
AIMMS Tutorial for Professionals - Production and Maintenance Model

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

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 7

Production and Maintenance Model

In this chapter you will enter all the model identifiers that are related to the mathematical model described in Chapter ???. As with most realistic models, the number of identifiers is quite large, and it pays to refine the model tree by declaring several additional declaration sections.

This chapter

7.1 Model structure

Please add the following declaration subsections to the section named Production and Maintenance Model:

Adding to the model tree

- Location Declarations
- Scenario Declarations
- Production Declarations
- Supply and Demand Declarations
- Maintenance and Vacation Declarations
- Cost Declarations
- Mathematical Program Declarations

The resulting section in the model tree is shown in Figure 7.1.

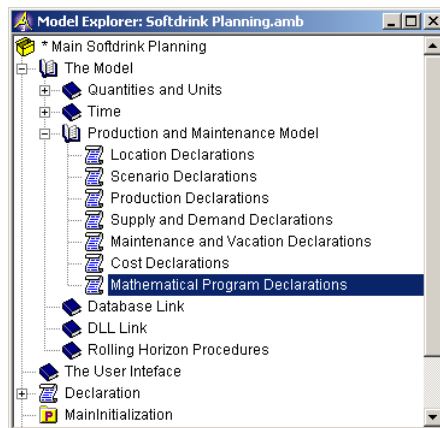


Figure 7.1: Seven declaration sections to increase model structure

Most of the declarations in this chapter are provided in a compact textual format that closely corresponds to the attribute format presented in previous chapters. Instead of providing detailed instructions, you are asked to complete the attribute forms on the basis of what you have learned so far.

*Proceed with
little assistance
...*



As explained in Chapter ??, you can avoid entering some, or all, of the declarations in this chapter by importing the model section 'Production and Maintenance Model.amb' into the Production and Maintenance Model section in the model tree. You are advised to at least examine the declarations listed in the remainder of this chapter if you choose the import file option.

*... or import all
identifier
declarations*

7.2 Topology

In Chapter ?? you already declared the set of locations and their corresponding x - and y -coordinates. You should now move these existing identifiers to their new position in the model tree by performing the following actions:

*Moving existing
identifiers*

- ▶ open the declaration section Declaration at the end of the model tree,
- ▶ select all three identifiers using the *Shift* key in combination with the left-mouse button,
- ▶ press the **Cut** button  on the toolbar,
- ▶ select and open the section named Location Declarations, and
- ▶ press the **Paste** button  on the toolbar.

As an alternative, you could have dragged the three identifiers to their new position.

In the Location Declarations section you can now declare the sets Factories and Centers as subsets of the set of all locations.

*Entering
location
declarations*

```
SET:
  identifier : Factories
  subset of  : Locations
  index      : f
```

```
SET:
  identifier : Centers
  subset of  : Locations
  index      : c
```

7.3 Demand scenarios

The following two identifiers need to be added to the section Scenario Declarations, and are used to set up the demand scenarios. Note that when a particular set has a fixed number of known elements, you can enter these elements as data in the **Definition** attribute (see the set Scenarios below).

*Entering
scenario
declarations*

```

SET:
  identifier : Scenarios
  index      : s
  definition : data { pessimistic, expected, optimistic }

PARAMETER:
  identifier : ScenarioProbability
  index domain : s
  definition : data { pessimistic : 0.25, expected : 0.50, optimistic : 0.25 }

```

7.4 Production

An interesting feature of AIMMS is that you can specify a global index domain condition as illustrated in the last three declarations below. In these examples, AIMMS will only consider the (f, p) combinations that exist. All other combinations will be ignored throughout the application. Note that the ‘|’ operator is to be interpreted as the ‘such that’ operator, and that the $\text{Ord}(p)$ operator returns the ordinal position of the corresponding element p within its domain set `ProductionLines`.

*Data definition
and domain
condition*

Please open the Production Declarations subsection, and enter the following declarations after having read the previous paragraph.

*Entering
production
declarations*

```

SET:
  identifier : ProductionLines
  index      : p
  definition : data { line-01 .. line-04 }

PARAMETER:
  identifier : NumberOfProductionLines
  index domain : f

SET:
  identifier : FactoryProductionLines
  index domain : f
  subset of : ProductionLines
  definition : { p | Ord(p) <= NumberOfProductionLines(f) }

PARAMETER:
  identifier : DeteriorationLevel
  index domain : (f,p) | p in FactoryProductionLines(f)

PARAMETER:
  identifier : DeteriorationLevelAtStartOfCalendar
  index domain : (f,p) | p in FactoryProductionLines(f)

PARAMETER:
  identifier : MaximumDeteriorationLevel
  index domain : (f,p) | p in FactoryProductionLines(f)



```

The **Unit** wizard can only complete the **Unit** attribute for you once either the desired unit or the unit expression has been entered. Therefore, when declaring the first of the two parameters below, you should enter the **Unit** attribute [hl/day] manually. When declaring the second parameter, you can use the **Unit** wizard and select the 'Implicit Quantities' entry.

Entering unit expressions once

```
PARAMETER:
  identifier : ProductionLineLevelAtStartOfCalendar
  index domain : (f,p) | p in FactoryProductionLines(f)
  unit : hl/day
```

```
PARAMETER:
  identifier : MaximumProductionLineLevel
  index domain : (f,p) | p in FactoryProductionLines(f)
  unit : hl/day
```

Once you have entered the declaration listed below, AIMMS still cannot compile the definition of the parameter `PotentialProduction`. This definition contains a reference to the three identifiers `LineInMaintenance`, `DropDueToVacation` and `IsVacationPeriod`, which have not yet been declared. In such a situation, you should use the **Commit and close** button  instead of the **Check, commit and close** button , and AIMMS will not complain (though the identifier names will be colored red). Instructing AIMMS to compile the model will result in errors reporting missing identifiers. The three identifiers will be declared at a later stage.

Postponing identifier checking

```
PARAMETER
  identifier : PotentialProduction
  index domain : (f,p,t) | p in FactoryProductionLines(f)
  unit : hl
  definition : ActualNumberOfDaysInPeriod(t) *
              ( 1 - LineInMaintenance(f,p,t) ) *
              ( 1 - DropDueToVacation * IsVacationPeriod(f,t) ) *
              MaximumProductionLineLevel(f,p)
```

An overview of all the declarations entered by you so far is shown in Figure 7.2.

Initial overview

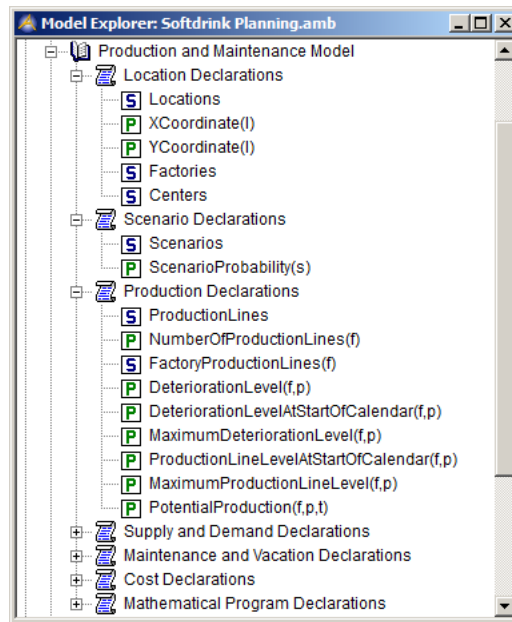


Figure 7.2: All location, scenario and production declarations

7.5 Supply and demand

Use the **Unit** wizard, the **Range** wizard, and the name completion functionality to enter the following supply and demand declarations in the appropriate declaration section of your model tree.

Entering supply and demand declarations

```
PARAMETER:
  identifier : Demand
  index domain : (c,t,s)
  unit : h1
```

```
PARAMETER:
  identifier : MinimumStock
  index domain : 1
  unit : h1
```

```
PARAMETER:
  identifier : MaximumStock
  index domain : 1
  unit : h1
```

```

PARAMETER:
  identifier : StockAtStartOfCalendar
  index domain : l
  unit      : hl

PARAMETER:
  identifier : MaximumTransportCapacity
  index domain : f
  unit      : TL

```

7.6 Maintenance and vacations

As was already mentioned in Chapter ??, most of the computations needed for maintenance planning can be placed outside the mathematical program. All you need to declare at this point is when a particular production line is undergoing maintenance. Please use the Maintenance and Vacation Declarations declaration section in your model tree.

*Maintenance
declaration*

```

PARAMETER:
  identifier : LineInMaintenance
  index domain : (f,p,t)
  range      : binary

```

The management of each factory knows the particular weeks in which a relatively large portion of its personnel will be on leave. During these weeks, production typically drops by 40% of the maximum production capacity. The following two parameters need to be declared.

*Entering
vacation
declarations*

```

SET:
  identifier : VacationWeeks
  index domain : f
  subset of : Weeks

PARAMETER:
  identifier : DropDueToVacation
  initial data : 0.4

PARAMETER:
  identifier : IsVacationPeriod
  index domain : (f,t)
  range      : binary
  definition : if ( WeekInPeriod(t) in VacationWeeks(f) )
               then 1
               else 0
               endif

```

At this point you should be able to compile the entire model, because the three identifiers missing in section 7.4 have now been declared. To compile the entire model you should execute the **Compile All** command from the **Run** menu. Alternatively, you could simply press the *F5* key. Please ignore any warnings concerning data initialization.

Compiling the entire model

7.7 Costs

The total scenario cost is divided into four components, each of which has its own unit cost declaration. The total cost is the weighted sum of the total scenario cost over all scenarios. You should enter the following declarations in the Cost Declarations section in your model tree.

Entering cost declarations

```
PARAMETER:
  identifier : UnitTransportCost
  index domain : (f,c)
  unit       : $/TL

PARAMETER:
  identifier : FixedCostDueToLevelChange
  unit       : $
  initial data : 25000

PARAMETER:
  identifier : UnitStockCost
  index domain : 1
  unit       : $/h1

PARAMETER:
  identifier : UnitProductionCost
  index domain : f
  unit       : $/h1
```

An overview of all supply and demand, maintenance and vacation, and cost declarations entered is shown in Figure 7.3.

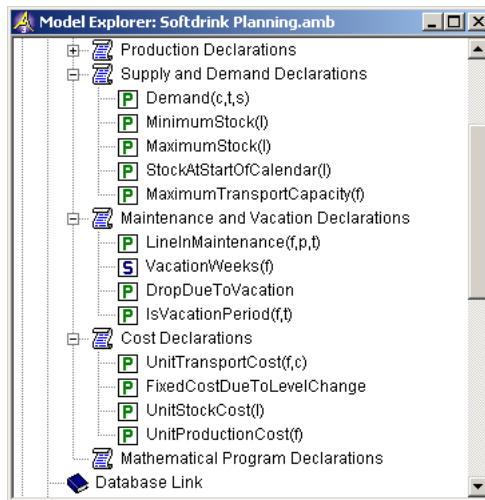


Figure 7.3: All supply, demand, maintenance, vacation, and cost declarations

7.8 Optimization model

There are seven variables in the formulation of the mathematical program in this tutorial. These should be entered in the declaration section Mathematical Program Declarations in your model tree.

Variables

```
VARIABLE:
  identifier : ProductionLineInUse
  index domain : (f,p,t) | p in FactoryProductionLines(f)
  range       : binary
```

```
VARIABLE:
  identifier : Production
  index domain : (f,t)
  range       : nonnegative
  unit       : h]
  definition  : sum[ p, PotentialProduction(f,p,t) *
                 ProductionLineInUse(f,p,t) ]
```

```
VARIABLE:
  identifier : ProductionLineLevelChange
  index domain : (f,p,t) | p in FactoryProductionLines(f)
  range       : [0, 1]
```

```
VARIABLE:
  identifier : Transport
  index domain : (f,c,t,s) | UnitTransportCost(f,c)
  range       : [0, MaximumStock(c)]
  unit       : TL
```

```
VARIABLE:
  identifier : Stock
  index domain : (l,t,s)
```

```

range      : [MinimumStock(l), MaximumStock(l)]
unit       : h1
definition  : Stock(l,t-1,s) + Production(l,t) +
              sum[ f, Transport(f,l,t,s) ] -
              sum[ c, Transport(l,c,t,s) ] - Demand(l,t,s)

VARIABLE:
identifier  : TotalScenarioCost
index domain : s
unit       : $
definition  : sum[ (f,p,t), FixedCostDueToLevelChange *
                    ProductionLineLevelChange(f,p,t) ] +
              sum[ (f,t), UnitProductionCost(f) *
                    Production(f,t) ] +
              sum[ (l,t), UnitStockCost(l) *
                    Stock(l,t,s) ] +
              sum[ (f,c,t), UnitTransportCost(f,c) *
                    Transport(f,c,t,s) ]

VARIABLE:
identifier  : TotalCost
unit       : $
definition  : sum[ s, ScenarioProbability(s) * TotalScenarioCost(s) ]

```

Note that four of the variables have their own definitions. AIMMS will treat these definitions as constraints when generating the corresponding mathematical program.

Defined variables

The remaining three constraints in the formulation of the mathematical program are listed below.

Constraints

```

CONSTRAINT:
identifier  : ChangeWhenIncrease
index domain : (f,p,t) | p in FactoryProductionLines(f)
definition  : ProductionLineLevelChange(f,p,t) >=
              ProductionLineInUse(f,p,t) - ProductionLineInUse(f,p,t-1)

CONSTRAINT:
identifier  : ChangeWhenDecrease
index domain : (f,p,t) | p in FactoryProductionLines(f)
definition  : ProductionLineLevelChange(f,p,t) >=
              ProductionLineInUse(f,p,t-1) - ProductionLineInUse(f,p,t)

CONSTRAINT:
identifier  : RestrictTransportCapacity
index domain : (f,t,s)
unit       : TL
definition  : sum[ c, Transport(f,c,t,s) ] <= MaximumTransportCapacity(f)

```

A mathematical program in AIMMS specifies the set of variables and constraints together with the objective, optimization direction and model type that are needed by AIMMS to generate the model. If you do not specify a variable set or a constraint set, AIMMS will assume that all model variables and all model constraints are included in the mathematical program. Please use

Mathematical program

the **Objective**, the **Direction** and the **Type** wizard to declare the mathematical program `LeastCostPlan` as shown in Figure 7.4.

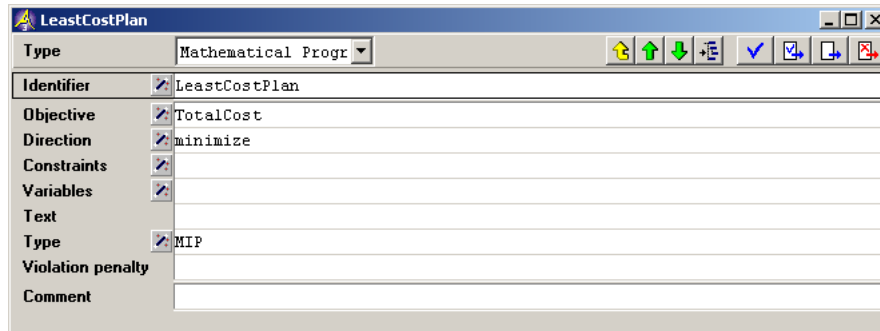


Figure 7.4: Attribute form of the mathematical program

All variables and constraints that are declared in the Mathematical Program *Model tree* Declarations are shown in Figure 7.5.

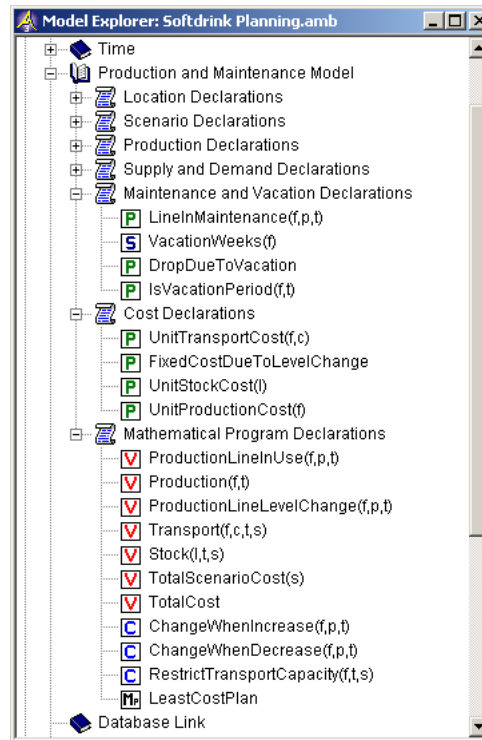


Figure 7.5: The model tree to date