Network Linear Programming

Introduction

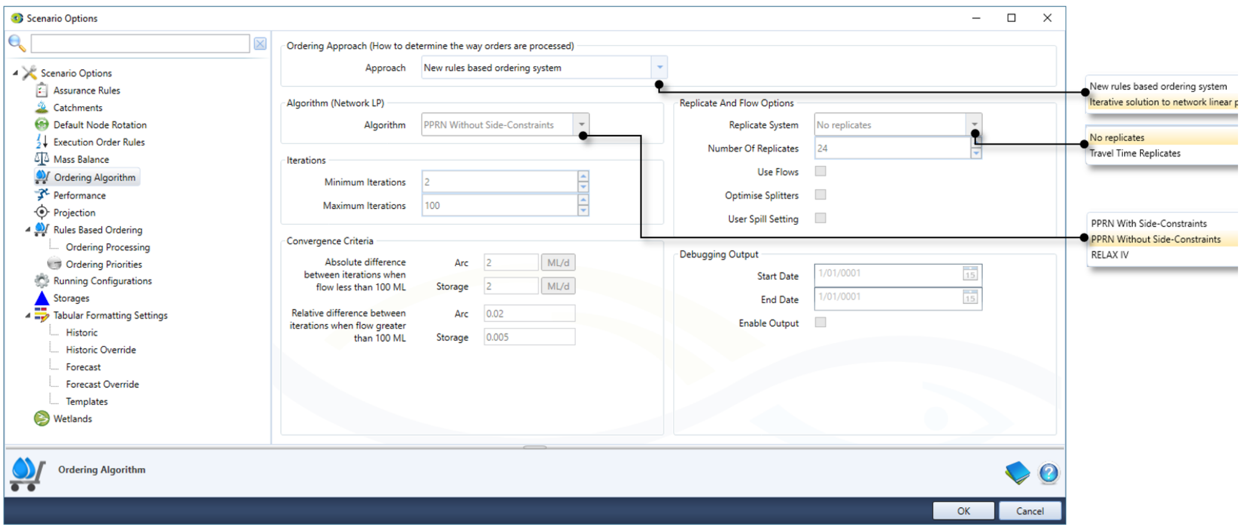
Linear programming techniques assume that the data used is perfect. In particular, when activating these features, you are assuming that the objective function and constraint coefficients are correct, rather than the best estimates available. If this assumption does not hold, the solutions found may be sub-optimal.

Within this section, the word "solver" means "the chosen linear programming optimisation algorithm".

Solvers use arc-node networks as their inputs. Source creates arc-node networks automatically for each time-step from the node-link network in the Schematic Editor. There may be differences between the solution found by the solver ("as predicted") and flows as modelled during the flow phase ("as released"). The flow distribution phase resolves these differences by considering the ordering phase as having provided a minimum target to aim for. The optimum flow for each arc is determined by the netLP so that water will flow along the least cost pathway.

To choose network linear programming as the ordering algorithm, choose ***Edit » Scenario Options*** and select ***Ordering Algorithm.*** This opens the Ordering Algorithm section of Scenario Options (Figure 1). To configure netLP, choose ***Iterative solution to network linear pro gram***, this will make the rest of the display active and allow you to configure options for netLP.

Figure 1. Enabling ordering



Ordering Approach (How to determine the way orders are processed)

Approach: New rules based ordering system

Algorithm (Network LP)

Algorithm: PPRN Without Side-Constraints

Iterations

Minimum Iterations: 2

Maximum Iterations: 100

Convergence Criteria

Absolute difference between iterations when flow less than 100 ML	Arc	2	ML/d
	Storage	2	ML/d
Relative difference between iterations when flow greater than 100 ML	Arc	0.02	
	Storage	0.005	

Replicate And Flow Options

Replicate System: No replicates

Number Of Replicates: 24

Use Flows ☐

Optimise Splitters ☐

User Spill Setting ☐

Debugging Output

Start Date: 1/01/0001

End Date: 1/01/0001

Enable Output ☐

New rules based ordering system
Iterative solution to network linear p

No replicates
Travel Time Replicates

PPRN With Side-Constraints
PPRN Without Side-Constraints
RELAX IV

About costs

The solvers use a scheme of costs to determine optimal flows. Costs can range from -1.1E13 through to +1.1E13. The larger the cost, the greater the *disincentive* for water to flow along an arc. Conversely, the smaller the cost, the greater the *incentive* for water to flow along an arc. For example, in the above list of solver priorities, satisfying evaporative and transmission losses have the lowest cost, and therefore the highest *incentive*, in any model. The question of relative costs becomes relevant when defining cost functions for storages that are being operated in harmony.

Solver priorities

The highest priority of the solver is to preserve mass balance. The remaining priorities are, in order:

- Satisfy evaporation losses in storages;
- Satisfy transmission losses;
- Satisfy minimum flow requirements;
- Satisfy consumptive demand;
- Minimise spills; and
- Where end-of-season targets are specified for storages, ensure these are met.

Linear programming solvers iterate until the solution falls within specified limits. For Source, these limits are defined in the ***Convergence Cr iteria*** fields. These limits apply, per arc and per storage, within the solver.

Choosing an optimisation algorithm

Source offers two basic linear programming algorithms: RELAX-IV and PPRN. From a modelling perspective, neither algorithm is "better" or "worse" than the other. They simply offer different features:

- RELAX-IV is generally faster than PPRN, particularly when PPRN is being run without side constraints. However, under certain circumstances, both PPRN without side constraints and RELAX-IV may become bogged down in excessive iteration;
- PPRN supports side constraints. RELAX-IV does not*. If side constraints are crucial to the correct outcome, or to avoid sub-optimal solutions, or to control excess iteration, RELAX-IV can not be used; and
- PPRN uses real numbers (at the precision of the underlying hardware) whereas RELAX-IV works with integers. Selecting the RELAX-IV algorithm implies integer conversion of the real numbers used internally by Source, during the optimisation process. Conversion implies rounding. Source scales values automatically before sending them to RELAX-IV to minimise loss of precision, and reconverts results returned by the solver to the proper range.

Any other algorithms in the list are experimental and should not be used.

If a model is sensitive to travel time, one of the ***Travel Time Replicates*** algorithms should be used.

Note: You should **not** change the solver algorithm once your model has been calibrated. Changing the solver algorithm will invalidate your calibration.

About side constraints*

Side constraints are implemented by translating between requirements set in various nodes and links in the Schematic Editor to the arcs and nodes used by the solver. In other words, you do not need to do anything to configure side constraints. Simply choosing PPRN with side constraints activates the necessary translations. For example, a Loss node creates an arc with high incentive that forces the solver to accept a particular loss.

The use of side constraints can reduce modelling time significantly. In the absence of side constraints, the solver iterates towards an optimal solution. The more iterations required per time-step, the greater the computation effort (CPU cycles = time). Side constraints can reduce the number of iterations required per time-step, sometimes to a single iteration. Each side constraint also guarantees implementation of the condition imposed by the relevant node in the Source schematic model.

Side constraints must be expressible as linear relationships for each time-step. In practice, real world phenomena are poorly modelled by straight lines and usually need curves or piecewise linear functions. Experience suggests that modellers often add numerous small line-segments to piecewise linear editors in an attempt to describe real-world behaviour accurately. The greater the number of small line-segments, the greater the risk that an artifact of the process used to derive the linear relationship for a time-step could generate an infeasible solution. Therefore, modellers should consider whether reasonable accuracy could be obtained from a smaller number of line segments. If not, linear programming solutions may not be appropriate.

Other situations where the use of side constraints are indicated include:

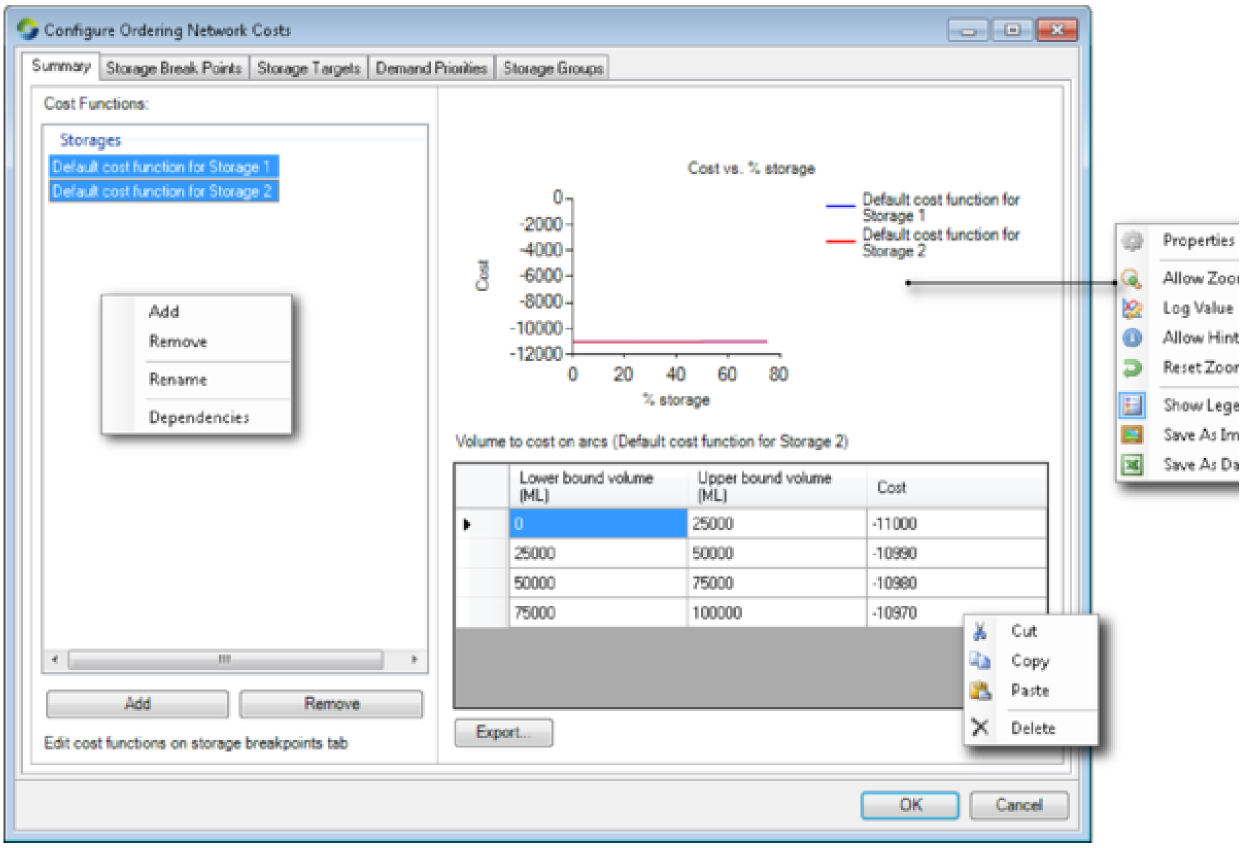
- Head versus outlet capacity relationships for storages, providing that the change in head across a single time-step does not also cause a change in the linear relationship (cross a control point in a piecewise linear function);
- Flow dependencies that cause excess iteration; and
- Circular constraints that prevent the solver from converging on an optimal solution.

*side constraints have not yet been fully implemented and tested in Source

Configuring optimisation

To configure optimised ordering, begin by choosing ***Edit » Ordering » Network Costs...*** or click ***Configure Ordering*** on the Ordering toolbar and choose ***Network Costs...***. This opens the Network costs dialog (Figure 1).

Figure 1. Network costs (summary)


Configure Ordering Network Costs
Summary
Storage Break Points
Storage Targets
Demand Priorities
Storage Groups
Cost Functions:
Storages
Default cost function for Storage 1
Default cost function for Storage 2
Add
Remove
Rename
Dependencies
Add
Remove
Edit cost functions on storage breakpoints tab
Cost vs. % storage
Cost
% storage
Default cost function for Storage 1
Default cost function for Storage 2
Properties
Allow Zoo
Log Value
Allow Hint
Reset Zoor
Show Lege
Save As Im
Save As Da
Volume to cost on arcs (Default cost function for Storage 2)
Cut
Copy
Paste
Delete
Export...
OK
Cancel

Lower bound volume (ML)	Upper bound volume (ML)	Cost
0	25000	-11000
25000	50000	-10990
50000	75000	-10980
75000	100000	-10970

Creating cost functions

By default, Source creates one default cost function for each Storage node in your model. You can add, remove or rename cost functions using a combination of the ***Add*** and ***Remove*** buttons, and the contextual menu that pops up when you right-click on the list of cost functions. A common pattern is to add cost functions that are applicable to various periods throughout the year. The finest level of granularity supported by Source is a cost function applicable to a single month.

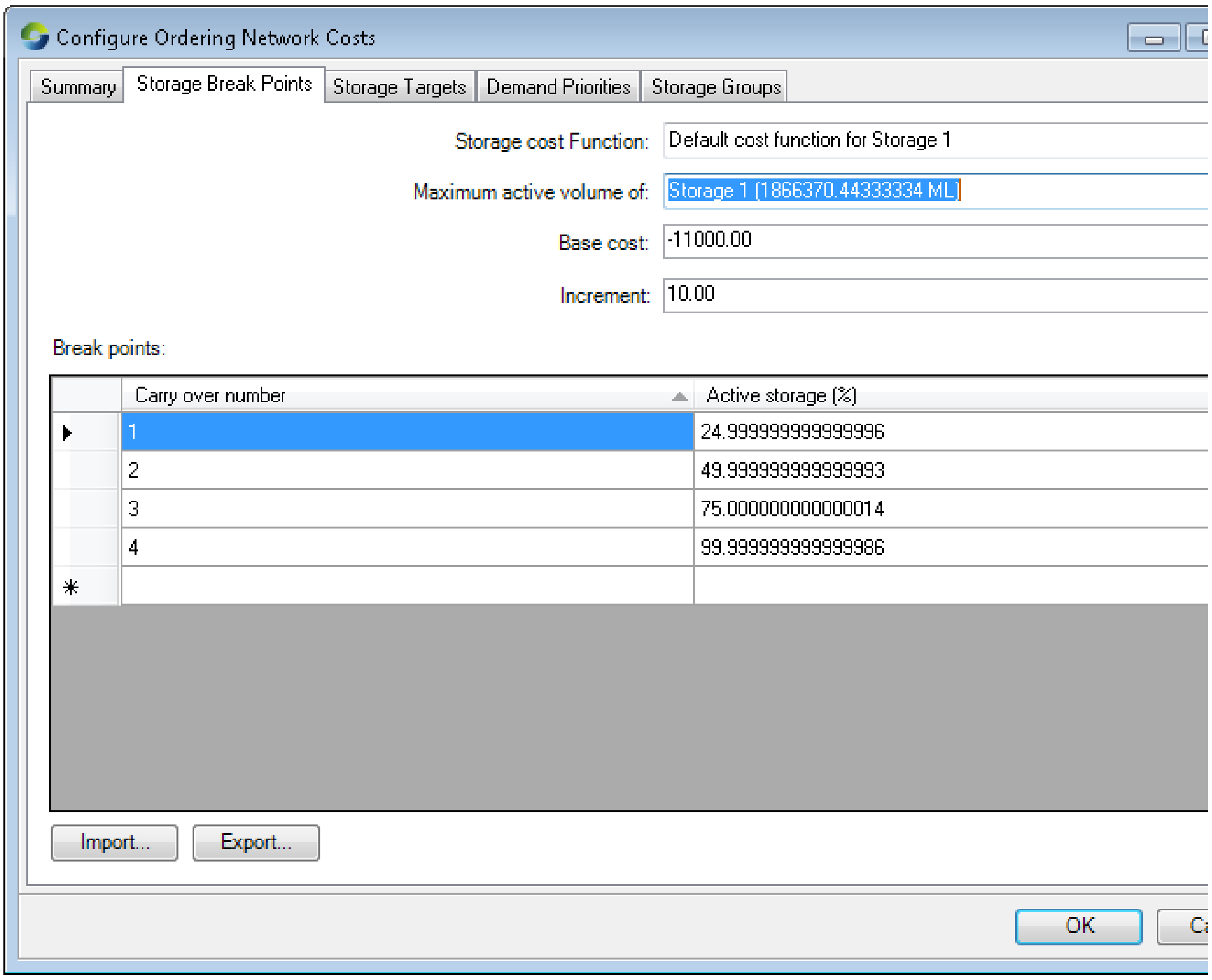
The graph on the right hand side of Figure 1 provides a graphical representation of the selected storage cost. You can also simultaneously display several functions based on your selection of storage cost functions from the list on the left hand side. The lower the cost (Y-axis) the greater the incentive to retain, or carry-over the associated storage volume (X-axis) to the next time-step. The same information is presented in the table, and can be exported to a .CSV file if required.

You can rename cost functions or display dependencies using the right-click contextual menu. A *dependency* is the association between a cost function and a storage. This can be useful if the name you choose for a cost function does not include the name of the storage for which it was defined.

Designing cost functions

Use the ***Storage Break Points*** tab (Figure 2) to design cost functions. Begin by using the ***Storage Cost Function*** pop-up menu to select a cost function that you defined in the ***Cost Functions*** list in the ***Summary*** tab. Next, select the storage that should be associated with this cost function.

Figure 2. Network Costs (Storage Break Points)



Storage cost Function: Default cost function for Storage 1

Maximum active volume of: Storage 1 (1866370.44333334 ML)

Base cost: -11000.00

Increment: 10.00

Break points:

Carry over number	Active storage (%)
1	24.9999999999996
2	49.9999999999993
3	75.0000000000014
4	99.9999999999986

Although you can define storage breakpoints for storage A in terms of storage B, you should avoid doing so because it can lead to infeasible solutions.

By default, Source provides four rows in the ***Break points*** table. Each row is associated with a percentage of the full supply storage volume, which is visible, and a cost, which is not visible. In the ***Break points*** table shown in Figure 2:

- The bottom-most 10% of the capacity of the storage is associated with a base cost of -11000.00;
- The next 40% (50%-10%) of the capacity of the storage has a cost of the base cost plus one unit of increment. Here, the increment is 10, so the cost will be -10990;
- The next 30% (80%-50%) of the capacity of the storage has a cost of -10980 (base plus two increments); and
- The remaining capacity of the storage has a cost of the base plus three increments: -10970.

A more formal specification of the cost calculation is:

Equation 1	$cost_{CarryOverNumber} = BaseCost + Increment \times (CarryOverNumber - 1)$

Recall that negative costs are incentives. Accordingly, in this table, the greatest incentive is to retain any water in the bottom-most 10% of the capacity of the storage (for carry-over to the next time-step), followed by the water in the next 40% of the capacity of the storage. By interleaving base costs and increment values, releases from multiple storages can be controlled quite precisely to maintain a desired balance.

Although Source provides four rows by default in a ***Break points*** table, you can define as many, or as few breakpoints as you need. The rules are:

- There must be at least one breakpoint and the first breakpoint must be numbered 1;

- Breakpoint numbers must increase monotonically down the table. If you edit a number so that it is more than one higher than the previous number, then close and re-open this dialog, Source will automatically insert the missing numbers and distribute the storage percentage capacity across them;
- Storage percentage numbers must increase monotonically down the table. The storage percentage associated with the first breakpoint may be zero; and
- The storage percentage in the final row must be 100%.

As you edit the breakpoint table, Source constantly checks that these conditions hold and places alert symbols next to offending values. Because of this, it may be necessary to visit each cell containing an alert symbol to force the integrity check. Alternatively, it may be easier to import more complex breakpoint tables from a .CSV file. The format of the file is shown in Table 1.

Table 1. Storage breakpoints (data file format)

Row	**Column (comma-separated)**	
	1	**2**
1	Carry over number	Active storage (%)
2..n	*n*	*point*

where:

n is an integer in the range 1..n representing the carry-over number.

point is the storage percentage (eg. “10” for 10%) when carry-over arc *n* takes effect.

To understand how cost functions work, consider Table 2 which shows the definition of cost functions for two storages that are intended to operate in parallel. The design intention is:

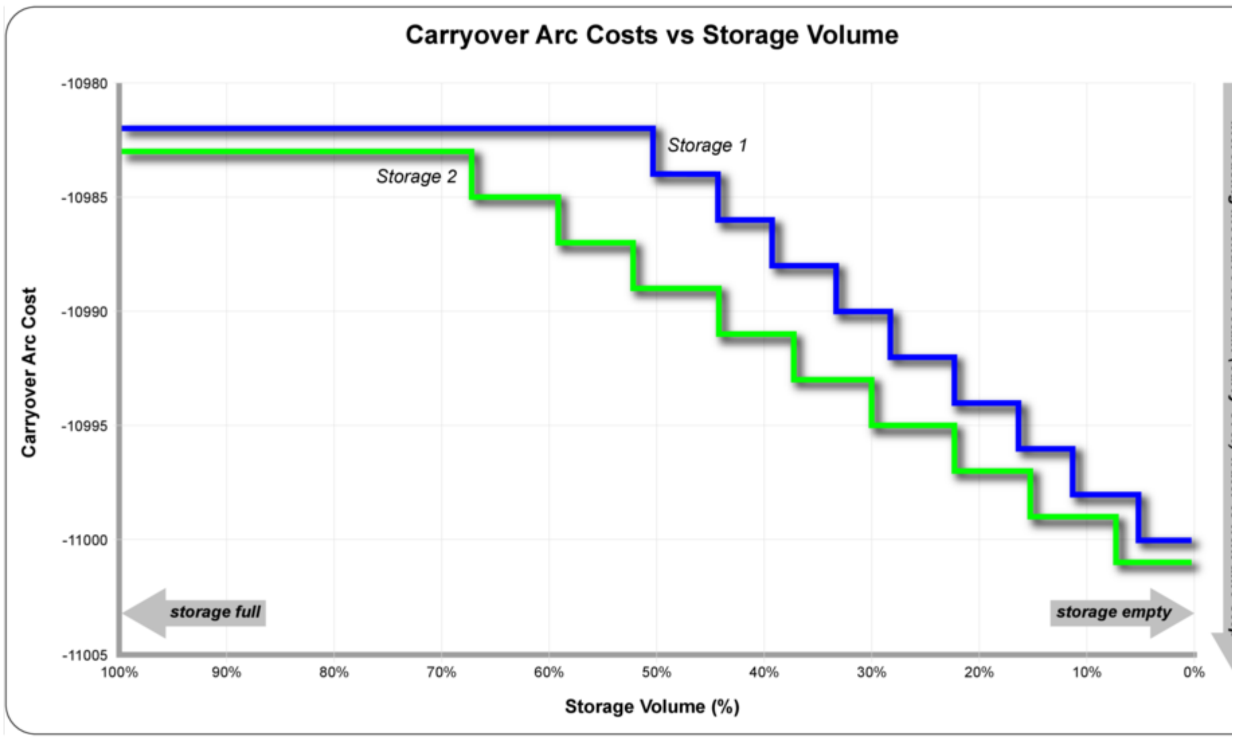
- Storage 1: the first 50% of this storage should be used first, after which its remaining capacity should be consumed in 9 equal steps of 5.6% each (50%÷9=5.6%);
- Storage 2: When storage 1 reaches 50% of its capacity, 33.3% of the capacity of storage 2 should be used, after which its remaining capacity should be consumed in 9 equal steps of 7.4% (66.6%÷9=7.4%); and
- Once the initial consumption levels have been achieved (ie. 50% of Storage 1, 33.3% of Storage 2), the two storages will be drawn down in proportion to each other.

Table 2. Storage cost functions (example)

Storage Carryover Arc	**Storage 1**		**Storage 2**	
	Base Cost	-11000	Base Cost	-11001
	Increment	2	Increment	2
	Storage %	**Arc Cost**	**Storage %**	**Arc Cost**
1	5.6	-11000	7.4	–11001
2	11.1	-10998	14.8	–10999
3	16.7	–10996	22.2	–10997
4	22.2	–10994	29.6	–10995
5	27.8	–10992	37.0	–10993
6	33.3	–10990	44.4	–10991
7	38.9	–10988	51.9	–10989
8	44.4	–10986	59.3	–10987
9	50.0	–10984	66.7	–10985
10	100.0	–10982	100.0	–10983

The result of the design intention is shown in Figure 3. Note that, in the absence of any inflows that replenish the storages, the X-axis can also be interpreted as expressing time.

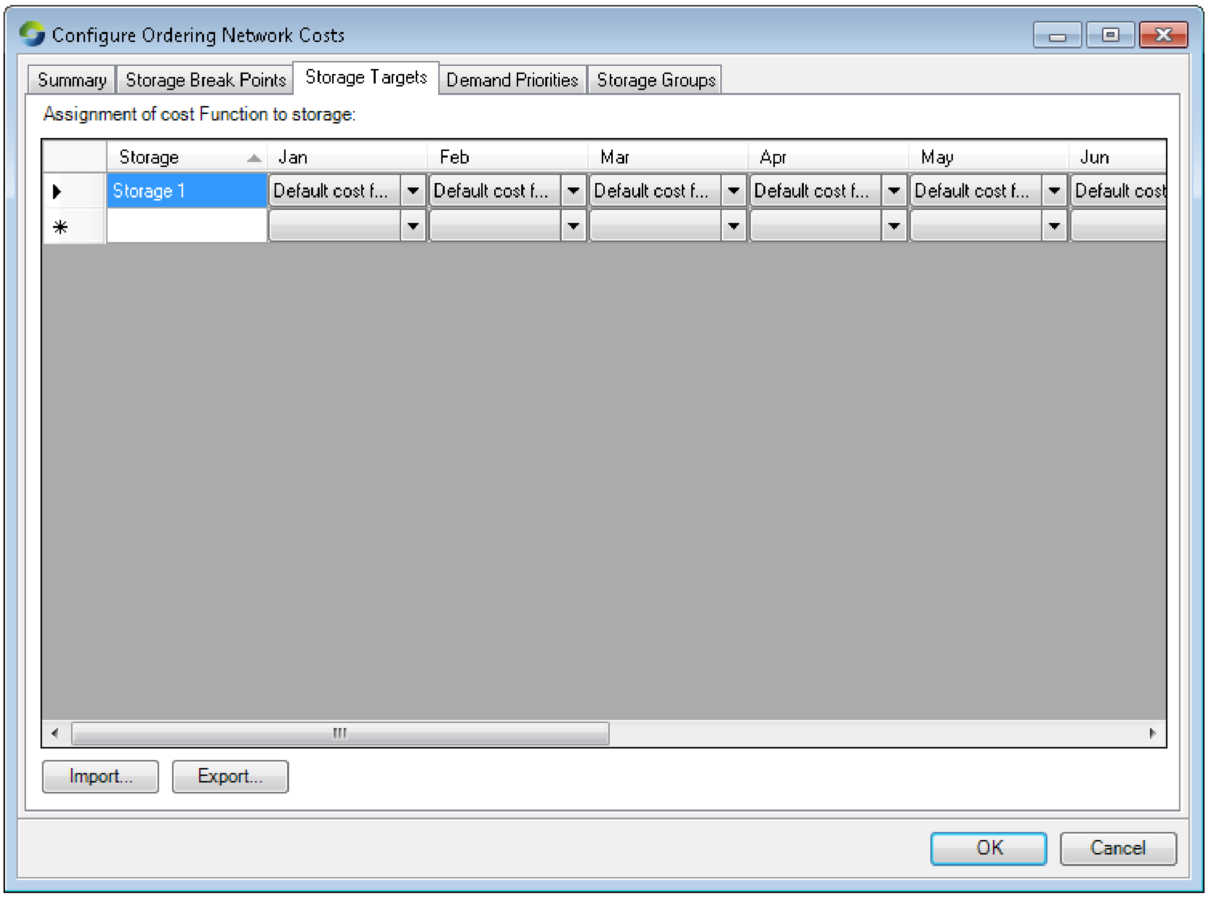
Figure 3. Carryover Arc Costs vs Storage Volume



Applying cost functions

You apply cost functions in the ***Storage Targets*** tab (Figure 4). Each storage in your model is represented by a row in this table. The remaining columns contain pop-up menus where you can connect an appropriate cost function to a selected storage for a particular month.

Figure 4. Network costs (Storage targets)



Assignment of cost Function to storage:

Storage	Jan	Feb	Mar	Apr	May	Jun
Storage 1	Default cost f...	Default cost f...	Default cost f...	Default cost f...	Default cost f...	Default cost

You can also import storage targets from a .CSV file. The format of the file is shown in Table 3. Note that the column ordering in the .CSV file does not match the display in Figure 4.

Table 3. Storage targets (data file format)

Row	Column (comma-separated)	
	1	2..13
1	Storage	*month*
2..n	*sname*	*cname*

Where

month is the first three characters of the month of the year (eg “Feb”)

sname is the name of the storage

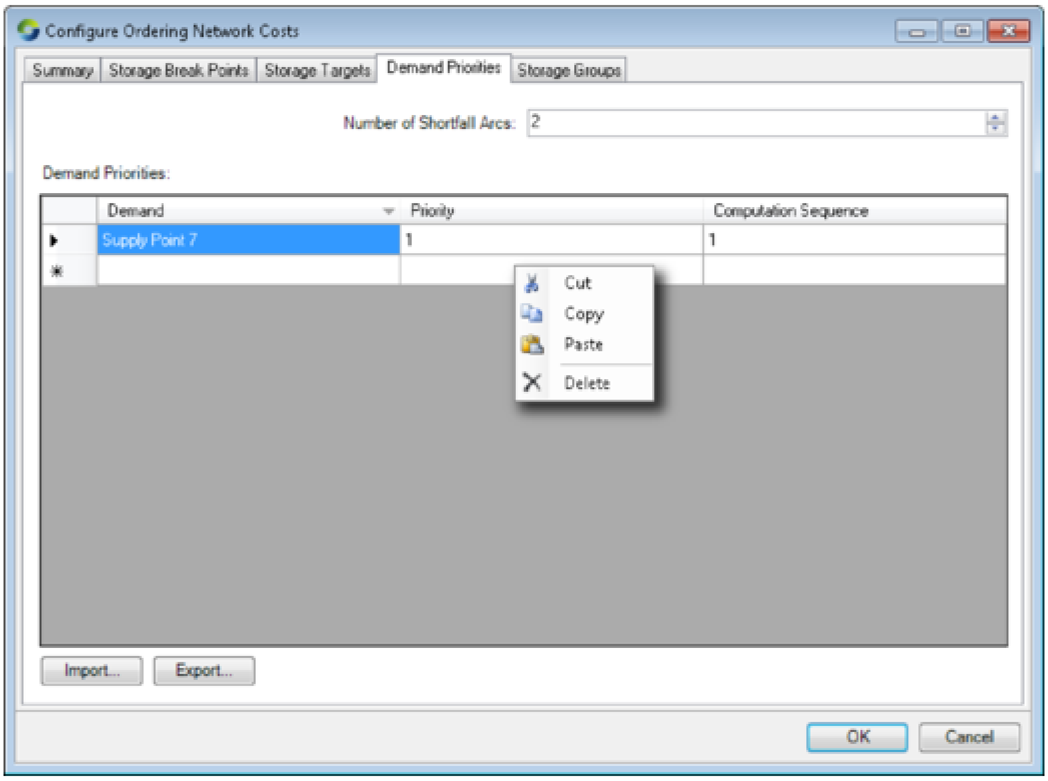
cname is the name of the storage cost function.

Controlling shortfall

A shortfall occurs when insufficient water is released to meet a demand. The arc-node model built by Source deals with this contingency by providing alternative, high-cost paths of last resort. These paths are known as shortfall arcs and their presence allows the model to satisfy mass balance.

You can specify the number of shortfall arcs that are generated for each of the demand nodes using the ***Number of Shortfall Arcs*** drop down menu (Figure 5). The default number is 4.

Figure 5. Network costs (Demand priorities)



Number of Shortfall Arcs: 2

Demand Priorities:

Demand	Priority	Computation Sequence
Supply Point 7	1	1

You can control the order in which shortfalls are satisfied. The ***Priority*** column in Figure 5 shows which demand component of the model has will receive water via its shortfall arcs.

You can import shortfall demand priority rules from a .CSV file. The format of the file is shown in Table 4.

Table 4. Demand priorities (data file format)

Row	**Column (comma-separated)**		
	1	**2**	**3**
1	Demand	Priority	Computation Sequence
2..n	*sname*	*pri*	*seq*

where:

sname is the name of the supply point

pri is the priority number

seq is the sequence number

Note: At the time of writing, it was only possible to adjust priorities or computation sequences by loading the settings from a .CSV file.

Table 5 shows the result of adjusting the priorities and computation sequences of a scenario containing two demand nodes. When priorities are equal, the first arc of the demand node with the highest computation sequence number is satisfied first, then the first arc of the demand node with the second-highest computation sequence is satisfied, and so on in round-robin fashion until either all arcs are filled or no more water remains. When priorities are unequal, all arcs of the demand node with the highest priority number are satisfied first, then the allocation moves to the demand node with the next-highest priority number, and so on until either all arcs are filled or no more water remains. You can combine priorities and computation sequences to achieve any desired outcome.

Table 5. Shortfall arc costs (example)

Scenario	**Supply point**	**Priority**	**Computation sequence**	**Shortfall arc**		**Satisfaction order**
				Number	Cost	
A	1	1	1	1	11000002	2

				2	11000004	4
				3	11000006	6
				4	11000008	8
	2	1	2	1	11000001	1
				2	11000003	3
				3	11000005	5
				4	11000007	7
B	1	1	2	1	11000001	1
				2	11000003	3
				3	11000005	5
				4	11000007	7
	2	1	1	1	11000002	2
				2	11000004	4
				3	11000006	6
				4	11000008	8
C	1	2	1	1	11000001	1
				2	11000002	2
				3	11000003	3
				4	11000004	4
	2	1	1	1	11000005	5
				2	11000006	6
				3	11000007	7
				4	11000008	8

NetLP ordering at nodes

In Source, networks are composed of arcs which have maximum capacities and costs associated with them. These costs can either be positive (disincentive) or negative (incentive). The optimum flow for each arc is determined by the netLP so that water will flow along the least cost pathway.

Storage node

While the minimum and maximum order times have to be calculated upstream of each relevant node to the relevant storages, the system also has to calculate the downstream order time which is the time frame over which the node must forecast the downstream orders for in advance.

Note: In a netLP system, refer to Operating targets when setting up a storage node.

Controlled splitter node

For the splitter node, in the netLP order phase, the main stream has two parallel arcs - a forced flow arc and the natural flow arc. The capacity of the forced flow arc in the mainstream is the opposing flow volume of the maximum effluent flow. The maximum effluent flow is dependant on the inflow to the controlled splitter node. The effluent side of the controlled splitter also has a forced and a natural flow arc, with the capacity of the forced flow arc the minimum volume of flow down the effluent side.

Rules for the flow phase handling of orders are:

- If the flow phase inflow of the node is higher than the order phase, then all excess flow will be sent down the main stream; and
- If the flow phase inflow of the node is lower than the order phase, then each side of the splitter will be apportioned water according to the proportion of the order phase main stream and effluent flows to the order phase splitter inflow.

To ensure that the above are met, you can configure each of the branches with a cost incentive. In a maximum flow constraint node that is immediately below the controlled splitter on one side of the stream, this will ensure that the flow rules are met. The maximum flow constraint should be configured with parallel arcs with either a cost for flowing water down the effluent or a cost incentive for flowing water down the main stream. The cost should be small, to ensure that the overall cost structure of the network is not impacted too much. Figure 9 shows an example of this.

Note: Ensure that the forecast supply is greater than the total order size. If this is not the case, the following error message will occur at the first node that this occurs at: ***Forecast supply greater than total order size in interpretation of network LP. Try checking inflow forecasts at inflow nodes, and that all confluence inputs are defined to be regulated.***

Troubleshooting It is possible to visualise the arc-node networks that are generated by Source using third party software. For more information please see: Troubleshooting and Debugging