iteration limit is hit or an infeasible MIP master problem becomes infeasible. This option can be used to verify that DICOPT does not stop too early when using one of the other stopping rules. In general it should not be used on production runs, as in general DICOPT will find often the optimal solution using one of the more optimistic stopping rules.

stop 1
Stop as soon as the bound defined by the objective of the last MIP master problem is worse than the best NLP solution found (a "crossover" occurred). For convex problems this gives a global solution, provided the weights are large enough. This stopping criterion should only be used if it is known or it is very likely that the nonlinear functions are convex. In the case of non-convex problems the bounds of the MIP master problem are not rigorous. Therefore, the global optimum can be cut-off with the setting `stop 1`.

stop 2
Stop as soon as the NLP subproblems stop to improve. This"worsening" criterion is a heuristic. For non-convex problems in which valid bounds can not be obtained the heuristic works often very well. Even on convex problems, in many cases it terminates the search very early while providing an optimal or a very good integer solution. The criterion is not checked before major iteration three.

stop 3
Stop as soon as a crossover occurs or when the NLP subproblems start to worsen. (This is a combination of 1 and 2).

Note: In general a higher number stops earlier, although in some cases stopping rule 2 may terminate the search earlier than rule 1. Section VI shows some experiments with these stopping criteria.

weight x
The value of the penalty coefficients. Default $x = 1000.0$.

9 DICOPT Output

DICOPT generates lots of output on the screen. Not only does DICOPT itself writes messages to the screen, but also the NLP and MIP solvers that handle the sub-problems. The most important part is the last part of the screen output.

In this section we will discuss the output that DICOPT writes to the screen and the listing file using the model `procsel.gms` (this model is part of the GAMS model library). A DICOPT log is written there and the reason why DICOPT terminated.

```
--- DICOPT: Checking convergence
--- DICOPT: Search stopped on worsening of NLP subproblems
--- DICOPT: Log File:
 Major Major     Objective    CPU time  Itera- Evaluation Solver
 Step  Iter       Function     (Sec)    tions   Errors
  NLP    1         5.35021      0.05        8      0      conopt
  MIP    1         2.48869      0.28        7      0      cplex
  NLP    2         1.72097<     0.00        3      0      conopt
  MIP    2         2.17864      0.22       10      0      cplex
  NLP    3         1.92310<     0.00        3      0      conopt
  MIP    3         1.42129      0.22       12      0      cplex
  NLP    4         1.41100      0.00        8      0      conopt
--- DICOPT: Terminating...
--- DICOPT: Stopped on NLP worsening

      The search was stopped because the objective function
      of the NLP subproblems started to deteriorate.

--- DICOPT: Best integer solution found: 1.923099
--- Restarting execution
--- PROCSEL.GMS(98) 0 Mb
--- Reading solution for model process
*** Status: Normal completion
```

Notice that the integer solutions are provided by the NLP's except for major iteration one (the first NLP is the relaxed NLP). For all NLP's except the relaxed one, the binary variables are fixed, according to a pattern determined by the previous MIP which operates on a linearized model. The integer solutions marked with a '<' are an improvement. We see that the NLP in cycle 4 starts to deteriorate, and DICOPT stops based on its default stopping rule.

It should be noted that if the criterion `stop 1` had been used the search would have been terminated at iteration 3. The reason is that the upper bound to the profit predicted by the MIP (1.42129) exceeds the best current NLP solution (1.9231). Since it can be shown that the MINLP involves convex nonlinear functions, 1.9231 is the global optimum and the criterion `stop 1` is rigorous.

A similar output can be found in the listing file:

```
               S O L V E      S U M M A R Y

     MODEL   process             OBJECTIVE  pr
     TYPE    MINLP               DIRECTION  MAXIMIZE
     SOLVER  DICOPT              FROM LINE  98

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      8 INTEGER SOLUTION
**** OBJECTIVE VALUE                1.9231
```

```
 RESOURCE USAGE, LIMIT          0.771      1000.000
 ITERATION COUNT, LIMIT        51         10000
 EVALUATION ERRORS              0             0

--- DICOPT: Stopped on NLP worsening

      The search was stopped because the objective function
      of the NLP subproblems started to deteriorate.

 ---------------------------------------------------------------------
 Dicopt2x-C    Jul  4, 2001 WIN.DI.DI 20.1 026.020.039.WAT
 ---------------------------------------------------------------------

     Aldo Vecchietti and Ignacio E. Grossmann
     Engineering Design Research Center
     Carnegie Mellon University
     Pittsburgh, Pennsylvania 15213

     Erwin Kalvelagen
     GAMS Development Corp.
     1217 Potomac Street, N.W.
     Washington DC 20007
 ---------------------------------------------------------------------
 DICOPT Log File
 ---------------------------------------------------------------------
 Major Major     Objective    CPU time  Itera- Evaluation Solver
 Step  Iter       Function     (Sec)    tions   Errors
  NLP    1         5.35021      0.05        8      0      conopt
  MIP    1         2.48869      0.28        7      0      cplex
  NLP    2         1.72097<     0.00        3      0      conopt
  MIP    2         2.17864      0.22       10      0      cplex
  NLP    3         1.92310<     0.00        3      0      conopt
  MIP    3         1.42129      0.22       12      0      cplex
  NLP    4         1.41100      0.00        8      0      conopt
 ---------------------------------------------------------------------
   Total solver times :  NLP =     0.05   MIP =     0.72
   Perc. of total     :  NLP =     6.59   MIP =    93.41
 ---------------------------------------------------------------------
```

In case the DICOPT run was not successful, or if one of the subproblems could not be solved, the listing file will contain all the status information provided by the solvers of the subproblems. Also for each iteration the configuration of the binary variables will be printed. This extra information can also be requested via the GAMS option:

```
option sysout = on ;
```

10 Special Notes

This section covers some special topics of interest to users of DICOPT.

10.1 Stopping Rule

Although the default stopping rule behaves quite well in practice, there some cases where it terminates too early. In this section we discuss the use of the stopping criteria.

When we run the example `procsel.gms` with stopping criterion 0, we see the following DICOPT log:

```
--- DICOPT: Starting major iteration 10
--- DICOPT: Search terminated: infeasible MIP master problem
--- DICOPT: Log File:
 Major Major     Objective    CPU time  Itera- Evaluation Solver
 Step  Iter       Function     (Sec)    tions   Errors
  NLP    1         5.35021      0.06        8      0      conopt
  MIP    1         2.48869      0.16        7      0      cplex
  NLP    2         1.72097<     0.00        3      0      conopt
  MIP    2         2.17864      0.10       10      0      cplex
  NLP    3         1.92310<     0.00        3      0      conopt
  MIP    3         1.42129      0.11       12      0      cplex
  NLP    4         1.41100      0.00        8      0      conopt
  MIP    4         0.00000      0.22       23      0      cplex
  NLP    5         0.00000      0.00        3      0      conopt
  MIP    5        -0.27778      0.16       22      0      cplex
  NLP    6        -0.27778      0.00        3      0      conopt
  MIP    6        -1.00000      0.16       21      0      cplex
  NLP    7        -1.00000      0.00        3      0      conopt
  MIP    7        -1.50000      0.22       16      0      cplex
  NLP    8        -1.50000      0.00        3      0      conopt
  MIP    8        -2.50000      0.11       16      0      cplex
  NLP    9        -2.50000      0.00        3      0      conopt
  MIP    9        *Infeas*      0.11        0      0      cplex
--- DICOPT: Terminating...
--- DICOPT: Stopped on infeasible MIP

      The search was stopped because the last MIP problem
      was infeasible. DICOPT will not be able to find
      a better integer solution.

--- DICOPT: Best integer solution found: 1.923099
--- Restarting execution
--- PROCSEL.GMS(98) 0 Mb
--- Reading solution for model process
*** Status: Normal completion
```

This example shows some behavioral features that are not uncommon for other MINLP models. First, DICOPT finds often the best integer solution in the first few major iterations. Second, in many cases as soon as the NLP's start to give worse integer solution, no better integer solution will be found anymore. This observation is the motivation to make stopping option 2 where DICOPT stops as soon as the NLP's start to deteriorate the default stopping rule. In this example DICOPT would have stopped in major iteration 4 (you can verify this in the previous section). In many cases this will indeed give the best integer solution. For this problem, DICOPT has indeed found the global optimum.

Based on experience with other models we find that the default stopping rule (stop when the NLP becomes worse) performs well in practice. In many cases it finds the global optimum solution, for both convex and non-convex problems. In some cases however, it may provide a sub-optimal solution. In case you want more reassurance that no good integer solutions are missed you can use one of the other stopping rules.

Changing the MIP or NLP solver can change the path that DICOPT follows since the sub-problems may have non-unique solutions. The optimum stopping rule for a particular problem depends on the MIP and NLP solvers used.

In the case of non-convex problems the bounds of the MIP master problem are not rigorous. Therefore, the global optimum can be cut-off with `stop 1`. This option is however the best stopping criterion for convex problems.

10.2 Solving the NLP Problems

In case the relaxed NLP and/or the other NLP sub-problems are very difficult, using a combination of NLP solvers has been found to be effective. For example, MINOS has much more difficulties to establish if a model is infeasible, so one would like to use CONOPT for NLP subproblems that are either infeasible or barely feasible. The `nlpsolver` option can be used to specify the NLP solver to be used for each iteration.

Infeasible NLP sub-problems can be problematic for DICOPT. Those subproblems can not be used to form a new linearization. Effectively only the current integer configuration is excluded from further consideration by adding appropriate integer cuts, but otherwise an infeasible NLP sub-problem provides no useful information to be used by the DICOPT algorithm. If your model shows many infeasible NLP sub-problems you can try to to use the `infeasder` option. Otherwise a strategy that can help is to introduce explicit slack variables and add them with a penalty to the objective function.

Assume your model is of the form:

$$\begin{aligned} \min\ & f(x,y) \\ & g(x,y) \sim b \\ & \ell \le x \le u \\ & y \in \{0,1\} \end{aligned} \tag{1.12}$$

where $\sim$ is a vector of relational operators $\{\le, =, \ge\}$. x are continuous variables and y are the binary variables. If many of the NLP subproblems are infeasible, we can try the following "elastic" formulation:

$$\begin{aligned} \min\ & f(x,y) + M \sum_i (s_i^+ + s_i^-) \\ & y = y^B + s^+ - s^- \\ & g(x,y) \sim b \\ & \ell \le x \le u \\ & 0 \le y \le 1 \\ & 0 \le s^+, s^- \le 1 \\ & y^B \in \{0,1\} \end{aligned} \tag{1.13}$$

I.e. the variables y are relaxed to be continuous with bounds $[0,1]$, and binary variables y^B are introduced, that are related to the variables y through a set of the slack variables s^+, s^-. The slack variables are added to the objective with a penalty parameter M. The choice of a value for M depends on the size of $f(x,y)$, on the behavior of the model, etc. Typical values are 100, or 1000.

10.3 Solving the MIP Master Problems

When there are many discrete variables, the MIP master problems may become expensive to solve. One of the first thing to try is to see if a different MIP solver can solve your particular problems more efficiently.

Different formulations can have dramatic impact on the performance of MIP solvers. Therefore it is advised to try out several alternative formulations. The use of priorities can have a big impact on some models. It is possible to specify a nonzero value for `OPTCA` and `OPTCR` in order to prevent the MIP solver to spend an unacceptable long time in proving optimality of MIP master problems.

If the MIP master problem is infeasible, the DICOPT solver will terminate. In this case you may want to try the same reformulation as discussed in the previous paragraph.

10.4 Model Debugging

In this paragraph we discuss a few techniques that can be helpful in debugging your MINLP model.

- Start with solving the model as an `RMINLP` model. Make sure this model solves reliably before solving it as a proper `MINLP` model. If you have access to different NLP solvers, make sure the `RMINLP` model solves smoothly with all NLP solvers. Especially CONOPT can generate useful diagnostics such as Jacobian elements (i.e. matrix elements) that become too large.
- Try different NLP and MIP solvers on the subproblems. Example: use the GAMS statement "`OPTION NLP=CONOPT3;`" to solve all NLP subproblem using the solver CONOPT version 3.
- The GAMS option statement "`OPTION SYSOUT = ON;`" can generate extra solver information that can be helpful to diagnose problems.
- If many of the NLP subproblems are infeasible, add slacks as described in section 10.2.
- Run DICOPT in pedantic mode by using the DICOPT option: "`CONTINUE 0`." Make sure all NLP subproblems solve to optimality.
- Don't allow any nonlinear function evaluation errors, i.e. keep the `DOMLIM` limit at zero. See the discussion on `DOMLIM` in section 7.1.
- If you have access to another MINLP solver such as SBB, try to use a different solver on your model. To select SBB use the following GAMS option statement: "`OPTION MINLP=SBB;`."
- Individual NLP or MIP subproblems can be extracted from the MINLP by using the `CONVERT` solver. It will write a model in scalar GAMS notation, which can then be solved using any GAMS NLP or MIP solver. E.g. to generate the second NLP subproblem, you can use the following DICOPT option: "`NLPSOLVER CONOPT CONVERT`." The model will be written to the file `GAMS.GMS`. A disadvantage of this technique is that some precision is lost due to the fact that files are being written in plain ASCII. The advantage is that you can visually inspect these files and look for possible problems such as poor scaling.

DICOPT References

[1] M. A. Duran and I. E. Grossmann, *An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs*, Mathematical Programming, 36 (1986), pp. 307–339.

[2] E. Kalvelagen, *Model building with GAMS*, to appear.

[3] G. R. Kocis and I. E. Grossmann, *Relaxation Strategy for the Structural Optimization of Process Flowsheets*, Industrial and Engineering Chemistry Research, 26 (1987), pp. 1869–1880.

[4] G. R. Kocis and I. E. Grossmann, *Computational Experience with DICOPT solving MINLP Problems in Process Systems Engineering*, Computers and Chemical Engineering, 13 (1989), pp. 307–315.

[5] J. Viswanathan and I. E. Grossmann, *A combined Penalty Function and Outer Approximation Method for MINLP Optimization*, Computers and Chemical Engineering, 14 (1990), pp. 769–782.

[6] H. P. Williams, *Model Building in Mathematical Programming*, 4-th edition (1999), Wiley.