SAS/OR® Optimization Procedures, with Applications to the Oil Industry

Marc-david Cohen and Trevor D. Kearney, SAS Institute Inc., Cary, NC

ABSTRACT

This paper explores the use of the optimization procedures in SAS/OR® software; (the LP, NETFLOW, and NLP procedures), with applications to the oil industry. The applications include refinery models, and production, inventory, and distribution models. This paper focuses on applying network modeling techniques and choosing an optimizer for solving these models.

INTRODUCTION

The oil and gas industry pioneered the application of mathematical programming techniques for modeling refinery planning (refer to Aronofsky et al. [1978]). These models fall into two general categories: strategic planning models and operations models. Strategic planning models range from those that seek to identify oil drilling sites (Hosseini [1986]) using techniques in decision analysis to those that seek sound capital investment strategies using linear and integer programming techniques (Buchanan et al. [1990] and Murtagh [1981]). Operations models tend to focus on the refinery using accounting and data processing (Bayard [1993]); linear programming (Buchanan [1990]), and nonlinear programming (Dewitt et al. [1989]). The SAS® System supports software components for solving many of these models. This paper will highlight some of these techniques. Although this paper focuses on processes in the oil and gas industry, many of the models can be applied to other process manufacturing: instead of refineries and oil, the models could focus on mills and grain, breweries and beer, or refineries and sugar.

Solution of these models relies on optimizing a linear or nonlinear function over a linear or nonlinear region. In the SAS System there are four optimization procedures, (all in the SAS/OR software product): PROC LP, PROC NETFLOW, PROC NLP, and PROC DTREE.

- PROC LP solves the linear and mixed-integer programming problem. It can evaluate the sensitivity of the solution to changes in resources and changes in returns. In addition to oil and gas applications, linear and mixed-integer programming have been used for many other applications.
- PROC NETFLOW solves various network flow problems. These problems are linear programming problems with embedded networks. By exploiting the structure of the embedded network, specialized algorithms (and those used by PROC NETFLOW) can solve these problems much faster than if they were solved as general linear programs. An important capability in PROC NETFLOW is its ability to solve problems that have side constraints and nonarc variables. Side constraints and nonarc variables are additional conditions and variables that are needed for modeling but do not represent typical network components such as flow conservation conditions and arcs. It is important to note that a large proportion of linear programming problems have embedded networks and could use PROC NETFLOW to exploit this structure.

  PROC NETFLOW can use Optimization Subroutine Library (OSL), an IBM software product, to perform some or all the optimization.
- PROC NLP offers a set of optimization techniques for minimizing or maximizing a continuous nonlinear function with boundary and nonlinear constraints. Special, more efficient algorithms are provided for nonlinear least-squares problems, and quadratic programs.

  The NLP procedure is new for Release 6.10 of SAS software.
- PROC DTREE solves decision analysis problems. A decision problem models a sequence of events that includes those at which decisions must be made and those at which some random outcome occurs. The procedure identifies optimal decision strategies and provides a means of accessing the value of information.

SAS/OR is documented in *SAS/OR User's Guide* [1993].

In this paper, we first discuss the use of networks for the modeling applications. A multi-refinery, multi-period model using the network paradigm is then presented. Next, we describe the refinery model in greater detail and use that description for an example refinery model. We show how that model is solved by PROC NETFLOW. We then discuss possible extensions appropriate for long range planning models. The capabilities of PROC LP, particularly integer programming, are described in the final section.

NETWORKS AS A MODELING TOOL

Networks can be a very useful tool for modeling. Typically, they represent a process that changes material as it flows through the network. One such example is the changes from crude oil to finished products that occurs in a refinery. A network consists of:

- *nodes*, which represent supply, demand, or transshipment points, and model components such as warehouses, tank farms, factories, refineries, and customers. At a supply node, material enters the network; at a demand node, material leaves the network; and at a transshipment node, the total flow of material into the network equals the total flow of supply out of the node. This last restriction is known as *conservation of flow*.
- *arcs* carry material between nodes. Each arc has a unique node at which it originates and another unique node at which it terminates. Arcs have a per unit traversal cost, which may be zero or negative (if the cost is really a profit), a capacity, (the largest amount of flow that arc can tolerate), and a minimum flow bound (useful to force some minimum quantity through that arc).

Many comprehensive models can be built using nodes and arcs; however, it is often necessary to specify additional constraints on

the flow of material through a network. These additional constraints, called *side constraints*, enable more general problems to be tackled. It would be impossible to formulate as realistic a refinery model without them. Side constraints are used for a wide variety of reasons in network models. They can describe any linear relationship among the flow in the arcs. For example, they are needed to model the situation when the amount of finished products flowing away from a node is proportion to the amount of raw material flowing toward the node. More detail on those constraints relevant to a refinery model are outlined in the section **SIDE CONSTRAINTS - More Detail**.

Network models as a representation of the movement of material or commodities are a natural choice in many situations. Since they can be depicted pictorially, they clarify the model's structure, so they are useful for explaining what the model does and how solutions should be interpreted.

Solving a network model consists of identifying the minimum cost flow through the network from the supply nodes to the demand nodes that obeys the conservation of flow at nodes, the side constraints, and the capacities on arcs. There are two procedures in SAS software for solving these problems: PROC LP and PROC NETFLOW. PROC NETFLOW is geared more toward solving network models than PROC LP. However, there are situations where PROC LP is the appropriate choice.

Which procedure to use to solve a model depends on the structure of the model. For both procedures, the model is described in SAS data sets; however, PROC NETFLOW supports a model description that is oriented more toward networks, while PROC LP supports a more general model description. For example, conservation of flow constraints must be modeled explicitly when using PROC LP, but these constraints are included implicitly by PROC NETFLOW. More importantly, the algorithms in PROC NETFLOW exploit the network structure in the model and can solve the problem much more quickly than PROC LP. Thus, if the model is a pure network (that is, it has no side constraints), or there are not many side constraints, PROC NETFLOW is the appropriate choice. However, if the model has a large number of side constraints relative to the size of the network constraints, then PROC LP is the appropriate choice. It is difficult to quantify the ratio of side constraints to network at which you should switch from one procedure to the other. However, if the model has restrictions that require integer restrictions or needs elaborate sensitivity analysis, PROC LP may be the appropriate choice. It is also worth noting that frequently an underlying network structure can be found in large or complicated linear programming models. In these cases, PROC NETFLOW should be considered as appropriate.

A MULTI-REFINERY, MULTI-PERIOD MODEL - The Big Picture

To make this discussion of network models a little more concrete, we consider a model of an oil refinery. Figure 1 is a diagram of a network model of a multi-refinery, multi-period model. It shows the flow of crude oil and products through two refineries for three months. There are several things to note about this model. First, crude oil can be transferred from one refinery to another or stored in inventory at the same refinery. Next, refined products are produced by each refinery each month, after which they may be stored in inventory at the refinery or transferred to another refinery. Last, horizontal arcs represent crude oil or products held in inventory, while other arcs represent the movement from one refinery to another. Since each arc can have its own costs and capacities, operational aspects such as operating costs, plant capacities, efficiencies, and capabilities, can vary with refinery and period.

Multi-period models such as these can be used to analyze refinery operations when factors such as crude oil purchase costs and availability, refined product demand and sales price, or the operating and inventory parameters of a refinery, change over time. Multi-period models also help determine the timing, sizing, and phasing of investments for construction of new refineries or new facilities, or investments to increase the capacity or efficiency of existing plants.

This model can be extended in several ways. There could be crude oil and refined products in inventory at the beginning and end of the planning horizon. There could be additional refineries and a longer planning horizon. There could be multiple sources and types of crude oil and multiple products. These types of extensions are incorporated by adding more nodes, similar to the ''crude'' and ''products'' nodes, and appropriate arcs to connect them to the model.

A REFINERY MODEL - Detail

In the multi-refinery multi-period model, in each month, each refinery is represented by a single node. To capture detail and make the model more realistic, it is necessary to model some of the operations that occur at the refinery level.

Figure 2 is a model of a typical refinery. The material that flows through the refinery represents different commodities at different points in the network. At first the flow material is crude oil. The nodes ''crudeA bought'' and ''crudeB bought'' are source nodes for two different types of crude oil. They have purchase prices, origins, availability, and transportation costs, which vary at each refinery, for each period.

The crude oil first passes through a Crude Unit, where it is fractionated or distilled (separated according to boiling point ranges). The amount of each intermediate product produced is proportional to the amount of feed streams used as input. Some intermediate products may be in stock or obtained from another location. Some intermediate products may be sent directly to final blending pools or held in stock. The remaining intermediate streams pass through additional processing units, where their chemical compositions are further altered. The products from each unit are either sent to another unit or are blended with other products with desired properties. Units that process intermediate products after crude oil distillation are called ''downstream units''. The desired final refined products are prepared by blending various output streams from crude distillation and primarily from other downstream processing units.

Last, flow represents finished products such as gasoline or jet fuel. The network model shows the amount and location of demand for each finished product. The quantity of final products demanded may be given, and a solution must satisfy the specified demand. Otherwise, the quantity of final products is obtained from the solution to the model. The model is used as a strategic tool to identify the best utilization of refineries given quantities of crude oil. In this case, the supply and cost of crude and the sales price of refined products drives the quantities of finished products produced.

In addition, blended finished products from each refinery are distributed to one or more regional terminals and to export markets. Transportation costs from refinery to terminal are based on specific modes of transportation (for example, truck, rail, ship, pipeline) that are available for use between each refinery and each terminal. Costs and returns may be affected by the type of packaging.

There are a number of details that are captured by the network representation of the refinery; the treatment of inventory is one of them. Notice that finished products may be held as inventory. Because inventory must traverse an arc, the model captures inventory holding costs and inventory capacity. Arcs that are horizontal in Figure 1 and Figure 2 represent holding inventory between periods since they convey commodities through time. In Figure 2 the arc between nodes ''crudeA Jan'' to ''crudeA Feb'' represents the amount of crude A that was in inventory at the beginning of January, or purchased in that month, and not used during the period. The node

"crudeA Jan" could be either a supply node, with supply being the quantity on hand, or an intermediate node capturing flow from a previous month's excess supply of type A crude. Similarly, excess supply in February could be made available to the March submodel. These extensions can be modeled by adding the appropriate arcs and nodes. There are several other inventory arcs in Figure 2; "crudeB Jan" to "crudeB Feb", "IntermedA Jan" to "IntermedA Feb", "IntermedB Jan" to "IntermedB Feb", and "productA Jan" to "productA Feb".

Another detail concerns modeling node throughput. For unit A, the amount of throughput is the flow through the arc between nodes U_A and UnitA. The limit on the amount of throughput that unit A can handle is specified as the capacity of this arc. The per unit operating cost is modeled by the cost of flow through that arc. Although there are other ways of modeling costs on node throughput by using arc and nonarc variables in a side constraint, it is usually more efficient to model this explicitly in the network.

The model also considers imported finished products. They are included at refineries that have port facilities capable of handling such products and are distributed to the terminals assigned to each refinery. Also notice the arc from "prodA import" to "productA Jan". This models the purchasing or importing of product A into the refinery.

Buchanan et al. [1990] suggest adding additional supply nodes, called *shortfalls*, connected to every refinery node and finished product node in every time period. These nodes supply products at a very high price, if it is necessary to do so to meet demands. Technically, the primary purpose of the shortfall supply nodes is to prevent infeasibility and, as a by-product, to pinpoint the nature and the magnitude of product shortages for planning purposes. When legitimate shortfalls do occur, managerial judgement is required to decide how to deal with them; for example, they might expand imports, or reduce demand. The existence of the expensive shortfalls in the model might force the refineries to use more expensive crude oils or adopt a flow pattern through the refineries so that, as much as possible, demands are met without using shortfalls.

There may be differences among the refineries. For example, all types of crude oil may not be available at a refinery, and not all refined products may be made at all refineries. To generate a particular refinery model, it might be convenient to start by replicating a "general" refinery model, then customize it by deleting inapplicable nodes and arcs.

SIDE CONSTRAINTS - More Detail

There are many types of side constraints that are needed to model a refinery at a level of detail that is useful for planning and managing operations. Three that we will discuss are *proportionality, blending*, and *coupling constraints*.

Proportionality Constraints

Side constraints are necessary when relationships between flows in arcs cannot be expressed by node flow conservation constraints implicit in network models. Side constraints are an important component of refinery models because you can model proportional constraints when the flow through one arc is proportional to the flow in another arc. This occurs in yield or blending problems where the relationship between two variables or the relationship between the flows in two arcs must maintain a specific ratio or range of ratios. For example, the flow through the arc from the node "P_A" to "productA" is the amount of product A that is manufactured by blending intermediate product A and intermediate product C. If every ton of product A contains 0.3 tons of intermediate product A and 0.7 tons of intermediate product C, then specify these side constraints:

```
0.3 P_A->productA - 1.0 IntermedA->P_A = 0
0.7 P_A->productA - 1.0 IntermedC->P_A = 0
```

You do not need to specify the second constraint: if the first constraint is true, the second must also be true, as flow is conserved at the transhipment node "P_A".

Many side constraints in the refinery model are necessary to capture blending recipes. Every ton of intermediate product B processed by the Unit B produces 0.4 tons of intermediate product C and 0.6 tons of intermediate product D. That is modeled by these constraints:

```
0.4 IntermedB->U_B - 1.0 UnitB->IntermedC = 0.0
0.2 IntermedB->U_B - 1.0 UnitB->IntermedD = 0.0
```

More sets of constraints might be necessary to relate input and output streams of other downstream units.

A set of proportional constraints are required for the Crude Unit. One ton of crude A produces 0.35 tons of intermediate product A. One ton of crude B produces 0.45 tons of intermediate product A. To model that, use these constraints:

```
0.35 crudeA Jan->CrudeUnit + 0.35 crudeA shortfall->CrudeUnit
- CrudeUnit->IntermedA = 0.
0.45 crudeB Jan->CrudeUnit + 0.45 crudeB shortfall->CrudeUnit
- CrudeUnit->IntermedA = 0.
```

There would be similar constraints for the other output streams of the Crude Unit.

Blending Constraints

Another type of side constraint ensures that a product has some desirable characteristic (for example, gasoline has some stipulated octane number, or permissible sulphur and lead content, viscosity, or volatility). These product characteristics are determined by what streams are blended, the blending recipes used to manufacture the product, and the properties of the individual feed streams.

Coupling Constraints

Side constraints are also useful to model limitations on some shared resource, or limits on overall production or demand in multi-commodity, multi-divisional or multi-period problems. When each commodity, division, or period has an independent network coupled by a shared resource, then a side constraint can be used to model the limitations represented by the limited resource. These are called coupling constraints. For example, when side constraints combine the outputs of subdivisions (either commodities, outputs in distinct time periods, or different process streams) to meet overall demands or to limit overall production or expenditures, they are a coupling constraint.

For example, a product is used to fill 60 and 210 liter drums. These drums occupy different volumes, $vol60$ and $vol210$, which are constants. If N60 is an arc representing the number of 60 liter drums sent from a refinery to a customer, and N210 is an arc representing the number of 210 liter drums sent from the same refinery to the same customer, the following side constraint is necessary:

```
vol60 . N60 + vol210 . N210 <= truckvol
```

where $truckvol$ is the total volume of drums that can be carried by the truck dispatched from the refinery to that customer. Similar constraints are necessary if the drums are held in inventory, and a tank or warehouse capacity cannot be exceeded.

Another example of a side constraint as a coupling constraint occurs with budgetary constraints. Although the refinery model is based on a monthly period, you might want to include annual budget constraints in the model. In this case, the side constraint ties together the monthly components of the annual budget.

All refineries compete for the same resources, such as raw materials (crude oil) and capital expenditure for expansion. Different

refineries might have differing efficiencies or other advantages. These situations are modeled by coupling side constraints.

SOLVING THE MODEL

Solving the model means identifying the flow over each arc that minimizes the total cost of the model's flow, so that the demand is met without exceeding the supply, the flow through each arc is between the arc's lower flow bound and capacity, while satisfying side constraints.

The objective function to be minimized is the discounted net cost over the multi-period planning horizon. This net cost is the sum of:

- the total cost of crude oil
- the total costs of carrying inventories
- the total operating costs of the refineries
- the total cost of importing refined products (including transportation costs from a foreign port)
- the total costs of transporting refined products from refineries and ports to terminals and other refineries
- the gross revenues from exporting refined products less transportation costs from domestic refineries to markets abroad

The solution gives the optimal flow of products through the network. This tells you how much crude to refine into what products, at which refineries, and in which periods. Of course, the solution depends on the supplies of raw materials, the projected demands of finished products, and the costs, returns, and capacities. In addition, there are various other ways to exploit the model. For example, suppose that you specify that nodes supply flow, but you do not specify how much. The solution then tells you the amounts and types of crude oil to buy so that the demand for products is met at minimum cost. As an alternative, you can specify that the demand nodes demand products but not specify how much. The solution then tells you the amount of finished product to produce when processing as much crude as possible. While this type of analysis is possible with a general linear programming formulation of the network model and solution using PROC LP, it is much easier to achieve with a network formulation and solution with PROC NETFLOW.

Other questions that can be answered include:

- How can a change in the price or quality of a given crude affect the optimal purchasing mix of crudes?
- Should a new refinery should be built? What is the best size of such a refinery? What types of units should be included?
- What is the optimal allocation of production to terminals and export markets, and how would this be affected by changes in demand?
- Where are potential bottlenecks to meeting future demand located in the refineries, and what are the most cost-effective ways to remove them?
- What changes are brought about because of changing environmental regulations or changing automobile engine requirements.

EXAMPLE

The example in Palmer et al. [1984, pp. 290-296] will be used to demonstrate some modeling techniques, data generation, and optimization using PROC NETFLOW.

This model's refinery processes three types of crude oils: Arabian Light, Arabian Heavy, and Brega. The amount of each type of crude oil available is 110, 165, and 80 thousand tons, respectively. The Crude Unit produces three downstream products: Light Virgin Naptha, Intermediate Virgin Naptha, Virgin Heating Oil, Vacuum Distillate, and Vacuum Residue. In addition, there is some oil lost in this step of the process. Additional products produced by the Platform, the Low Severity Catalytic Cracker, and the High Severity Catalytic Cracker are blended to make Gasoline, Jet Fuel, Heating Oil, and Fuel Oil. The demand for these are 40, 20, 50, and 145 thousand tons, respectively. However, if necessary, Fuel Oil may be piped in from another refinery to satisfy some Fuel Oil demand.

Figure 3 illustrates the flow through the refinery. Note the similarities between Figure 2 and Figure 3.

SUPPLY AND DEMAND

In this example, there are upper limits on the availablilty of the Arabian Light, Arabian Heavy, and Brega crude oils, and there is known demand for refined products Gasoline, Jet Fuel, Heating Oil and Fuel Oil. If these were the only supplies and demands of flow in the model, the model would be simple. However, the situation is complicated by the fact that Fuel Oil can be bought from another refinery. In Figure 3, the supply node refinry2 is the source of this additional Fuel Oil.

The loss from the Crude Unit is another complicating factor. In effect, this loss is a demand for flow that is dependent on the amounts of each type of crude oil used. Since these amounts are not known beforehand, they must be accounted for in the model. A special modeling technique is used to handle cases like this, where total supply and total demand are not known. Construct a model as in Figure 4 having two new nodes, SuperSource and Supersink, and a new set of arcs connecting these to the network. The first set of arcs is directed from SuperSource to the source nodes and should have as capacity the supply of the source node; and the second set of arcs is directed from the demand nodes to the SuperSink node and should have as lower flow bound the demand at the demand nodes. In this example, these are the arcs from SuperSource to the nodes ArabLite, ArabHvy, Brega, and refinry2, and the arcs from nodes Gasoline, JetFuel, HeatOil, FuelOil, and CrudeU to SuperSink. Note that the flow through this last arc represents fuel consumption and loss of the Crude Unit.

The SuperSource node becomes the unique supply node with an arbitrarily large supply. Similarly, the SuperSink node becomes the unique demand node with demand equal to the SuperSource's supply. This new model has a known total supply and total demand. Finally, another arc from the SuperSource to the SuperSink is added to the model. The trick is to drive as much flow through the network as is feasible and profitable and to drain off excess flow through this additional arc connecting the SuperSource and SuperSink. In this example, demand for Gasoline, Jet fuel, Heating Oil, and Fuel Oil is known and must be met. The capacities and lower flow bounds through arcs directed from these nodes to the SuperSink should equal the demand for these products. These flow bounds cause flow to pass through the network, not through the arc from SuperSource to SuperSink. In some models, it may be beneficial to provide an incentive for flow to go through the network by specifying that the SuperSource to SuperSink arc is expensive.

The cost of arcs directed from SuperSource are either the purchase price per thousand tons of each type of crude oil or the cost of transporting Fuel Oil from the other refinery. The cost of arcs directed toward the SuperSink node are the revenue per thousand tons of each type of refined product.

The data sets required by the NETFLOW procedure are initialized by the following SAS steps:

```
data nodes;
   input _node_ $ _supdem_ ;
   cards;
Source  999999
Sink   -999999
;
```

```
data arcs;
   input _tail_ $ _head_ $ _cost_ _capac_ _lo_ ;
   cards;
Source   Sink        0     .       .
Source   ArabLite 280.74 110       .
Source   ArabHvy  265.53 165       .
Source   Brega    331.05  80       .
Source   refinry2  245     .       .
CrudeU   Sink        0     .       .
Gasoline Sink     -430    40      40
JetFuel  Sink     -300    20      20
HeatOil  Sink     -315    50      50
FuelOil  Sink     -250   145     145
;
```

The arcs in the previous data set are shown in Figure 4. The data for some of the remaining arcs of the model, as shown in Figure 3 are:

```
data arcs2;
   input _tail_ $ _head_ $ _cost_ _capac_ _lo_ ;
   cards;
ArabLite CrudeU      .     .       .
ArabHvy  CrudeU      .     .       .
Brega    CrudeU      .     .       .
CrudeU   LVNaptha    .     .       .
CrudeU   IVNaptha    .     .       .
CrudeU   VHeatOil    .     .       .
CrudeU   Vacdistl    .     .       .
IVNaptha Platform   5.81 23.25     .
VHeatOil LoCatCrk    .     .       .
     ...
;
```

The Appendix contains all the SAS code to initialize SAS data sets, run PROC NETFLOW, and print the solution.

THE EXAMPLE'S SIDE CONSTRAINTS

This example demonstrates proportional constraints. Recall that these are used to restrict the flow through some arc to be proportional to the flow in another arc. In particular, these constraints:

- specify the amounts of downstream products of the Crude Unit (Light Virgin Naptha, Intermediate Virgin Naptha, Virgin Heating Oil, Vacuum Distillate, Vacuum Residue (conveyed in an arc from the nodes CrudeU to FuelOil2) and, because of flow conservation, Crude Unit fuel consumption and loss (conveyed in an arc from the nodes CrudeU to SuperSink)) as being dependent on the amount and type of crude oil used:

```
LiteVNap: 0.035*ArabLite_CrudeU + 0.030*ArabHvy_CrudeU +
          0.045*Brega_CrudeU = CrudeU_LVNaptha
ImedVNap: 0.1  *ArabLite_CrudeU + 0.075*ArabHvy_CrudeU +
          0.135*Brega_CrudeU = CrudeU_IVNaptha
VirHtOil: 0.390*ArabLite_CrudeU + 0.300*ArabHvy_CrudeU +
          0.430*Brega_CrudeU = CrudeU_VHeatOil
VacuumDt: 0.285*ArabLite_CrudeU + 0.230*ArabHvy_CrudeU +
          0.280*Brega_CrudeU = CrudeU_Vacdistl
VacResdu: 0.165*ArabLite_CrudeU + 0.335*ArabHvy_CrudeU +
          0.100*Brega_CrudeU = CrudeU_FuelOil2
```

- specify the amounts of downstream products of the Platform (Butane, Fuel Equivalent and, because of flow conservation, Reformate (conveyed in an arc from the nodes Platform to Gasoline)) as being dependent on the amount of Intermediate Virgin Naptha used by the Platform.

```
ReformBu: 0.02*IVNaptha_Platform = Platform_Butane
ReformFE: 0.08*IVNaptha_Platform = Platform_FuelEqui
```

- specify the amounts of downstream products of the Low Severity Catalytic Cracker (Butane, Catalytic Cracked Distillate, Fuel Equivalent and, because of flow conservation, Catalytic Cracked Naptha (conveyed in an arc from the nodes LoCatCrk to Gasoline)) as being dependent on the amounts of input streams Virgin Heating Oil and Vacuum Distillate:

```
LoCCBut: 0.020*VHeatOil_LoCatCrk + 0.050*Vacdistl_LoCatCrk
         = LoCatCrk_Butane
LoCCVHO: 0.680*VHeatOil_LoCatCrk + 0.585*Vacdistl_LoCatCrk
         = LoCatCrk_CCrDistl
LoCCFE:  0.025*VHeatOil_LoCatCrk + 0.040*Vacdistl_LoCatCrk
         = LoCatCrk_FuelEqui
```

- downstream products of the High Severity Catalytic Cracker:

```
HiCCBut: 0.0325*VHeatOil_HiCatCrk + 0.060*Vacdistl_HiCatCrk
         = HiCatCrk_Butane
HiCCVHO: 0.5550*VHeatOil_HiCatCrk + 0.440*Vacdistl_HiCatCrk
         = HiCatCrk_CCrDistl
HiCCFE:  0.0350*VHeatOil_HiCatCrk + 0.050*Vacdistl_HiCatCrk
         = HiCatCrk_FuelEqui
```

- recipes for blending Jet Fuel. There are three of them, modeled by these constraints:

```
JFRecip1: IVNaptha_JetFuel1 = 0.3*JetFuel1_JetFuel
JFRecip2: LVNaptha_JetFuel2 = 0.2*JetFuel2_JetFuel
JF3IVN:   IVNaptha_JetFuel3 = 0.1*JetFuel3_JetFuel
JF3VHO:   VHeatOil_JetFuel3 = 0.1*JetFuel3_JetFuel
```

The constraint JFRecip1 states that the 30 percent of JetFuel made using recipe 1 is Intermediate Virgin Naptha. Because of flow conservation, the remaining 70 percent of JetFuel made using recipe 1 must be Virgin Heating Oil. By similar reasoning, 80 percent of JetFuel made using recipe 2 must be Virgin Heating Oil, and 80 percent of JetFuel made using recipe 3 must be Catalytic Cracked Distillate.

NONARC VARIABLES

Nonarc variables are general variables that can be introduced into the model. They are like arc variables in that they can have costs, lower and upper bounds, and coefficients in side constraints.

The main use of nonarc variables in this example is that in some cases quality or attribute data are expressed as a per ton rate, a unit of weight. For example, the cost of transportation of crude oil shipped to the refinery is $24.15/ton. In other cases, data are expressed as a per barrel rate, a unit of volume. The units of flow through the arcs of the network are tons. To convert from tons to barrels, the specific gravities of streams or blend ingredients are used. The specific gravity is the number of tons a cubic meter of product weighs. Also needed is the fact that 6.29 barrels occupy one cubic meter.

The Crude Unit has a throughput capacity (excluding fuel and loss) of 65,000 barrels per day, and an operating cost of 0.15 dollars per barrel of feed. Given that the specific gravities of Arabian Light, Arabian Heavy, and Brega crude oil are 0.858, 0.886, and 0.823 tons per cubic meter, respectively, the following constraint is required:

```
CrudeUthruput = 6.29/0.858*ArabLite_CrudeU +
                6.29/0.886*ArabHvy_CrudeU  +
                6.29/0.823*Brega_CrudeU

CrudeUthruput = 7.33*ArabLite_CrudeU +
                7.10*ArabHvy_CrudeU  +
                7.64*Brega_CrudeU
```

CrudeUthruput is the name of a nonarc variable, that has a value the throughput of the Crude Unit in thousands of barrels. The objective function and capacity is $0.15 times 1000 = 150$ dollars per thousand barrels and $65 times 31 = 2015$ barrels, respectively.

The Platform also has an operating cost and capacity, but as it has only one feed, which is conveyed in the arc from the nodes IVNaptha to Platform, the cost and capacity can be associated with that arc, rather than setting up a nonarc variable and side constraint.

The model requires another nonarc variable, Gasolinebarrels, that is the thousands of barrels of Gasoline that is produced. The specific gravities of products that are blended into gasoline, Butane, Light Virgin Naptha, Reformate, Catalytic Cracked Distillate from the Low Severity Catalytic Cracker or the High Severity Catalytic Cracker are 0.570, 0.650, 0.737, 0.730, and 0.750, respectively.

```
Gasolinebarrels = 11.04*Butane_Gasoline   +
                   9.68*LVNaptha_Gasoline +
                   8.83*Platform_Gasoline +
                   8.62*LoCatCrk_Gasoline +
                   8.39*HiCatCrk_Gasoline
```

Gasoline is blended with 0.4 grams per liter and lead costs 7.80 dollars per kilogram. Therefore, the nonarc variable Gasolinebarrels should have an objective function coefficient of 0.4 grams/liter $times$ 7.80/1000 dollars/gram $times$ 1000 liters/cubic meter $dividedby$ 6.29 barrel/cubic meter = 0.50 dollars/barrel.

Gasolinebarrels appears in three other constraints: GasRVP (Reid's Vapor pressure), Gas100dg (percent of gasoline distilled at 100 degrees Celcius), and RsOctane (Research Octane of gasoline). These requirements are expressed in volumetric terms, not by weight.

```
GasRVP  : 827.63*Butane_Gasoline   +
          116.12*LVNaptha_Gasoline +
           51.21*Platform_Gasoline +
           60.32*LoCatCrk_Gasoline +
           75.48*HiCatCrk_Gasoline >= 10*Gasolinebarrels
Gas100dg: 1103.51*Butane_Gasoline   +
           919.31*LVNaptha_Gasoline +
           298.71*Platform_Gasoline +
           516.99*LoCatCrk_Gasoline +
           536.75*HiCatCrk_Gasoline <= 60*Gasolinebarrels
RsOctane: 1121.28*Butane_Gasoline   +
           835.12*LVNaptha_Gasoline +
           874.20*Platform_Gasoline +
           818.04*LoCatCrk_Gasoline +
           830.70*HiCatCrk_Gasoline >= 99.37*Gasolinebarrels
```

The last side constraint specifies that fuel oil must have an acceptable sulphur content.

The data set that creates side constraint data is:

```
data cons;
   input _column_ $17. _row_ $ _coef_ _type_ $;
   cards;
ArabLite_CrudeU   LiteVNap    0.035 .
ArabHvy_CrudeU    LiteVNap    0.03  .
Brega_CrudeU      LiteVNap    0.045 .
CrudeU_LVNaptha   LiteVNap   -1.0   .
ArabLite_CrudeU   ImedVNap    0.1   .
      ...
;
```

The Appendix contains data for all observations for this data set. To run NETFLOW and print the optimal solution, use:

```
proc append
   base=arcs data=arcs2;
proc netflow
   nodedata=nodes arcdata=arcs
   condata=cons sparsecondata defcontype=eq
   conout=solution;
run;
proc print data=solution(where=(_flow_^=0.0 &&
                                _tail_^="Source  " &&
                                _head_^="Sink    " &&
                                _name_=" "));
   var _tail_ _head_ _flow_;
run;
```

The following messages are output to the SAS log by the NETFLOW procedure. PROC NETFLOW first ignores the side constraints and solves the network, then performs more optimization by taking the side constraints into account. For many problems, this strategy pays off.

```
NOTE: Number of nodes= 25 .
NOTE: Number of supply nodes= 1 .
NOTE: Number of demand nodes= 1 .
NOTE: Total supply= 999999 , total demand= 999999 .
NOTE: Number of arcs= 56 .
NOTE: Number of nonarc variables= 2 .
NOTE: Number of iterations performed (neglecting any
      constraints)= 16 .
NOTE: Of these, 10 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Minimal total cost= -10466.7 .
NOTE: Number of <= side constraints= 2 .
NOTE: Number of == side constraints= 19 .
NOTE: Number of >= side constraints= 2 .
NOTE: Number of arc and nonarc variable side constraint
      coefficients= 85 .
NOTE: Number of iterations, optimizing with constraints= 35 .
NOTE: Of these, 7 were degenerate.
NOTE: Optimum reached.
NOTE: Minimal total cost= -5044.711003 .
NOTE: The data set WORK.SOLUTION has 58 observations and
      14 variables.
```

The optimal solution is:

OBS	_TAIL_	_HEAD_	_FLOW_
4	Platform	Butane	0.275
5	LoCatCrk	Butane	0.197
6	HiCatCrk	Butane	2.551
7	LoCatCrk	CCrDistl	6.711
8	HiCatCrk	CCrDistl	20.854
9	Platform	FuelEqui	1.098
10	LoCatCrk	FuelEqui	0.247
11	HiCatCrk	FuelEqui	2.180
12	FuelOil2	FuelOil	145.000
16	FuelEqui	FuelOil2	3.525
19	CrudeU	FuelOil2	20.435
20	refinry2	FuelOil2	121.041
21	LoCatCrk	Gasoline	2.714
22	HiCatCrk	Gasoline	20.039
23	LVNaptha	Gasoline	1.867
24	Platform	Gasoline	12.357
25	Butane	Gasoline	3.023
26	CCrDistl	HeatOil	27.565
27	VHeatOil	HeatOil	22.435
28	VHeatOil	HiCatCrk	6.777
29	Vacdistl	HiCatCrk	38.847
30	CrudeU	IVNaptha	14.847
31	JetFuel1	JetFuel	3.724
32	JetFuel2	JetFuel	16.276
34	IVNaptha	JetFuel1	1.117
35	VHeatOil	JetFuel1	2.607
36	LVNaptha	JetFuel2	3.255
37	VHeatOil	JetFuel2	13.021
41	CrudeU	LVNaptha	5.122
42	VHeatOil	LoCatCrk	9.869
44	ArabLite	CrudeU	103.785
46	Brega	CrudeU	33.100
47	IVNaptha	Platform	13.730
54	CrudeU	VHeatOil	54.709
55	CrudeU	Vacdistl	38.847

EXTENDING MODELS

In the optimal solution of the refinery model, the gasoline produced has an octane number of 99.37, the specified minimum octane that gasoline must have. In refinery models discussed by DeWitt et al. [1989], the finished gasoline blends often exceeded some of their specifications. They called this "giveaway". Of the requirements for gasoline, they identified octane giveaway as being especially important.

The octane rating of gasoline can be boosted by lead additives. In the past, adding lead was a viable method to ensure gasoline was within specifications. However, governmental regulations lowered the amount of lead allowed in regular gasoline, and new vehicles were required to use unleaded gasoline. With these new regulations, the higher octane blend ingredients were in more demand

and the value of these increased. The price of gasoline also varied a great deal.

Models for determining the operational parameters of refineries usually do not have long planning horizons. Predicting the value of blend ingredients and gasoline revenues for these models is less difficult than for strategic planning models that require costs and revenues farther into the future. Giveaway is then a significant factor in determining blending profitability. The objective is not only minimizing costs and maximizing revenues but also minimizing giveaways.

Nonarc variables are used to model giveaways. Recall the octane constraint in the example model:

```
RsOctane: 1121.28*Butane_Gasoline   +
          835.12*LVNaptha_Gasoline +
          874.20*Platform_Gasoline +
          818.04*LoCatCrk_Gasoline +
          830.70*HiCatCrk_Gasoline >= 99.37*Gasolinebarrels
```

A nonarc variable OctaneGiveAway is introduced into the model. The constraint RsOctane is changed from being of type ''greater than or equal to'' (an inequality), to ''equal to'' (an equality).

```
RsOctane: 1121.28*Butane_Gasoline   +
          835.12*LVNaptha_Gasoline +
          874.20*Platform_Gasoline +
          818.04*LoCatCrk_Gasoline +
          830.70*HiCatCrk_Gasoline =
                  99.37*Gasolinebarrels + OctaneGiveAway
```

The value of OctaneGiveAway should be nonnegative and can optionally be limited by specifying an upper bound. Setting OctaneGiveAway's objective function coefficient requires judgement for assigning the weight giveaway should have compared to the weight for other costs and revenues.

Octane cannot be modeled directly. Octane blending equations have complex forms, usually containing exponentials, logarithms, and quadratic and other interaction terms. This introduces nonlinearities into the model. You can use PROC NLP to solve such models.

Palmer et al. [1984] state that they convert Octane characteristics of blend ingredients of gasoline into blending ''numbers''. These can be used in linear octane equations, such as the RsOctane side constraint in the example. We are not told how they determined the blending numbers. Sometimes, nonlinear relationships are approximated using linear relationships, but the linear approximations are good only in a small region, such as when the octane giveaway is small. Specifying a big objective function coefficient for OctaneGiveAway may be necessary to ensure that the linear octane constraint is applicable to the optimal solution.

Using nonarc variables to change inequality constraints into equality constraints are also useful in cases when you want to be rewarded for not using all available resource. In these cases, the objective function coefficients of these nonarc variables are profits (unlike OctaneGiveAway's objective function coefficient).

INTEGER RESTRICTED VARIABLES

Not only should models that have no network component, or only a small network component, be solved by PROC LP, but also you should use PROC LP when you need the following:

- parametric programming
- range analysis
- analysis of solution sensitivities to changes in constraint right-hand-side constants and objective function coefficients
- integer programming, (some decision variables must have only integer values)
- the ability to have Special Ordered Sets (SOS)

For a refinery model, these features are illustrated in several examples in the PROC LP chapter in the *SAS/OR User's Guide* [1993].

Integer programming is used extensively in planning models that allow only one or a few options to be chosen from a larger set of actions. For example, if two new refineries can be built at five possible sites, a model has binary variables b_1, b_2, b_3, b_4, and b_5. There is a constraint to limit the number of refineries built: $b_1 + b_2 + b_3 + b_4 + b_5 \leq 2$. b_i has value 1.0 if a refinery is built at site i, and 0 otherwise. If $b_i = 0$, the part of the model associated with a refinery at that site is ''closed''. If $b_i = 1$, the part of the model associated with a refinery at that site is ''open'' to process crude oil and product refined products. The objective function coefficient of b_i is the construction cost of a refinery at site i. If refinery i is built, $b_i = 1$ and the construction cost is added to the solution total cost. If refinery i is not built, $b_i = 0$ and no construction cost is incurred.

The capital for new refineries may not be available immediately. Therefore, this type of model can be used not only to choose where the refineries are built, but also when. Also, binary variables might be required to determine the optimal size of the new refinery.

For short term models such as our refinery example, integer programming would be useful if all jet fuel produced must use only one blend recipe, or if the Low Severity Catalytic Cracker is used, the High Severity Catalytic Cracker cannot be used, and vice-versa.

The following simple example illustrates how to choose which of two possible refineries should be built. The first refinery costs \$100K to build and has an operating cost of \$0.25K per unit of throughput and a capacity of 675 units. The second refinery is cheaper to build (\$75K) but has a higher operating cost (\$0.75K per unit of throughput) but, in it's favor, has a larger capacity (1800 units). The optimal throughputs of the refineries are assigned to the variables x1 and x2. Because of the constraints setbin1 and setbin2, and as the variables b1 and b2 are flagged as binary and have expensive objective function coefficients, (the refinery construction costs), if $x1 = 0$, then $b1 = 0$. If $x1 > 0$, then $b1 = 1$. Similarly, if $x2 = 0$, then $b2 = 0$. If $x2 > 0$, then $b2 = 1$. The last constraint, only_one, ensures only one refinery is built. To initialize input data sets and run PROC LP, use:

```
data;
   input _row_ $ x1 x2 b1 b2 _type_ $ _rhs_;
   cards;
object  .25  .75   -100    -75 max     .
upperbd 675 1800      .      . upperbd .
setbin1   1    0 -10000      0 le      0
setbin2   0    1      0 -10000 le      0
only_one  0    0      1      1 eq      1
bin       .    .      1      1 binary  .
;
proc lp;
   run;
```

PROC LP provides several reports concerning the optimal solution. The Variable Summary indicates that the second type of refinery should be built (note the Activity value for B2).

```
                    VARIABLE  SUMMARY

   Variable                                                  Reduced
Col Name     Status   Type      Price     Activity           Cost

  1 X1       DEGEN UPPERBD       0.25     0.000000       0.000000
  2 X2             UPPERBD       0.75     1800.000       0.750000
  3 B1       DEGEN  BINARY       -100     0.000000       0.000000
  4 B2              BINARY        -75     1.000000      -2475.000
  5 setbin1          SLACK                0.000000      -0.250000
  6 setbin2  BASIC   SLACK                8200.000       0.000000
```

PROC NETFLOW can call upon the IBM software product Optimization Subroutine Library (OSL), if available, to solve integer programming problems, possibly with special ordered sets (SOS), even if the model has no network.

CONCLUSION

This paper presents modeling aspects of the oil and gas process industry from the perspective of network flows. It illustrates that with the addition of side constraints and nonarc variables, network flow models can be as comprehensive a modeling tool as general linear programs. This paper also shows that these models can be a valuable tool for managing operations. When the network component of such a model is large compared to the non-network component, PROC NETFLOW is the appropriate optimizer for solving the model. This can be seen from several perspectives. First, the optimizer has a specialized algorithm that can take advantage of the network structure during optimization. In addition, the network representation of the model is more easily maintained and updated. For example, if the model has to be expanded to take into account other aspects of the business operation, or more detail is needed, additional structure can be readily appended to the data files containing the model description.

For some models, however, PROC NETFLOW is not appropriate. Often, in models used for strategic planning, integer constraints on some variables are needed. This is apparent when evaluating possible construction of new refinerys. In this case, PROC LP is the appropriate choice for model solution. There are also cases where a nonlinear optimizer is appropriate, such as when modeling nonlinear functions for octane determination. In these cases, it may be necessary to use PROC NLP.

Finally, an advantage of using the SAS System for process modeling in the oil and gas industry is that, in addition to providing the optimization procedures for solving the models, the SAS System provides components for supporting data entry, model generation and management, and reporting. The data may be entered or extracted from various sources. The model can be generated using the SAS language. Statistical analysis and forecasts are needed for:

- the demand for refined products
- crude oil and refined product prices
- refinery activities, costs, and capacities
- transportation and storage costs

These important issues, not addressed in this paper, provide the data environment for the modeling exercise and the optimization process. It is well established that the SAS System is a leader in the data management area and provides sound support for the optimization procedures provided in the SAS/OR software product.

REFERENCES

1. Aronofsky, J.S., Dutton, J.M., and Tayyabkhan, M.J. (1978), ''Managerial Planning with Linear Programming: In Process Industry Operations'', New York: John Wiley and Sons, Inc.
2. P. Bayard (1993), ''Improved, Integrated Yield-Accounting System Saves Time and Money'', in *Oil & Gas Journal* 1993, 67-69.
3. Buchanan J.E., Garven S.C., Genis O., Shapiro J.F., Singhal V., Thomas J.M., and Torpis S., (1990), ''A Multi-Refinery, Multi-Period Modeling System for the Turkish Petroleum Refining Industry'', *Interfaces 20* 4 1990, 48-60.
4. DeWitt C.W., Lasdon L.S., Waren A.D., Brenner D.A., and S.A. Melhem (1989), ''OMEGA: An Improved Gasoline Blending System for Texaco'', *Interfaces 19* 1 1989, 85-101.
5. J. Hosseini (1986), ''Decision Analysis and Its Application in the Choice Between Two Wildcat Oil Ventures'', *Interfaces 16* 2 1986, 75-85.
6. B.A. Murtagh (1981), ''Advanced Linear Programming'', McGraw-Hill International Book Company.
7. Palmer K.H., Boudwin N.K., Patton H.A., Rowland A.J., Sammes J.D., and Smith D.M. (1984), ''A Model-Management Framework for Mathematical Programming'', New York: John Wiley and Sons, Inc.
8. SAS Institute Inc. (1993), *SAS/OR User's Guide, Version 6, First Edition*, Cary, NC., SAS Institute, Inc.

APPENDIX: SAS CODE- REFINERY EXAMPLE

```
data nodes;
   input _node_ $ _supdem_ ;
   cards;
Source  999999
Sink   -999999
;
data arcs;
   input _tail_ $ _head_ $ _cost_ _capac_ _lo_ ;
   cards;
Source   Sink        0     .       .
Source   ArabLite 280.74 110       .
Source   ArabHvy  265.53 165       .
Source   Brega    331.05  80       .
Source   refinry2  245     .       .
CrudeU   Sink        0     .       .
Gasoline Sink     -430    40      40
JetFuel  Sink     -300    20      20
HeatOil  Sink     -315    50      50
FuelOil  Sink     -250   145     145
;
data arcs2;
   input _tail_ $ _head_ $ _cost_ _capac_ _lo_ ;
   cards;
ArabLite CrudeU      .     .       .
ArabHvy  CrudeU      .     .       .
Brega    CrudeU      .     .       .
CrudeU   LVNaptha    .     .       .
CrudeU   IVNaptha    .     .       .
CrudeU   VHeatOil    .     .       .
CrudeU   Vacdistl    .     .       .
IVNaptha Platform   5.81 23.25     .
VHeatOil LoCatCrk    .     .       .
Vacdistl LoCatCrk    .     .       .
VHeatOil HiCatCrk    .     .       .
Vacdistl HiCatCrk    .     .       .
Platform Butane      .     .       .
Platform FuelEqui    .     .       .
LoCatCrk Butane      .     .       .
LoCatCrk CCrDistl    .     .       .
LoCatCrk FuelEqui    .     .       .
LoCatCrk Gasoline    .     .       .
HiCatCrk Butane      .     .       .
HiCatCrk CCrDistl    .     .       .
HiCatCrk FuelEqui    .     .       .
HiCatCrk Gasoline    .     .       .
LVNaptha Gasoline    .     .       .
LVNaptha JetFuel2    .     .       .
LVNaptha FuelOil2    .     .       .
IVNaptha JetFuel1    .     .       .
IVNaptha JetFuel3    .     .       .
VHeatOil JetFuel1    .     .       .
VHeatOil JetFuel2    .     .       .
VHeatOil JetFuel3    .     .       .
VHeatOil FuelOil2    .     .       .
Vacdistl FuelOil2    .     .       .
FuelEqui FuelOil2    .     .       .
Platform Gasoline    .     .       .
Butane   Gasoline    .     .       .
Butane   FuelOil2    .     .       .
CCrDistl JetFuel3    .     .       .
```

```
CCrDistl HeatOil     .     .       .
VHeatOil HeatOil     .     .       .
CCrDistl FuelOil2    .     .       .
CrudeU   FuelOil2    .     .       .
JetFuel1 JetFuel     .     .       .
JetFuel2 JetFuel     .     .       .
JetFuel3 JetFuel     .     .       .
refinry2 FuelOil2    .     .       .
FuelOil2 FuelOil     .     .       .
;
data cons;
   input _column_ $17. _row_ $ _coef_ _type_ $;
   cards;
ArabLite_CrudeU   LiteVNap    0.035 .
ArabHvy_CrudeU    LiteVNap    0.03  .
Brega_CrudeU      LiteVNap    0.045 .
CrudeU_LVNaptha   LiteVNap   -1.0   .
ArabLite_CrudeU   ImedVNap    0.1   .
ArabHvy_CrudeU    ImedVNap    0.075 .
Brega_CrudeU      ImedVNap    0.135 .
CrudeU_IVNaptha   ImedVNap   -1.0   .
ArabLite_CrudeU   VirHtOil    0.39  .
ArabHvy_CrudeU    VirHtOil    0.3   .
Brega_CrudeU      VirHtOil    0.43  .
CrudeU_VHeatOil   VirHtOil   -1.0   .
ArabLite_CrudeU   VacuumDt    0.285 .
ArabHvy_CrudeU    VacuumDt    0.23  .
Brega_CrudeU      VacuumDt    0.28  .
CrudeU_Vacdistl   VacuumDt   -1.0   .
ArabLite_CrudeU   VacResdu    0.165 .
ArabHvy_CrudeU    VacResdu    0.335 .
Brega_CrudeU      VacResdu    0.1   .
CrudeU_FuelOil2   VacResdu   -1.0   .
IVNaptha_Platform ReformBu    0.02  .
Platform_Butane   ReformBu   -1.0   .
IVNaptha_Platform ReformFE    0.08  .
Platform_FuelEqui ReformFE   -1.0   .
VHeatOil_LoCatCrk LoCCBut     0.02  .
Vacdistl_LoCatCrk LoCCBut     0.05  .
LoCatCrk_Butane   LoCCBut    -1.0   .
VHeatOil_LoCatCrk LoCCVHO     0.68  .
Vacdistl_LoCatCrk LoCCVHO     0.585 .
LoCatCrk_CCrDistl LoCCVHO    -1.0   .
VHeatOil_LoCatCrk LoCCFE      0.025 .
Vacdistl_LoCatCrk LoCCFE      0.04   .
LoCatCrk_FuelEqui LoCCFE     -1.0    .
VHeatOil_HiCatCrk HiCCBut     0.0325 .
Vacdistl_HiCatCrk HiCCBut     0.06   .
HiCatCrk_Butane   HiCCBut    -1.0    .
VHeatOil_HiCatCrk HiCCVHO     0.555  .
Vacdistl_HiCatCrk HiCCVHO     0.44   .
HiCatCrk_CCrDistl HiCCVHO    -1.0    .
VHeatOil_HiCatCrk HiCCFE      0.035  .
Vacdistl_HiCatCrk HiCCFE      0.05   .
HiCatCrk_FuelEqui HiCCFE     -1.0    .
IVNaptha_JetFuel1 JFRecip1   -1.0    .
JetFuel1_JetFuel  JFRecip1    0.3    .
LVNaptha_JetFuel2 JFRecip2   -1.0    .
JetFuel2_JetFuel  JFRecip2    0.2    .
IVNaptha_JetFuel3 JF3IVN     -1.0    .
JetFuel3_JetFuel  JF3IVN      0.1    .
VHeatOil_JetFuel3 JF3VHO     -1.0    .
JetFuel3_JetFuel  JF3VHO      0.1    .
CrudeUthruput     Cthruput   -1.0    .
ArabLite_CrudeU   Cthruput    7.33   .
ArabHvy_CrudeU    Cthruput    7.10   .
Brega_CrudeU      Cthruput    7.64   .
CrudeUthruput     .           0.15   objfn
CrudeUthruput     .        2015.0    upperbd
Gasolinebarrels   GasBarel   -1.0    .
Butane_Gasoline   GasBarel   11.04   .
LVNaptha_Gasoline GasBarel    9.68   .
Platform_Gasoline GasBarel    8.53   .
LoCatCrk_Gasoline GasBarel    8.62   .
HiCatCrk_Gasoline GasBarel    8.39   .
Gasolinebarrels   .           0.50   objfn
Gasolinebarrels   GasRVP    -10.0    ge
Butane_Gasoline   GasRVP    827.63   .
LVNaptha_Gasoline GasRVP    116.12   .
Platform_Gasoline GasRVP     51.21   .
LoCatCrk_Gasoline GasRVP     60.32   .
HiCatCrk_Gasoline GasRVP     75.48   .
Gasolinebarrels   Gas100dg  -60      le
Butane_Gasoline   Gas100dg 1103.51   .
LVNaptha_Gasoline Gas100dg  919.31   .
Platform_Gasoline Gas100dg  298.71   .
LoCatCrk_Gasoline Gas100dg  516.99   .
HiCatCrk_Gasoline Gas100dg  536.75   .
Gasolinebarrels   RsOctane  -99.37   ge
Butane_Gasoline   RsOctane 1121.28   .
LVNaptha_Gasoline RsOctane  835.12   .
Platform_Gasoline RsOctane  874.20   .
LoCatCrk_Gasoline RsOctane  818.04   .
HiCatCrk_Gasoline RsOctane  830.70   .
FuelOil2_FuelOil  FOSulphr   -3      le
ArabLite_CrudeU   FOSulphr    4      .
ArabHvy_CrudeU    FOSulphr    5      .
Brega_CrudeU      FOSulphr    0.6    .
VHeatOil_FuelOil2 FOSulphr    1      .
Vacdistl_FuelOil2 FOSulphr    1.7    .
CCrDistl_FuelOil2 FOSulphr    1.5    .
;
proc append
   base=arcs data=arcs2;
proc netflow
   nodedata=nodes arcdata=arcs
   condata=cons sparsecondata defcontype=eq
   conout=solution;
run;
proc print data=solution(where=(_flow_^=0.0 &&
                                _tail_^="Source  " &&
                                _head_^="Sink    " &&
                                _name_=" "));
   var _tail_ _head_ _flow_;
run;
```