VenextLP: LP_solve link for Vensim™

Overview

Some problems involve static detail complexity that can be difficult to manage in a dynamic model. This external function library links to the lp_solve library, a well-regarded open source linear/mixed integer programming solver, which is suited to such problems.

Via lp_solve, the library provides two functions: LP_ALLOC, which solves a many-to-many allocation problem with fixed channel capacity, and LP_SOLVE, which provides a generic capacity to solve linear programs.

This library is in a very preliminary state. It's already quite adequate for modest problems, but much could be done to speed it up and to develop additional functions. If you have suggestions, let us know, because we'll only pursue it if there's apparent demand.

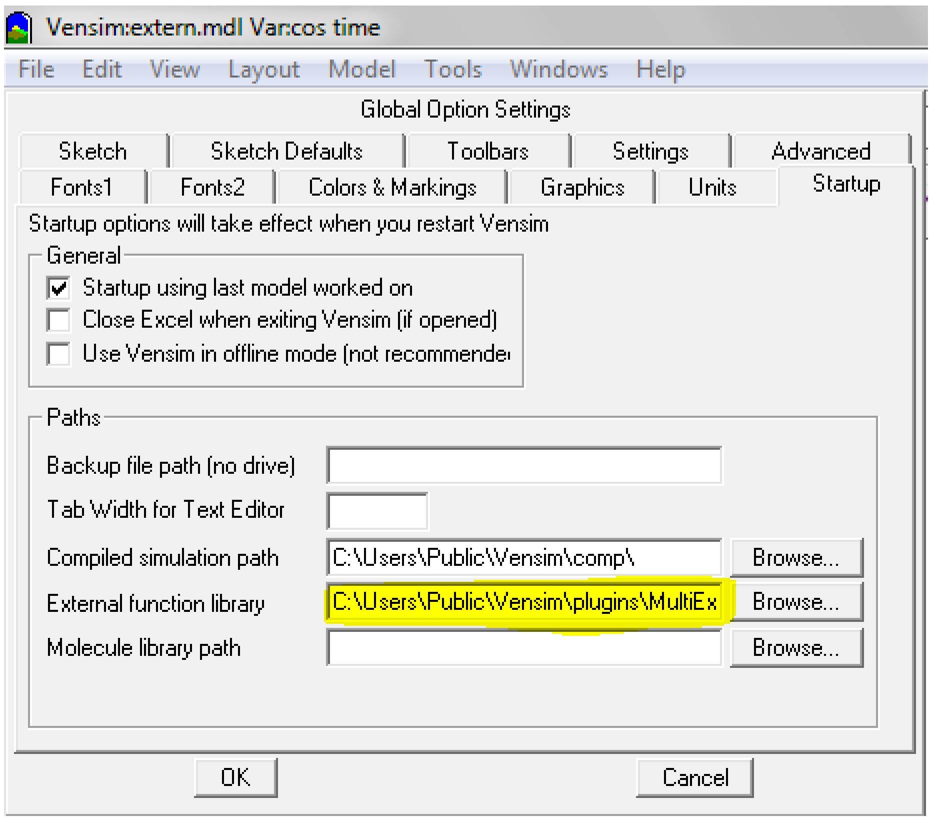
The library was developed by Tom Fiddaman at Ventana Systems, Inc.

http://ventanasystems.com/

https://www.vensim.com/

Installation

1. Place the VenextLP.dll in a convenient location (you may want to use the MultiExtLib library, also available from the Vensim Workbench site – see its instructions for detail). You can either use the binary .dll files supplied, or compile the library yourself (see next section). The latest library will generally be available at http://www.vensim.com/workbench .
2. Download LP_solve 5.5, http://sourceforge.net/projects/lpsolve/files/?source=navbar , and put its dll in a location where VenextLP can find it.
    a. The default location at present is C:\lp_solve\ lpsolve55.dll .
    b. If you want to put it elsewhere, add the location to your system path.
3. If you're not using MultiExtLib, point Vensim at the VenextLP.dll, or its double precision VenextLP_dp.dll sibling, using Tools>Options>Startup>External Function Library.


Vensim:extern.mdl Var:cos time

Global Option Settings

Startup options will take effect when you restart Vensim

General

- [✓] Startup using last model worked on
- [ ] Close Excel when exiting Vensim (if opened)
- [ ] Use Vensim in offline mode (not recommende

Paths

Backup file path (no drive)	
Tab Width for Text Editor	
Compiled simulation path	C:\Users\Public\Vensim\comp\
External function library	C:\Users\Public\Vensim\plugins\MultiEx
Molecule library path	

4. After an Open/Close model cycle, Reform & Clean, or quit/restart of Vensim, your functions will be available.

Usage

The library provides two functions, LP_ALLOC and LP_SOLVE. The sample model included with this distribution provides examples of usage for each.

Both must occur first on the left hand side of an equation. Their inputs must be ordinary subscripted variables, not expressions or numbers.

A constraint of external functions is that they do not match units. Therefore you either need to normalize your inputs, as in `relative capacity = actual capacity/reference capacity`, or use real units and ignore the errors reported. For reference, I have indicated what units conceptually represent below, following the ~. Note that, where units are indicated as a “quantity”, this might imply “quantity/time,” depending on the setting.

If the functions fail, e.g. because a problem is infeasible, they will dump the problem in matrix form to lp_solve_dump.txt for further analysis.

LP_ALLOC

```
Allocation[demander,supplier] = LP ALLOC( "supply-demand
      priority"[demander,supplier] , "supply-demand
      capacity"[demander,supplier], demand capacity[demander], supply
      capacity[supplier]  )
```

LP ALLOC uses a linear program to solve a generic allocation problem, where allocations from demanders to suppliers pass through a demander->supplier network of constraints, like shipping capacity, and each demander and supplier has individual capacity constraints. It does this by maximizing the value of the throughput, subject to each of the capacity constraints.

- Returns: the [demander,supplier] matrix of allocations from [demander] to [supplier] ~ quantity
- Supply-Demand Priority: the objective function, i.e. the net value of each allocation from a supplier to a demander ~ value/quantity
- Supply-Demand Capacity: the maximum allocation over each [demander,supplier] combination (the minimum is 0) ~ quantity
- Demand Capacity: the maximum total allocation to each demander ~ quantity
- Supply Capacity: the maximum total allocation from each supplier ~ quantity

LP ALLOC setup					
supply-demand priority	supplier:				
	s1	s2	s3		supplier capacity
demander:					
d1	1.1	1.2	1.05		3
d2	2.1	2.2	2.3		4
supplier capacity	2	3	4		
supply-demand capacity	1.1	2.2	3.3		
	2.1	2.2	2.3		

LP_SOLVE

```
Solution[variable] = LP SOLVE( objective contribution[variable] ,
      lower bounds[variable], upper bounds[variable],
      constraints[firstconstraint,variable], constraint
      values[firstconstraint], constraint types[firstconstraint]  )
```

LP SOLVE takes a linear program in matrix form and returns its solution. The lp_solve documentation has a brief description of the problem format at http://lpsolve.sourceforge.net/5.5/ .

- Returns: a vector containing the solution value for each [variable] ~ varQuantity
- Objective contribution: the contribution of each [variable] to the objective function. The total objective function is the dot product of the objective contributions with the solution. ~ value/varQuantity
- Lower bounds: the minimum value for each [variable] (normally 0) ~ varQuantity

- Upper bounds: the maximum value for each [variable]. Set to a large value if no constraint is desired. ~ varQuantity
- Constraints: the matrix of constraints on the solution, of [constraint,variable] size. Note that in actual usage, you must point to the first constraint element, i.e. [firstconstraint,variable] rather than the constraints range, because the constraints do not appear on the left hand side. ~ constrQuantity/varQuantity
- Constraint values: the vector of total constraints, one per constraint. When the solution[variable] is multiplied by the constraints[constraint,variable], the sums must conform to the constraint values[constraint]. Note that you must point to the first constraint element, i.e. [firstconstraint] rather than the [constraint] range, because the constraints do not appear on the left hand side. ~ constrQuantity
- Constraint types: an integer code indicating the type of constraint. 1 = less than or equal; 2 = greater than or equal; 3 = equality; :NA: = ignore (normally if a constraint is always ignored, you can simply omit its row from the model). Note that you must point to the first constraint element, i.e. [firstconstraint] rather than the [constraint] range, because the constraints do not appear on the left hand side. ~ dimensionless

LP SOLVE setup					
constraints	variable:			constraint types	constraint values
	cookies	burritos	milkshakes	(numeric)	
constraint:					
oven	1.1	1.2	0	<=	60
mixer	2.1	0	2.3	<=	60
lower bound:	0	0	0		
upper bound:	100	20	50		
objective function:	1	5	2		

Adapting & Improving the Library

Source code for the library is provided under the Apache 2.0 license, a permissive Open Source license, so that you may adapt it to your own purposes or improve its features. We welcome submissions. If you plan to do some work on the library that you would like to share, let us know in advance and we can set up a source code repository.

The source code provided includes a Microsoft Visual Studio 2010 project and a test model.

http://www.apache.org/licenses/LICENSE-2.0

Support

Strictly speaking, this is unsupported software, and Ventana neither assumes liability nor provides any warranty or support. However, we are happy to entertain questions and suggestions in the Vensim forum, http://www.ventanasystems.co.uk/forum/viewforum.php?f=2 , and we will endeavor to be as helpful as we can.

If you need support or extensions for mission critical work, Ventana or Ventana UK may be able to undertake that as a consulting project.

License