AIMMS Modeling Guide - Diet Problem

AIMMS 4

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
WWW: www.aimms.com

This documentation was typeset by AIMMS B.V. using LATEX and the LUCIDA font family.

Chapter 10

A Diet Problem

This chapter

This chapter introduces a simplified diet problem with an example small data set. The problem is transformed into an integer programming model with range constraints. The main purpose of the chapter is to illustrate the use of measurement units to achieve data consistency and to improve data communication. Reference is made to the special features in AIMMS that support unit analysis. One feature is the availability of unit-valued parameters. These parameters are useful when indexed identifiers contain individual entries measured in different units. Another feature is the availability of unit conventions that allow users with different backgrounds to view and enter data in their own choice of measurement units without having to change either the model or its data.

References

The example in this chapter is based on two articles that appeared in OR/MS Today ([Bo93, Er94]). Problems of this type can also be found in, for instance, [Ch83] and [Wa75].

Keywords

Integer Program, Measurement Units, Worked Example.

10.1 Example of a diet problem

Diet problems in general

The example discussed in this chapter is a McDonald's diet problem. It belongs to the class of blending problems in which various feed items (for animals) or food items (for humans) are put together to form a diet. In most applications there are weight, nutrition, and taste requirements, and clearly cost minimization is an objective.

McDonald's

The McDonald's diet problem has been used in popular literature as an example for building an introductory optimization model. The McDonald's situation is familiar, and the problem structure is simple enough for translation into a mathematical model. In addition, McDonald's provides a brochure with detailed nutritional information for every item on the menu.

Example

The example considers a small data set, which includes 9 different food types and 4 different nutrients. The 9 food types form a small but representative selection of the McDonald's menu. The 4 nutrients are calories, protein, fat, and carbohydrates. The goal is to determine a daily diet to cover the afternoon and the evening meals. Table 10.1 contains the nutritional values for each food type, nutritional requirements, food prices, and bounds on individual servings.

	Calo-ries [kcal]	**Pro-tein [gram]**	**Fat [gram]**	**Carbo-hydrates [gram]**	**max. ser-vings**	**Price [Hfl]**
Big Mac	479	25	22	44	2	5.45
Quarter Pounder	517	32.4	25	40.4	2	4.95
Vegetable Burger	341	11.7	10.6	50	2	3.95
French Fries	425	5	21	54	2	1.95
Salad	54	4	2	5	2	3.95
Lowfat Milk	120	9	4	12	2	1.75
Coca Cola	184	–	–	46	2	2.75
Big Mac Menu	1202.4	31.3	48.7	158.5	2	8.95
Quarter Pounder Menu	1240.4	38.7	51.7	154.9	2	8.95
Minimum Requirement	3000	65		375		
Maximum Allowance			117			

Table 10.1: Data for different food types

10.2 Model formulation

This section

In this section the diet problem is translated into an integer program with a single symbolic range constraint.

Verbal model statement

A verbal model statement of the problem is as follows.

Minimize: *the total cost of the menu,*
Subject to:
- *for all nutrients: the quantity of nutrient in the menu satisfies the minimum and maximum requirements,*
- *for all food types: an upper bound on the number of servings.*

Notation

The verbal model statement of the diet problem can be specified as a mathematical model using the following notation.

Indices:

f	*food types*
n	*nutrients*

Parameters:

v_{fn}	*value of nutrient n in one unit of food f*
u_f	*upper bound on number of servings of food f*
$\overline{m}_n$	*maximum allowance of nutrient n in the menu*
$\underline{m}_n$	*minimum requirement of nutrient n in the menu*
p_f	*price of one unit of food f*

Variable:

x_f	*number of servings of food f in menu*

Objective function

The objective is to minimize the cost of the menu measured in Dutch guilders (Hfl).

$$\textbf{Minimize:} \quad \sum_f p_f x_f$$

Nutrient requirements

The following equation expresses the range constraints using symbolic notation. For the McDonald's problem, inspection of the last two lines of Table 10.1 shows that the amounts of calories, protein and carbohydrates must meet minimum requirements, while the amount of fat in the diet is limited by a maximum allowance.

$$\underline{m}_n \le \sum_f v_{fn} x_f \le \overline{m}_n \quad \forall n$$

Syntax versus semantics

From a syntactical point of view the above range constraint indicates the existence of both a lower and an upper bound on the amount of each nutrient in the diet. However, it is not clear what values should be used to represent the blank entries for the parameters $\underline{m}_n$ and $\overline{m}_n$ in Table 10.1. Should they to be interpreted as zero, infinity, or minus infinity? In the AIMMS modeling language are there several ways to specify the semantics of symbolic range constraints. By restricting the domain of definition of the above nutrient requirement constraint you can avoid the generation of particular individual instances of this range constraint. By setting the default of $\underline{m}_n$ to minus infinity and the default of $\overline{m}_n$ to plus infinity, all blank entries have a meaningful interpretation. The corresponding inequalities are of course non-binding, and AIMMS will not pass these redundant inequalities to the underlying solver.

Modeling bounds on variables

Simple upper bounds on the amount of each food type f to be consumed can be expressed in the form of symbolic inequality constraints. Such translation of simple bounds into inequalities is not generally recommended as it leads to the explicit introduction (and subsequent generation) of extra constraints which are likely to be eliminated again by the underlying solution algorithm. In AIMMS you can avoid these extra symbolic constraints by specifying upper bounds as part of the declaration of the corresponding variables. In general, you should avoid formulating simple bounds on variables as explicit symbolic constraints in your model.

Model summary

The following mathematical statement summarizes the model developed in this section.

Minimize:

$$\sum_f p_f x_f$$

Subject to:

$$\underline{m}_n \leq \sum_f v_{fn} x_f \leq \overline{m}_n \qquad \forall n$$

$$x_f \in \{0 \ldots u_f\}, \quad \textit{integer} \qquad \forall f$$

Integer values

Note that the amount of food x_f is bounded from above and restricted to integer values. In the McDonald's diet problem it does not make sense to allow fractional values. In other diet applications, such as animal feed problems with bulk food items, the integer requirement can be dropped as fractional amounts of feed have a meaningful interpretation.

10.3 Quantities and units

This section

In this section the role and importance of measurement units is highlighted. Some special features in AIMMS, such as unit-valued parameters and unit conventions, are explained. The diet model is used for illustrative purposes. A more extensive discussion of quantities and units can be found in [Bi99].

Quantities

Measurement plays a central role in observations of the real world. Measurements give *quantity information* that is expressed in terms of *units*. A unit is a predefined amount of a quantity. Quantity information describes *what* is being measured, such as time, mass or length. For a given quantity it is possible to define various units for it. Take time as an example. A time period can be expressed equivalently in terms of minutes, hours or days. Table 10.2 shows some of the quantities used in the diet model.

quantity	**application in diet model**
mass	to measure the amount of protein, fat and carbohydrates, and the weight of the food types
energy	to measure the amount of calories
currency	to measure the cost

Table 10.2: Quantities used in the diet model

Units . . .

To provide a meaningful description of a quantity, it is necessary to express its measurement in terms of a well defined unit. For each quantity it is possible to define a *base unit*. For instance, in the International System of Units, the quantity 'length' has the base unit of 'meter' (denoted by [m]). From each base unit, it is also possible to define *derived units*, which are expressed in terms of the base unit by means of a linear relationship. The base units and derived units for the diet model are provided in Table 10.3.

quantity	**base unit**	**derived units**
mass	[kg]	[gram]
energy	[J]	[kJ], [kcal]
currency	[$]	[Hfl]
unitless	[-]	

Table 10.3: Units used in the diet model

. . . for consistency

When all parameters are specified in terms of their units, Aimms can check the expressions and constraints for unit consistency. For example, Aimms will not allow two parameters, one in kilograms and the other in joules, to be added together. More generally, Aimms will report an error when terms in an expression result in unit inconsistency. Therefore, if you specify the units for each parameter, variable, and constraint in the diet model, Aimms will carry out this extra check on the correctness of your model.

. . . for proper communication

Apart from unit consistency, expressing your model formulation with complete unit information will help to ensure proper communication with external data sources and other users. That is, when you want to link with other models or databases which may use different units, or, when you need to communicate with other users who may use other definitions.

Feed units and food units

Quantities used in animal feed blending problems are typically expressed in 'mass' units such as [ton] or [kg]. Therefore, a nutrient such as calories is often measured in [kcal/ton] or [kcal/kg]. This convention is not as natural for the McDonald's problem. Instead, nutrient values are specified per individual item (e.g. per Big Mac or per Coca Cola). Expressing the calories for a 'Big Mac Menu' in [kcal/gram] is not immediately useful since the weight of the Menu is generally not known. For the McDonald's problem, [kcal/item] or [kcal/BigMac] or just [kcal] is a more meaningful expression, and one could argue the plausibility of each of these choices.

Choice of food units

Throughout the remainder of this section, the last option of expressing quantities of food in terms of items will be used. This is a typical choice when dealing with discrete quantities that can be expressed in terms of the natural numbers. For instance, it is sufficient to state that the number of calories in a Big Mac is 479 [kcal], and that the number of Big Macs in the diet is at most 2.

Unit conversions

The relationship between base units and the derived units in Table 10.3 must be clearly defined. In the AIMMS modeling language, the following syntax is used to specify *unit conversions.*

```
[gram]  -> [kg]    : # -> # / 1000
[kJ]    -> [J]     : # -> # * 1000
[kcal]  -> [J]     : # -> # * 4.1868E+03
[Hfl]   -> [$]     : # -> # * exchange_rate
```

The interpretation of these conversions (indicated with an ->) is straightforward. For instance, [gram] is converted to [kg] by considering any number # measured in [gram], and dividing this number by 1000. Note that the unit conversion from [Hfl] to [$] associated with the quantity 'currency' is parameterized with an identifier 'exchange_rate'. This identifier needs to be declared as an ordinary parameter in the model.

Unit-valued parameters

When specifying a model using the symbolic indexed format, it is possible that not every element of an indexed identifier is measured in the same unit. For instance, in the diet model, the parameter v_{fn}, the value of nutrient n of food f is measured in both [gram] and [cal] as shown below. In such a situation there is a need for an indexed unit-valued parameter to complement the use of symbolic indexed identifiers in your model. In the diet model, the unit-valued parameter U_n provides this key role, and it is specified in the following table.

Nutrient	U_n
Calories	[cal]
Protein	[gram]
Fat	[gram]
Carbohydrates	[gram]

Table 10.4: Values for the unit-valued parameter U_n

With the nutrient units specified, unit consistency can be enforced by attaching units in Table 10.5 to all (symbolic) model identifiers introduced thusfar.

Total weight of diet

In order to determine the total weight of the optimal diet, the following additional parameters are declared.

Parameters:

w_f	*weight for food type f*
W	*total weight of the optimal diet*

Model Identifier	**Unit**
x_f	[-]
u_f	[-]
p_f	[Hfl]
$\overline{m}_n$	U_n
$\underline{m}_n$	U_n
v_{fn}	U_n

Table 10.5: Model identifiers and their associated units

where

$$W = \sum_f w_f x_f$$

Note that the variable x_f represents the number of items (a unitless quantity) and consequently, the units of W are the same as the units of w_f (namely [gram]).

Food Type	w_f **[gram]**	**Food Type**	w_f **[gram]**
Big Mac	200	Lowfat Milk	250
Quarter Pounder	200	Coca Cola	400
Vegetable Burger	133	Big Mac Menu	730
French Fries	130	Quarter Pounder Menu	730
Salad	127		

Table 10.6: Weight for each food type

Solution

The optimal diet satisfying the nutrient requirements for the afternoon and the evening meal costs 24.60 [Hfl] and contains one 'Vegetable Burger', one 'Coca Cola' and two 'Quarter Pounder Menus' (which include drinks). This diet contains 3006 [kcal] of energy, 89 [gram] of protein, 114 [gram] of fat, and 406 [gram] of carbohydrates. The formulation in this chapter has not included any requirements reflecting taste or minimum quantities of foods to be consumed. By adding such requirements, you can study the increase in cost and the effect on nutrient consumption.

Unit conventions

There are model-based applications which may be used by end-users from around the world. In that case, it is important that all users can work with their particular *unit convention*, and view the model data in the units associated with that convention. In the AIMMS modeling language it is possible to define one or more unit conventions, and all data transfer from and to an external medium is interpreted according to the units that are specified in the convention. By switching between unit conventions, different end-users can use their own choice of units.

Conventions applied

To illustrate the use of conventions, consider the following two convention definitions. The 'DutchUnits' convention specified by attaching the unit [Hfl] to the 'Currency' quantity and the unit [kJ] to the 'Energy' quantity. The 'AmericanUnits' convention specified by attaching the unit [$] to the 'Currency' quantity and the unit [kcal] to the 'Energy' quantity. When the 'AmericanUnits' convention is selected by a user, and given the exchange rate of 0.50, AIMMS will report the optimal cost as 12.3 [$]. In the 'DutchUnits' convention, the total amount of energy in the optimal diet will be reported as 12,586 [kJ].

10.4 Summary

In this chapter a simplified diet problem was introduced together with a small data set. The problem was transformed into an integer programming model with symbolic range constraints. The role of measurement units to obtain data consistency and proper data communication was highlighted. Special reference was made to the use of the unit-valued parameters and unit conventions available in AIMMS.

Exercises

10.1 Implement the mathematical program summarized at the end of Section 10.2 using the example data provided in Section 10.1. Verify that the solution coincides with the one presented in Section 10.3.

10.2 Introduce quantities, units and a unit parameter into your AIMMS model as suggested in Section 10.3, and include them in the graphical display of your model results.

10.3 Introduce the two unit conventions into your AIMMS model as suggested at the end of Section 10.3. Design a page in AIMMS, so that the user can select the convention of his choice and the input-output data adjusts itself accordingly.

Bibliography

[Bi99] J.J. Bisschop and M.R. Roelofs, AIMMS*, the language reference*, 1999.

[Bo93] R.A. Bosch, *Big mac attack*, OR/MS Today (1993).

[Ch83] V. Chvátal, *Linear programming*, W.H. Freeman and Company, New York, 1983.

[Er94] E. Erkut, *Big mac attack revisited*, OR/MS Today (1994).

[Wa75] H.M. Wagner, *Principles of operations research*, 2nd ed., Prentice-Hall, Englewood Cliffs, N.J., 1975.