**MOSDEX Module Example**

Dr. Jeremy A. Bloom  
[jeremyblmca@gmail.com](mailto:jeremyblmca@gmail.com)  
February 27, 2019

1. **The Multinet Example**

This example is adapted from the Net1 example, which is a single commodity transshipment model. The extension represents a problem in which multiple products flow through the network and share the capacities of the links.

Multinet-v1 shows the extensive form of the model, that is, an ordinary linear program with all constraints and variables represented. The key substantive changes are as follows:

1. addition of a new index set, products, to differentiate among the flows;
2. expansion of the ship variables to index the combination (cross join) of links and products,
3. addition of a set of capacity constraints that limit the total flow along each link, replacing the link capacity bounds on the ship variables;
4. definition of coefficients for the ship variables in the capacity constraints to represent the sums of the ship variables over the products dimension.

Several MOSDEX syntax elements have been modified as follows:

1. Combined definition of table schema and content;
2. Use of SQL statements in defining data entries in bounds on variables and constraints and in coefficients, allowing SQL anywhere a table reference would be legal
3. Extension of the SQL syntax to support a MATCH operator in the WHERE clause, as a compact shorthand for chained field equality terms.

The multinet problem has a natural modular structure in that each product has the same transshipment structure, which results from the shipment balance requirement at each city. These modules interact through the capacity constraint on each link. (One is tempted to use the terms *subproblem* and *master problem* but we envision applications of modularity to be broader than decomposition algorithms). The problem factors into two modules. The first is the single commodity transshipment model, transshipment-v2 (the v2 is to maintain consistency with the versioning of multinet, as there is no v1 of the transshipment model). This module is almost identical to the original net1 example, except for the following:

1. MOSDEX syntax has been updated as discussed above;
2. The data tables each have a schema but no content, which will be passed from the parent;
3. The capacity bounds on the ship variables have been relaxed since they will be enforced by the parent.
4. Note that there are no references either to the parent or to the products index set which is part of the parent, and with content supplied for the data tables and bounds, it could be optimized independently by a solver.

The multinet-v2 module makes use of the transshipment-v2 module to specify the multi-commodity flow model succinctly. It has the following features:

1. Defines the products index set;
2. Expands the supplies input data table and the shipments output data table to include a product index;
3. Calls the transshipment model as a module, as discussed in detail below;
4. Defines the capacity constraint and the totalCost decision expression;
5. Defines the coefficients for these two objects.

The syntax of the call to the transshipment model is exhibited in the Modules object. The name definition includes introducing the product index, which is a member of the index set products. Since we have chosen to house the transshipment model in a separate file, the path to that file is referenced; alternatively, we could have incorporated the transshipment model text within this part of the multinet-v2 file. Then the linkages between the two modules are defined in the EXPORT\_TO and IMPORT\_FROM arrays. The former lists the tables transferred from the multinet model to the transshipment model; note the use of an SQL query to select those rows of the supplies table that match the product index, which itself is not exported. The latter lists the tables that are returned from the transshipment model to the multinet model; since these tables do not have a product index in the transshipment model, that index is implicitly added when the tables are returned. In this example, all of the tables in the transshipment model have been made visible in the multinet model; however, that is not generally the case, and there are many situations, particularly when the solver uses a decomposition algorithm, in which some of the tables could be private within the subproblem module. Note that visibility is controlled by the caller module, rather than the more usual case where the called module controls visibility. For optimization at least, having the caller control visibility has a couple of advantages: it frees the called module from having any dependence on the caller, and it enables using alternative solver algorithms with minimal changes to the model specification. For instance, the multinet model as specified could be solved in extensive form, since all of the transshipment modelling objects are visible; however, one could use a Dantzig-Wolfe decomposition algorithm by having the multinet master problem import only the ship variables (although it would have to export dual information on the capacity constraints to the transshipment subproblem).