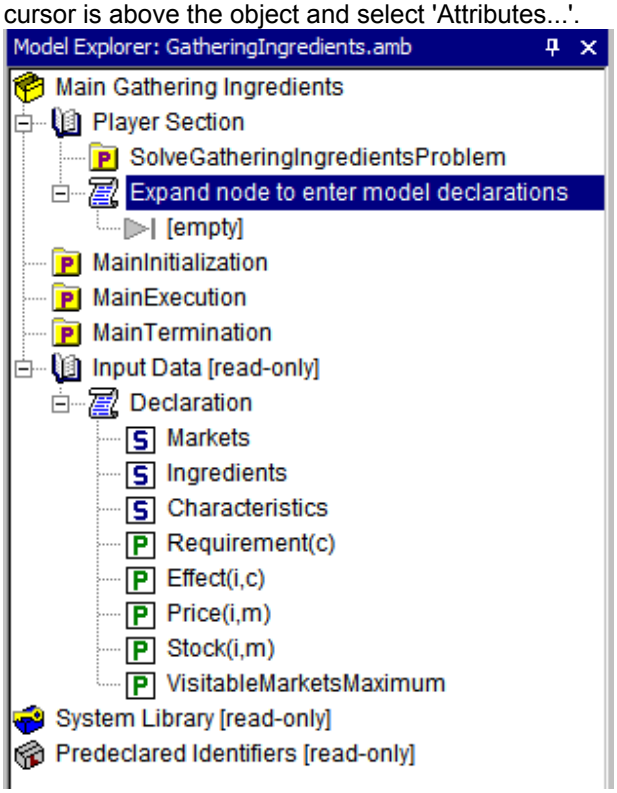
Tutorial for G-AIMMS: Witch Apprentice - Gathering Ingredients

Introduction to G-AIMMS
G-AIMMS is a series of AIMMS based games aimed to extend your modeling skills and AIMMS knowledge. G-AIMMS: Witch Apprentice - Gathering Ingredients is the first G-AIMMS. First-time AIMMS users are strongly recommended to read this tutorial before starting to play G-AIMMS, since it explains the basics of AIMMS in a much shorter time than it will take you to discover them on your own. You can always review this tutorial while playing G-AIMMS. It is located on the Help page.

Solving a problem in AIMMS
To solve a problem in AIMMS, you only need to worry about formulating a good model. Finding the solution to your model is easy using one of the solvers included with AIMMS. AIMMS clearly differentiates between the model and the data. In this level all the data you need is provided to you at the start of the level via predeclared identifiers located in the Input Data section of the model. Your task is to formulate a good model.

Using the Model Explorer
To help you formulate your model, you can use the Model Explorer. The Model Explorer provides a graphical overview of your complete model. It allows you to declare your own identifiers, procedures and functions and to organize your model in multiple sections. This first level contains a button on the interface pages to open the model explorer. Not all sublevels also contain this button, but you can also open the Model Explorer via the shortcut key F8, the Tools menu or the Model Explorer toolbar button.
To open or close a section, double-click the icon or use the '+' or '-' in front of it. To open the attributes page of an identifier, procedure, function or section, either double-click its name or select it and click the  icon at the top of your screen or press the right mouse button while your cursor is above the object and select 'Attributes...'.



Declaring new identifiers
New identifiers to use in the model can be declared in the Declaration section (called "Expand node to enter model declarations") in the Player Section. Identifiers always have to be declared in a Declaration section. To declare a new identifier, open the Player Section and then open the Declaration section. In the Toolbar at the top of your screen the following icons should be enabled:  . The first four icons represent the most frequently used identifier types, respectively sets, parameters, variables and constraints. If you want to declare an identifier of one of these types, you can click on the corresponding icon. If you want to declare another type of identifier, you can use the fifth icon, allowing you to choose between all possible types of identifiers. After you have selected the type of identifier you want to declare, you have to provide a name. For the name you can use all letters a-z, all digits 0-9 and the underscore '_', however a name can not start with a digit. To keep your identifier names intuitive use descriptive names like 'MaximumTruckCapacity' instead of short names like 'c'. After you have named your identifier, open the attributes page to change some of the identifier attributes. If you want to change one of the attributes, but are unsure how to do so, you can click on the wizard icon  . This will open a wizard dialog, which guides you through the necessary steps to change the attribute. After you have made your changes, use the check and commit icon  or the check, commit and close icon  to check the changes for errors and to apply them to the model. Below you can find descriptions of the most frequently used identifier types.

Sets
A set is a finite collection of elements. These elements can be strings or integers. With each set, indices can be associated. With these indices you can refer to the elements of the set in other identifiers, functions or procedures. For example, suppose you have declared a set Cities with associated index c. You can now declare a parameter Demand(c) which represents the demand for each city. Below you can find a description of the most frequently used attributes of sets:

Index domain: Leaving this attribute empty will create a scalar set. Should you want to declare an indexed set, that is a different set for each of the elements of the set associated with the index/indices, you can specify one or more indices here. For example, specifying the index c in this field allows you to declare a set NeighbourCities(c) that specifies for each of the cities c a set with all cities close to c.

Subset of: If you specify another set S here, the set will be a subset of S. You can also select a predeclared set, for example the set Integers, to make your set a subset of all integers.

Index: Here you can provide all indices you want to associate with your set.

Parameters
A parameter can hold numerical values. You can use parameters for example, to provide input data to your model. Suppose you have declared a parameter Demand(c) with the demand input data for each city c. You can later use this data by using the parameter Demand(c) in your model formulation. Below you can find a description of the most frequently used attributes of parameters:

Index domain: Leaving this field empty will create a scalar parameter. If you specify one or more indices here, you can declare indexed parameters. For example, specifying c in this field allows you to declare a parameter Demand(c) that represents the demand for each city c. You can also specify a more complex index domain. Suppose i and j are indices associated with the set Cities. You can declare a parameter Distance(i,j) with index domain (i,j)|i<>j that represents the distance between two cities i and j. This index domain makes sure the parameter is defined for all pairs of cities (i,j) under the condition that i is unequal to j.

Range: In this field you can specify the range of values the parameter is allowed to take. You can specify whether the parameter is continuous, meaning it can take all real values, or integer. You can also specify the lower and upper bounds on the values the parameter can take.

Definition: In this field you can give a definition of the parameter. You can for example declare a parameter LargestDemand and give it the definition 'max(c,Demand(c))'. Note that if you give a definition to the parameter, its value will be equal to the definition and you will not be able to assign a different value to it. If you use a definition for a parameter, AIMMS makes sure its value is always up-to-date when it is used.

Variables

A variable is similar to a parameter. The main difference is that parameters typically have known values, while variables represent unknown quantities and are assigned values by the solver. Variables typically represent decisions that have to be taken in a model. For example, in a transport model with cities that have a certain demand and factories that have a certain supply, you can declare a variable Transport(f,c) that represents how much to transport from factory f to city c. Below you can find a description of the most frequently used attributes of variables:

Index domain: With this field you can declare indexed variables in a similar way as you can declare indexed parameters (see above).

Range: You can specify which values the variable is allowed to take. One of the ranges that is often useful is the binary range, which allows the variable to take only the values 0 or 1. A binary variable can be used to represent a yes/no decision. For example, suppose you have a number of possible depot locations d. You can declare a binary variable OpenDepot(d) that represents whether a depot is opened on depot location d. In this case, it will take the value 1 and otherwise it will take the value 0.

Definition: Here you can define a variable in terms of other variables and parameters. For example, you can define the variable LargestTransport as 'max((f,c),Transport(f,c))'. Alternatively, you also could have declared a constraint with definition 'LargestTransport = max((f,c),Transport(f,c)). *Note the non-linearity of this expression*!!

Constraints

Constraints are used to specify relationships between combinations of variables and parameters. These relationships restrict the values the variables in your model can take. For example, suppose you have declared the variable OpenDepot(d) (see above for its interpretation). If you don't want to open more depots than the value of the parameter MaximumNumberOfDepots, you can declare a constraint NumberOfDepotsConstraint with the definition 'sum(d,OpenDepot(d))<=MaximumNumberOfDepots'. Below you can find a description of the most frequently used attributes of constraints:

Index domain: With this field you can declare indexed constraints. For example, if you want to make sure the transport to each city is at least the demand of the city, you can declare the indexed constraint DemandConstraint(c) with definition 'sum(f,Transport(f,c))>=Demand(c)'. The index domain c of the constraint makes sure that the constraint is created for each element in the set associated with index c, which is the set Cities.

Definition: In this field you can specify a relationship between some of the variables and parameters in your model. A constraint definition always consists of two or three expressions separated by one of the relational operators '=', '<=' or '>='.

Mathematical programs

To find an optimal solution to your model, you have to declare a mathematical program. A mathematical program consists of an objective variable that has to be optimized under the restriction that all variables have to take values in their ranges and all constraints have to be

satisfied. Below you can find a description of the most frequently used attributes of mathematical programs:

Objective: In this field you specify the variable that you want to optimize. Note that this should always be a scalar variable. Valid options would be for example 'TotalCost' or 'Transport('Factory 1',Amsterdam)', but not 'Transport(f,c)' since this variable contains free indices.

Direction: In this field you specify whether you want to minimize or maximize the objective variable.

Declaring and using new procedures
New procedures to use in the model can be declared in the Player Section. Make sure you have opened the player section and you are not inside a declaration section. In the toolbar at the top of your screen the  icon should be enabled. Click on the icon to declare a new procedure. After you have named the procedure, open the attribute page. In the body field you can specify the sequential steps the procedure has to execute when it is run.

Procedures can be used to execute a number of steps in order. In a procedure you can use most of the statements you can use in other programming languages. In the next section are some hints on how to use the AIMMS language. Suppose for example you have created a mathematical program TransportModel that when solved, assigns values to the variables Transport(f,c) in such a way that the total cost of all transport is minimized and suppose that after the mathematical program is solved, you want to write the data associated with the identifier Transport to a file named "TransportData.txt". To do so you could declare a procedure with the following body:

```
solve TransportModel;
write Transport to file "TransportData.txt";
```

To actually run a procedure, press the right mouse button while your mouse cursor is above it in the Model Explorer and select 'Run Procedure'.

Using the AIMMS language
Below you can find some hints on how to use the AIMMS language:
- Statements should be ended with a semicolon: ';'.
- Assignments can be made with the ':=' operator, for example 'MaximumNumberOfDepots := 5;'.
- If you want to make a simple assignment for all elements in a domain, it is not necessary to use a 'for loop'. For example, suppose you want to assign the Euclidean distance to the parameter Distance(i,j) for each pair of cities (i,j). Instead of the statement

```
for (i,j) do
        Distance(i,j) := sqrt(sqr(Coordinate(i,'x')-Coordinate(j,'x'))+sqr(Coordinate(i,'y')-
Coordinate(j,'y')));
endfor;
```

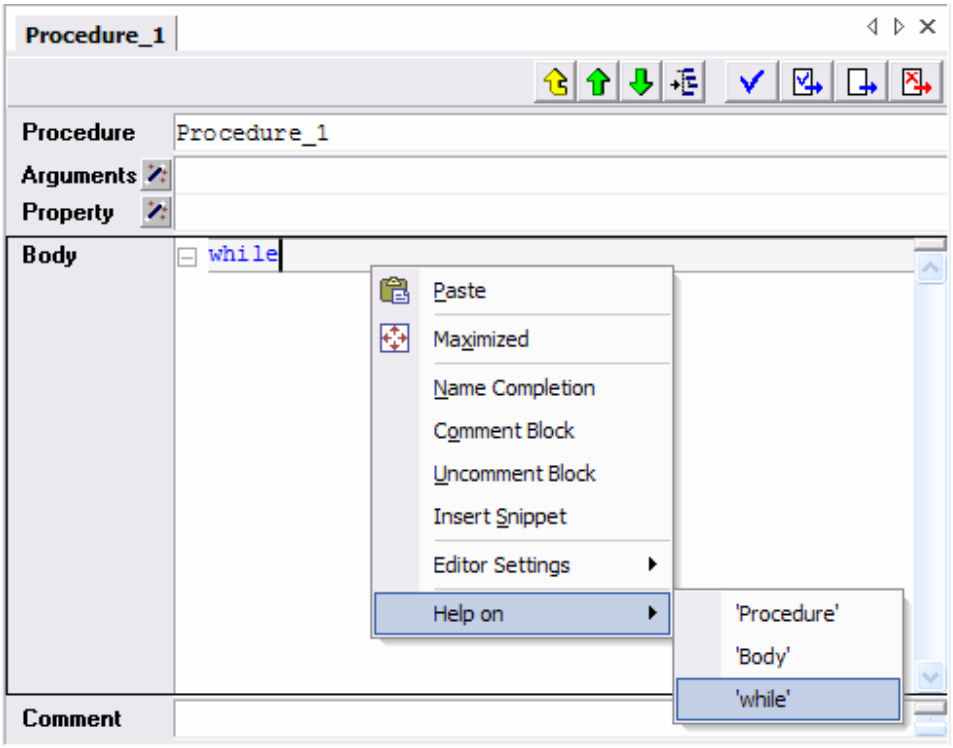
you can simply use the statement 'Distance(i,j) := sqrt(sqr(Coordinate(i,'x')-
Coordinate(j,'x'))+sqr(Coordinate(i,'y')-Coordinate(j,'y')));'.

Note that this statement is shorter, but it also executes much faster.
- AIMMS has a name completion feature to prevent you from having to type long identifier names and to prevent typing mistakes. Just type the first few letters of an identifier name and press CTRL+SPACEBAR to use it. You can also use CTRL+SHIFT+SPACEBAR to use name completion for predeclared identifiers, functions and procedures.
- AIMMS supports 'if', 'for' and 'while' statements. See the AIMMS Documentation for the exact syntax.

Viewing and changing data associated with identifiers
To view the data associated with an identifier, press the right mouse button while your mouse cursor is above the identifier in the Model Explorer and select 'Data...'. Alternatively, you can open the attributes page associated with the identifier and click the  icon. On the data page, you can see all data associated with the identifier and you can also change this data. Note that all input data for this level is provided in the Input Data section in the Model Explorer. If you want to import large amounts of data, other possibilities are available, for example reading data from a file or from a database.

Playing this G-AIMMS Level
Below is an overview of the steps you have to take to complete this G-AIMMS level:
- Open the file 'GatheringIngredients.aimms'. This will start this G-AIMMS level. For this G-AIMMS level you can use your own AIMMS license, a free trial license, or a free student license.
- Take a look at the story, the problem description and the input data to get an impression of the problem.
- Open the Model Explorer. Take a look at all identifiers declared in the Declaration section located in the Input Data section. You can use all these identifiers in your model, for example in the definition of constraints. All data for these identifiers is already provided, as you can see when you open the data pages of these identifiers.
- Now the time has come to build your model. You can declare your identifiers and procedures in the Player Section of the Model Explorer. Remember that identifiers always have to be declared in a Declaration section, while procedures have to be declared outside of such a section. For the first level you will only need to declare variables, constraints, a mathematical program and a single procedure. A procedure with the name SolveGatheringIngredientsProblem has been predeclared in the Player Section. If you open the attribute page of this procedure, you will see comment text describing the steps you have to take and some more information.
- Run the procedure that solves your model. You can press CTRL+p to open the Progress Window to see the progress that has been made solving your model. When the solver is done, all of your declared variables have gotten their optimal values. You can open the data pages of the variables to see those values. Should the solver take a long time to solve your model you probably either made a mistake in your model or you did not formulate your model efficiently enough. In this case you can press CTRL+SHIFT+s to abort the solver and change your model.
- The easiest way to check your solution is to first assign your solution to the parameter sl::AmountOfIngredientBoughtAtMarket(i,m). If you go to the Submit Solution page your solution should be displayed in the table. Alternatively, you can input your solution values in this table by hand. You can now press the 'Check Solution' button to check your solution. Should your solution be incorrect, G-AIMMS will tell you why it is incorrect. In this case, please correct your model. If your solution is correct, congratulations on solving the first G-AIMMS: Witch Apprentice level!

Where to find help while playing
While playing G-AIMMS, there are a lot of ways to get help. See below for some of the options:
- You can always review this tutorial. It is located on the Help page.
- You can look for extra help in the Help menu, where you can find links to the complete AIMMS documentation and where you can search all of the documentation for a certain topic.
- You can click the  icon and select an object on your screen to get help on that object.
- You can type the word you want help on in a procedure, press the right mouse button while your cursor is above it and select 'Help on'.



Final Tips

Below you can find some final tips before you start playing the first level:

- Should you accidentally close all of the pages, you can open the start page by clicking the Open first page icon at the top of your screen. You can also use the File menu to do so.
- Remember to often save your project. You can do so by pressing the Save all icon at the top of your screen or by using the File menu.
- Try to keep all constraints in your model linear, which means you should avoid products of variables in constraints. Products of parameters and variables are fine. If your model is nonlinear, it is much harder to solve and solving it may lead to inexact solutions.
- If you feel like you need more knowledge on AIMMS to play the first level, you can do the tutorial for beginners first. It is linked in the Help menu in G-AIMMS under 'Additional Documentation'.
- The best way to learn how to use AIMMS, is to just try it, so go ahead and play G-AIMMS!