**AIMMS Language Reference - Node and Arc Declaration**

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.

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 24

# Node and Arc Declaration

*This chapter*

This chapter discusses the special identifier types and language constructs that AIMMS offers to allow you to formulate network optimization problems in terms of nodes and arcs. In addition, it is illustrated how you can formulate an optimization problem that consists of a network combined with ordinary variables and constraints.

## 24.1 Networks

*Networks*

There are several model-based applications which contain networks and flows. Typical examples are applications for the distribution of electricity, water, materials, etc. AIMMS offers two special constructs, `Arcs` and `Nodes`, to formulate flows and flow balances as an alternative to the usual algebraic constructs. Specialized algorithms exist for pure network problems.

*Mixed formulations*

It is possible to intermingle network constructs with ordinary variables and constraints. As a result, the choice between `Arcs` and `Variables` on the one hand, and `Nodes` and `Constraints` on the other, becomes a matter of convenience. For instance, in the formulation of a flow balance at a node in the network you can refer to flows along arcs as well as to variables that represent import from outside the network. Similarly, you can formulate an ordinary capacity constraint involving both network flows and ordinary variables.

*Flow keywords*

It is assumed here that you know the basics of network flow formulations. Following are three flow-related keywords which can be used to specify a network flow model:

- `NetInflow`—the total flow into a node minus the total flow out of that node,
- `NetOutflow`—the total flow out of a node minus the total flow into that node, and
- `FlowCost`—the cost function representing the total flow cost built up from individual cost components specified for each arc.

The first two are always used in the context of a node declaration, while the third may be used for the network model declaration.

## 24.2 Node declaration and attributes

*Node attributes*

Each node in a network has a number of associated incoming and outgoing flows. Unless stated otherwise, these flows should be in balance. Based on the flows specified in the model, AIMMS will automatically generate a balancing constraint for every node. The possible attributes of a Node declaration are given in Table 24.1.

| Attribute | Value-type | See also page |
|---|---|---|
| IndexDomain | *index-domain* | 44, 212, 220 |
| Unit | *unit-valued expression* | 47, 215 |
| Text | *string* | 19, 48 |
| Comment | *comment string* | 19 |
| Definition | *expression* | 221 |
| Property | NoSave, Sos1, Sos2, Level, Bound, ShadowPrice, RightHandSideRange, ShadowPriceRange | 47, 217, 222 |

Table 24.1: Node attributes

*Nodes are like constraints*

Nodes are a special kind of constraint. Therefore, the remarks in Section 14.2 that apply to the attributes of constraints are also valid for nodes. The only difference between constraints and nodes is that in the definition attribute of a node you can use one of the keywords NetInflow and NetOutflow.

*NetInflow and NetOutflow*

The keywords NetInflow and NetOutflow denote the net input or net output flow for the node. The expressions represented by NetInflow and NetOutflow are computed by AIMMS on the basis of all arcs that depart from and arrive at the declared node. Since these keywords are opposites, you should choose the keyword that makes most sense for a particular node.

*Example*

The following two Node declarations show natural applications of the keywords NetInflow and NetOutflow.

```
Node CustomerDemandNode {
    IndexDomain  : (j in Customers, p in Products);
    Definition   : {
        NetInflow >= ProductDemanded(j,p)
    }
}
```

```
Node DepotStockSupplyNode {
    IndexDomain  : (i in Depots, p in Products);
    Definition   : {
        NetOutflow <= StockAvailable(i,p) + ProductImport(i,p)
    }
}
```

The declaration of `CustomerDemandNode(c,p)` only involves network flows, while the flow balance of `DepotStockSupplyNode(d,p)` also uses a variable `ProductImport(d,p)`.

## 24.3 `Arc` declaration and attributes

*Arc attributes*

Arcs are used to represent the possible flows between nodes in a network. From these flows, balancing constraints can be generated by AIMMS for every node in the network. The possible attributes of an arc are given in Table 24.2.

| Attribute | Value-type | See also page |
|---|---|---|
| `IndexDomain` | *index-domain* | 44 |
| `Range` | *range* | 212 |
| `Default` | *constant-expression* | 46, 214 |
| `From` | *node-reference* | |
| `FromMultiplier` | *expression* | |
| `To` | *node-reference* | |
| `ToMultiplier` | *expression* | |
| `Cost` | *expression* | |
| `Unit` | *unit-valued expression* | 215 |
| `Priority` | *expression* | 215 |
| `NonvarStatus` | *expression* | 216 |
| `RelaxStatus` | *expression* | 217 |
| `Property` | `NoSave`, *numeric-storage-property*, `Inline`, `SemiContinuous`, `ReducedCost`, `ValueRange`, `CoefficientRange` | 34, 47, 217 |
| `Text` | *string* | 19, 48 |
| `Comment` | *comment string* | 19 |

Table 24.2: `Arc` attributes

*Arcs are like variables*

Arcs play the role of variables in a network problem, but have some extra attributes compared to ordinary variables, namely the `From`, `To`, `FromMultiplier`, `ToMultiplier`, and `Cost` attributes. Arcs do not have a `Definition` attribute because they are implicitly defined by the `From` and `To` attributes.

*The From and To attributes*

For each arc, the `From` attribute is used to specify the starting node, and the `To` attribute to specify the end node. The value of both attributes must be a reference to a declared node.

*The Multiplier attributes*

With the `FromMultiplier` and `ToMultiplier` attributes you can specify whether the flow along an arc has a gain or loss factor. Their value must be an expression defined over some or all of the indices of the index domain of the arc. The result of the expression must be positive. If you do not specify a `Multiplier` attribute, AIMMS assumes a default of one. Network problems with non unit-valued `Multipliers` are called *generalized networks*.

*FromMultiplier and ToMultiplier*

The `FromMultiplier` is the conversion factor of the flow at the source node, while the `ToMultiplier` is the conversion factor at the destination node. Having both multipliers offers you the freedom to specify the network in its most natural way.

*The Cost attribute*

You can use the `Cost` attribute to specify the cost associated with the transport of one unit of flow across the arc. Its value is used in the computation of the special variable `FlowCost`, which is the accumulated cost over all arcs. In the computation of the `FlowCost` variable the component of an arc is computed as the product of the unit cost and the level value of the flow.

*Graphically illustrated*

In the presence of `FromMultiplier` and `ToMultipliers`, the drawing in Figure 24.1 illustrates

- the level value of the flow,
- its associated cost component in the predefined `FlowCost` variable, and
- the flows as they enter into the flow balances at the source and destination nodes (denoted by SBF and DBF, respectively).

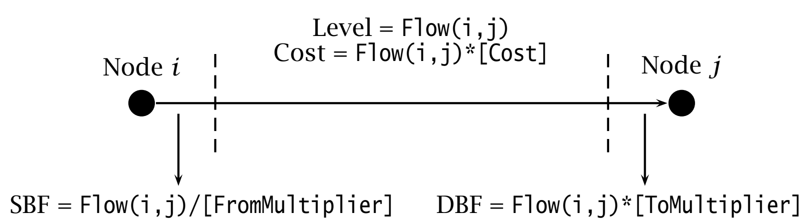


Figure 24.1: Flow levels and cost from node $i$ to node $j$

*Semi-continuous arcs*

You can only use the SemiContinuous property for arcs if you use an LP solver to find the solution. If you use the pure network solver integrated in AIMMS, AIMMS will issue an error message.

*Example*

Using the declaration of nodes from the previous section, an example of a valid arc declaration is given by

```
Arc Transport {
    IndexDomain  :  (i,j,p) | Distance(i,j);
    Range        :  nonnegative;
    From         :  DepotStockSupplyNode(i,p);
    To           :  CustomerDemandNode(j,p);
    Cost         :  UnitTransportCost(i,j);
}
```

Note that this arc declaration declares flows between nodes i and j for multiple products p.

## 24.4 Declaration of network-based mathematical programs

*The FlowCost variable*

If your model contains arcs and nodes, the special variable FlowCost can be used in the definition of the objective of your mathematical program. During the model generation phase, AIMMS will generate an expression for this variable based on the associated unit cost for each of the arcs in your mathematical program.

*Pure network models*

AIMMS will mark your mathematical program as a pure network, if the following conditions are met:

- your mathematical program consists of arcs and nodes only,
- all arcs are continuous and do not have one of the SOS or the SemiContinuous properties,
- the value of the Objective attribute equals the variable FlowCost, and
- all Multiplier attributes assume the default value of one,

For pure network models you can specify network as its Type.

*Network versus LP solver*

If your mathematical program is a pure network model, AIMMS will pass the model to a special network solver. If your mathematical program is a generalized network or a mixed network-LP problem, AIMMS will generate the constraints associated with the nodes in your network as linear constraints and use an LP solver to solve the problem. AIMMS will also use an LP solver if you have specified its type to be lp. You may assert that your mathematical program is a pure network model by specifying network as its type.

*Example*

A pure network model containing the arc and node declarations of the previous sections, but without the additional term `ProductImport(d,p)` in the node `DepotStockSupplyNode(d,p)`, is defined by the following declaration.

```
MathematicalProgram ProductFlowDecisionModel {
    Objective   : FlowCost;
    Direction   : minimize;
    Constraints : AllConstraints;
    Variables   : AllVariables;
    Type        : network;
}
```

If the arc `Transport(i,j)` declared in the previous section is the only arc, then the variable `FlowCost` can be represented by the expression

```
    sum [(i,j,p), UnitTransportCost(i,j) * Transport(i,j,p)]
```

Note that the addition of the term `ProductImport(i,p)` in `DepotStockSupplyNode(i,p)` would result in a mixed network/linear program formulation, which requires an LP solver.