



Optimizer Feasibility Example:

Transportation Assignment Scenario

Document version 1.0

Contents

- Feasibility Example 3
 - Prerequisites 3
 - Transportation Assignment Scenario 3
 - Using Optimizer to achieve an Objective 3
 - Optimization Problem 5
 - Required Optimizer elements 5
 - Ensuring feasibility in the end-user dashboard 12

Feasibility Example

This guide provides an example for an Anaplan Workspace Administrator who has little or no experience with the mathematical concepts and tools associated with linear programming, linear optimization, and related fields.

Prerequisites

Experience of working with Anaplan. We assume that you know how to create:

- Lists
- Modules with line items
- Actions, such as a process
- Dashboards, including a button that runs an action (in this case, a process)

Transportation Assignment Scenario

To help with understanding the complex concept of optimization, we've created a Transportation Assignment problem, where our aim is to minimize the Objective Function. In this scenario, we want to minimize the cost of shipping products from each factory plant to each retail center.

The example provides a step-by-step guide on how to set up a series of modules for Optimizer. There are a number of rules these modules must adhere to:

- The line items used in modules should only contain **Numeric** items.
- The line items used in modules must have **Time Scale** and **Versions** set as **Not Applicable**.
- Any variables in expressions must be numeric - real number variables are faster to solve than binary or integer types.
- Intermediate (calculation) cells - the **Summary** setting must be **SUM**.
- Objectives - the **Summary** setting must be **SUM** to optimize all cells.
- Constraints must give a Boolean result - the **Summary** setting for Boolean line items must be **All**.

Using Optimizer to achieve an Objective

Suppose you are a Supply Chain Analyst. You have three factory plants making products, and four retail centers selling products. Your broad business goal is to figure out how many products each plant should send to each center to minimize the costs of transportation. However, you understand that there are some constraints in your network that you need to account for.

The Optimizer can help you find the optimal transportation assignment of products from factory plants to retail centers.

The constraints are that:

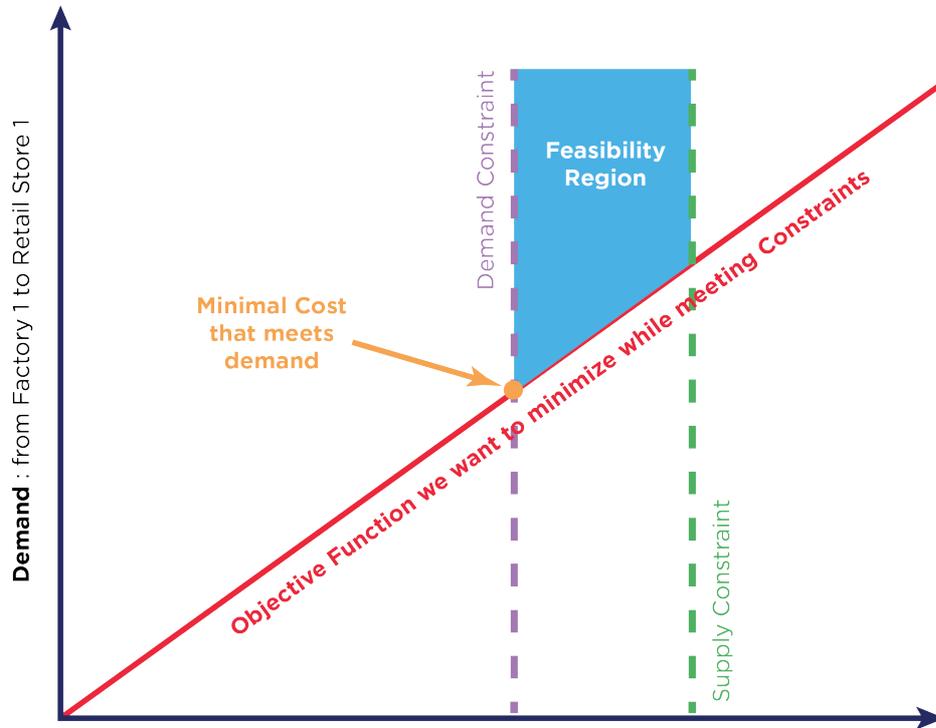
- We can't ship more products than a given factory plant can produce.
- We must ship enough products to meet the demand of each retail center.

The formula for the Objective Function and the formulas for the Constraints must be linear.

Formula	Description	Formula	Linear Relationship?
Objective Function	Minimize cost of shipping from each Factory Plant to each Retail Center	$\text{Variable}.\text{Assignment} * \text{InputData}.\text{Cost}$	Yes, because if the sum of all outlets require more products, the shipping cost increases proportionally
Factory Constraint	Can't ship more products than the factory can produce	$\text{Variable}.\text{Assignment} \leq \text{InputData}.\text{Supply}$	Yes, because if a given factory plant ships more products, the shipping cost increases proportionally
Retail Center Constraint	Must receive enough products to meet customer demand	$\text{Variable}.\text{Assignment} = \text{InputData}.\text{Demand}$	Yes, because if a given outlet requires more products, the shipping cost increases proportionally

These problems can be solved with linear programming because:

- A graph of each formula is a straight line
- A graph of the set of formulas gives us exactly one feasibility region



Supply : Number of products to ship

At the simplest level, a linear programming problem is straightforward. However, if we have many factory plants and many retail centers, the possible combinations become large and complex.

Optimization Problem

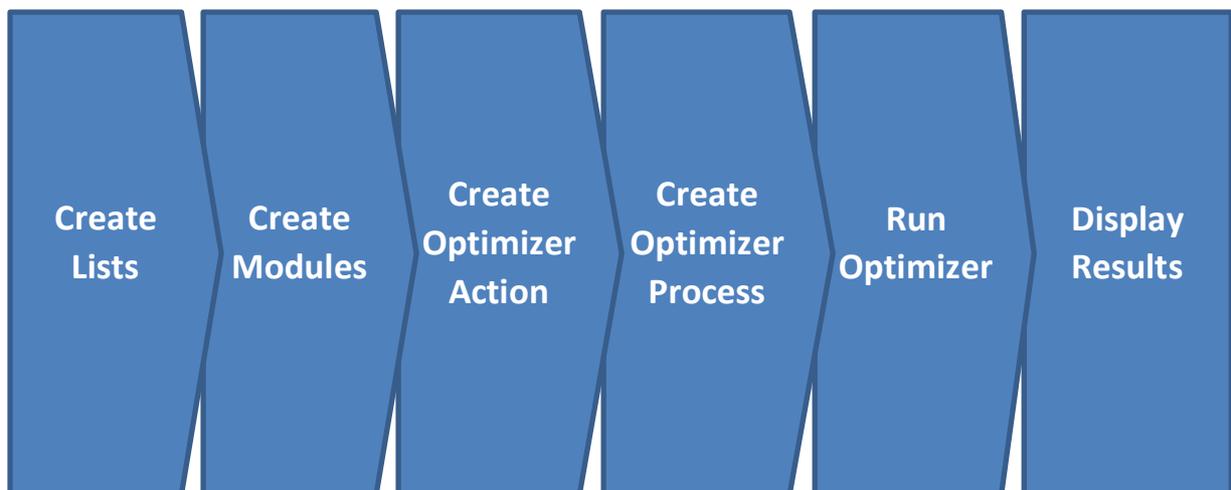
Your broad business goal is to minimize the costs of transportation by finding the optimal number of widgets each plant should send to each center.

To solve this problem, we define the Objective, Variable, and Constraints.

	Requirement	In this example...
Objective	The Objective must be a numeric line item with a formula depending on the variable(s). This formula must be linear regarding the variable.	The objective is to minimize the Objective Function, the total cost of transporting widgets from our set of factory plants to our set of retail centers.
Variable	The Variable must be a numeric line item.	‘How many widgets should each plant send to each center?’ This is the variable we change to obtain the objective of minimal cost. The optimal values for the variable are the output of the optimization problem.
Constraints	Constraints must be Boolean line items with formulas depending on the variable(s). These formulas must be linear.	There are two constraints: <ul style="list-style-type: none"> • The total demand of each center must be met exactly, so each center must receive exactly the number of widgets that it demands. • A plant can’t supply more than what it’s production capacity, so each plant can’t send more widgets than what it can produce.

Required Optimizer elements

Now that we’ve defined the optimization problem, we build elements in the model that enable us to set up the Optimizer action.



Create lists

You need two lists: **Centers** that lists your retail centers, and **Plants**, to hold your factory plants. Set the [Top Level Item](#) in each list to **All Centres** and **All Plants** respectively.

Next, you'll build four modules to hold the **Input Data**, **Variables**, **Objective** and **Constraints**.

Create modules

The collection of data that we know when we approach this problem allow us to provide input data.

1. Create the **Input Data** module with three line items **Supply**, **Demand** and **Cost** as **Rows**, and no **Columns** dimension.
2. After creating the **Input Data** module, open **Blueprint** and edit the **Applies To** column so:
 - the **Plants** list applies to **Supply**
 - the **Centres** list applies to **Demand**
 - the **Plants** and the **Centres** lists applies to **Cost**
3. In **Blueprint**, select the **Cost** line item and click the **Pivot** button. Set the **Centres** lists as **Columns**, and the **Plants** list as **Rows**.

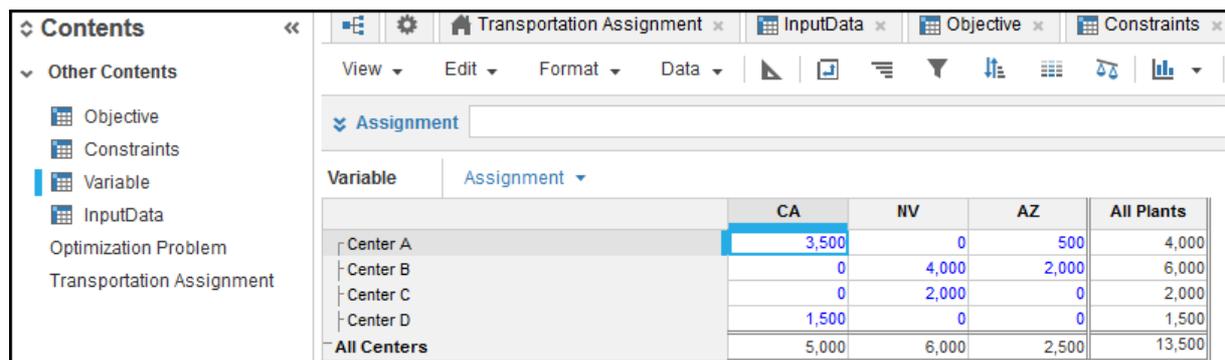
The **Input Data** module is set up like this when complete:

Supply	How many products each plant can produce.																															
Demand	How many products each center requires																															
Cost	Unit price to transport a single widget from each specific plant to each specific center	<table border="1" style="margin-top: 10px;"> <thead> <tr> <th></th> <th>Center A</th> <th>Center B</th> <th>Center C</th> <th>Center D</th> <th>All Centers</th> </tr> </thead> <tbody> <tr> <td>CA</td> <td>7</td> <td>222</td> <td>7</td> <td>6</td> <td>242</td> </tr> <tr> <td>NV</td> <td>3</td> <td>5</td> <td>2</td> <td>3</td> <td>13</td> </tr> <tr> <td>AZ</td> <td>2</td> <td>5</td> <td>4</td> <td>5</td> <td>16</td> </tr> <tr> <td>All Plants</td> <td>12</td> <td>232</td> <td>13</td> <td>14</td> <td>271</td> </tr> </tbody> </table>		Center A	Center B	Center C	Center D	All Centers	CA	7	222	7	6	242	NV	3	5	2	3	13	AZ	2	5	4	5	16	All Plants	12	232	13	14	271
	Center A	Center B	Center C	Center D	All Centers																											
CA	7	222	7	6	242																											
NV	3	5	2	3	13																											
AZ	2	5	4	5	16																											
All Plants	12	232	13	14	271																											

4. Create the **Variable** module using the **Centers** list as **Rows**, the **Plants** list as **Columns**, and a line item titled **Assignment** as **Pages**. Set the **Summary** method to SUM for **Rows** and **Columns**.

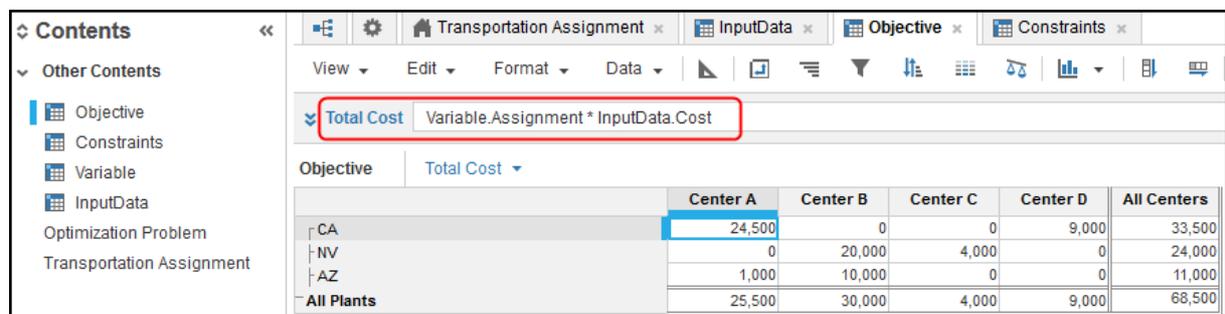
The module called **Variable** contains the line item variable that the Optimizer will change to minimize the total transportation cost. We call the line item **Assignment** because we're changing the transportation assignment of widgets.

As this line item will contain the output of the Optimizer action, you can't use a formula. If a formula is used on a line item, the calculation result can't be updated by Optimizer. The blue text in the image below indicates that the fields are editable.



	CA	NV	AZ	All Plants
Center A	3,500	0	500	4,000
Center B	0	4,000	2,000	6,000
Center C	0	2,000	0	2,000
Center D	1,500	0	0	1,500
All Centers	5,000	6,000	2,500	13,500

- Create the **Objective** module using the **Centers** list as **Columns**, the **Plants** list as **Rows** and a line item titled **Total Cost** as **Pages**.
The **Objective** module contains a line item called **Total Cost** used to express the Objective Function, which calculates the total cost of transporting the widgets from the factory plants to the retail centers.
- Add the Objective Function formula to the Total Cost line item. The Objective Function is:
`Variable.Assignment*InputData.Cost`



	Center A	Center B	Center C	Center D	All Centers
CA	24,500	0	0	9,000	33,500
NV	0	20,000	4,000	0	24,000
AZ	1,000	10,000	0	0	11,000
All Plants	25,500	30,000	4,000	9,000	68,500

As we can't just ship all the widgets from a given plant to whichever center is the cheapest, we must also consider the constraints.

- Create the **Constraints** module with two line items **TotalDemand** and **SupplyMax** as **Rows** with no **Column** dimension.
- After creating the **Constraints** module, open **Blueprint** and:
 - edit the **Applies To** column so the **Centres** list applies to **TotalDemand** and the **Plants** list applies to **SupplyMax**.
 - set the **Format** for both line items to **Boolean**.
 - scroll across to the **Summary** setting and open the **Summary Method** window. Set the **Summary** to **All**. Leave the **Time Summary** setting.
- TotalDemand** says that the widgets received by each center must equal exactly the number of widgets the center demanded. We can represent this using the Constraint function:
`Variable.Assignment = InputData.Demand`

The screenshot shows the Anaplan interface with the 'TotalDemand' constraint selected. The constraint formula is `Variable.Assignment = InputData.Demand`. The data table below shows the configuration for four centers:

Center	Assignment
Center A	<input checked="" type="checkbox"/>
Center B	<input checked="" type="checkbox"/>
Center C	<input checked="" type="checkbox"/>
Center D	<input checked="" type="checkbox"/>
All Centers	

SupplyMax says that the widgets transported by each plant can't exceed the capacity of each plant, which we can represent with this **Constraint** function: `Variable.Assignment <= InputData.Supply`.

The screenshot shows the Anaplan interface with the 'SupplyMax' constraint selected. The constraint formula is `Variable.Assignment <= InputData.Supply`. The data table below shows the configuration for three plants:

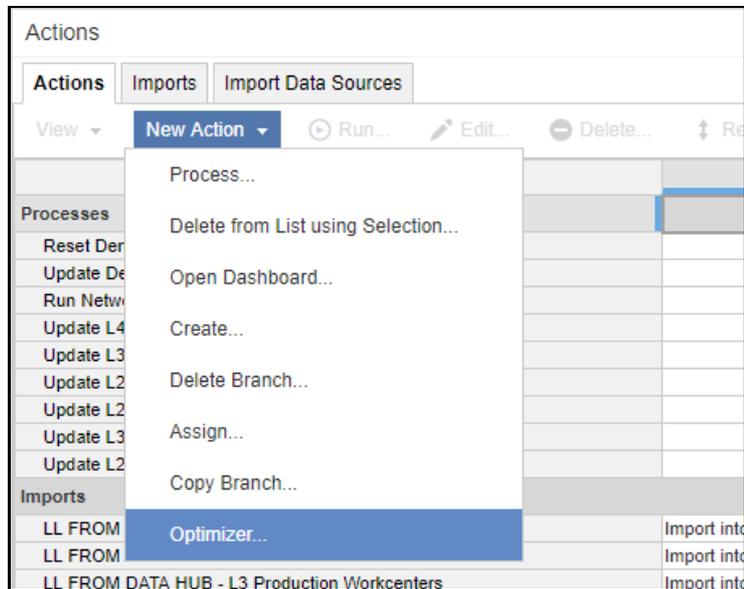
Plant	Assignment
CA	<input checked="" type="checkbox"/>
NV	<input checked="" type="checkbox"/>
AZ	<input checked="" type="checkbox"/>
All Plants	

Now that we have all the necessary elements in the Anaplan model, we can set up the Optimizer action.

Create the Optimizer Action

Create an Optimizer action that defines the type of problem you want to solve, the objective you want to achieve, whether you want the smallest or largest value, and which line item contains the value you want to optimize.

1. Go to **Actions > New Actions > Optimizer**.



2. The Optimizer dialog displays. In the **Button text** box, enter a suitable name to describe the optimization.

 A screenshot of the 'Optimizer' dialog box. It contains several fields and options:

- Button text:** A text input field with a red asterisk indicating it is required.
- Time Out:** A numeric input field set to '300' with a 'seconds' label and minus/plus buttons.
- Problem:** A section with a red asterisk and two radio button options: 'Linear Programming' (selected) and 'Feasibility Programming'.
- Objective:** A section with a red asterisk and two radio button options: 'Minimize' (selected) and 'Maximize'.
- Line Item:** A dropdown menu with a red asterisk.
- Variable(s):** A section with a red asterisk, a dropdown menu, and a plus/minus button. Below it are 'Integer', 'min', and 'max' options.
- Constraint(s):** A section with a dropdown menu and a plus/minus button.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

3. In the **Problem** section, select either:
 - **Linear Programming** - to find the most optimal solution.
 - **Feasibility Programming** - to find the most feasible solution.

4. In the **Objective** section, select either:
 - **Minimize**, to find the smallest value (e.g. minimized cost).
 - **Maximize**, to find the highest value (e.g. profits).

On the **Line Item** list, select the line item **Total Cost** used to express the Objective Function.

5. In the **Variable** section, select at least one line item that specifies the value you want to change. The lists below each variable enable you to select the variable type, which defaults to **Integer**.

Optionally, you can configure **min** and **max** bounds as special case restrictions on each variable to help to define the feasible region. The optimization algorithm can handle these bounds more efficiently than other restrictions. They consume less memory and the algorithm is faster with them.

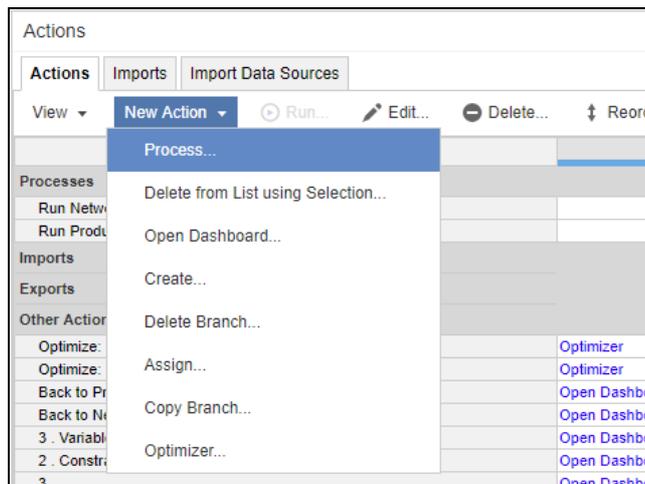
6. Optionally, in the **Constraint** section, select one or more constraint functions.
7. Click **Save**. The Optimizer action is saved.

Run Optimizer

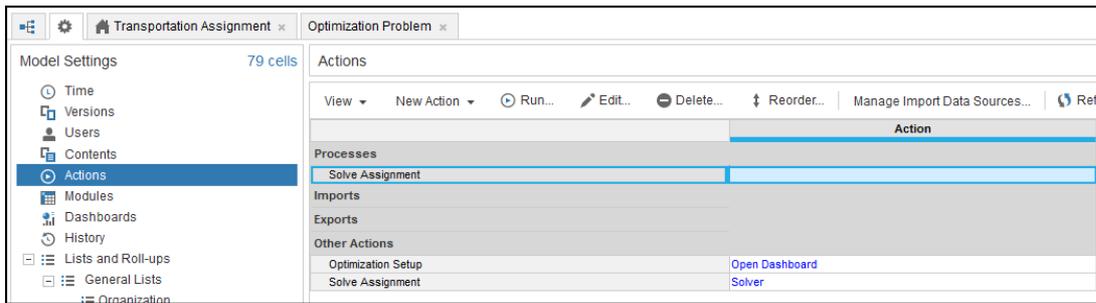
You can only run the Optimizer action once it has been [added to a process](#). A process is a sequence of Actions. An Action is an import, export, Optimizer or delete function.

The process can then be [published to the dashboard](#). Once run, you can't undo. However you can [restore the model to an historical ID](#).

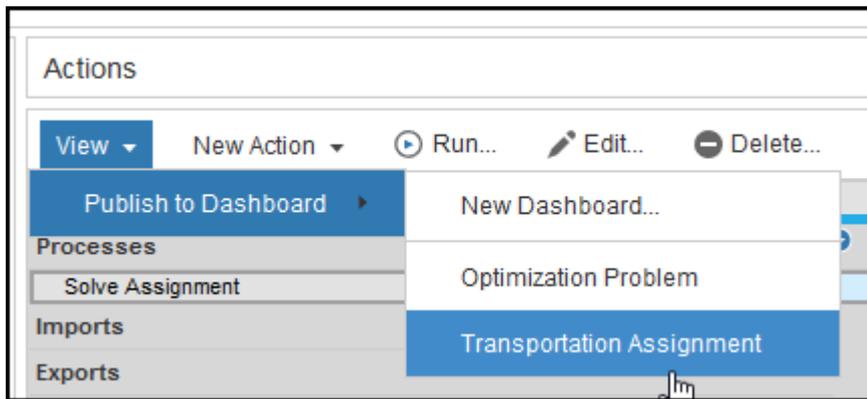
1. On the **New Action** list, click **Process**.



2. Enter a name for the process that will be used as button text when the process is added to the dashboard, and click **OK**. The new process is added to the list.
3. Select the process and click **Edit**.
4. In the **Available Actions** panel, select the Optimizer process you've created and click **OK** to save.
5. Select your process then on the **View** menu, click **Publish to Dashboard** and choose a target dashboard.



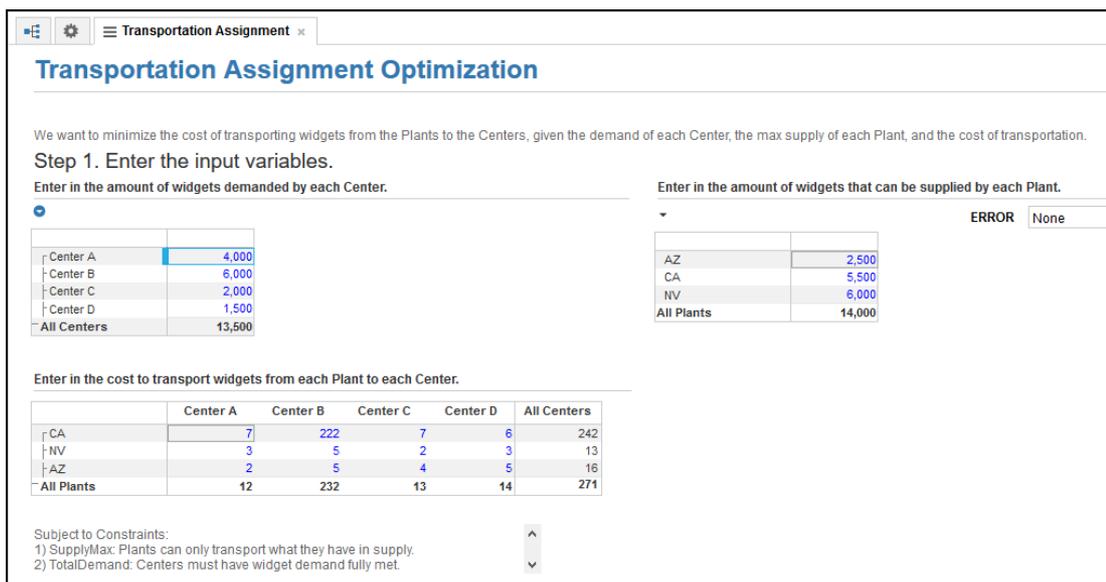
6. Click **Solve Assignment** to run the calculation. (Any end-user with access to the target dashboard can do this.)



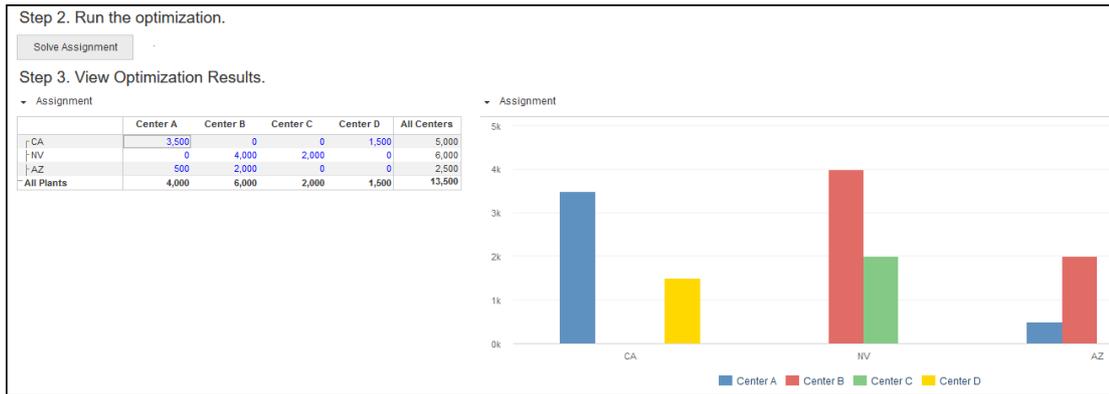
Display results through the end-user dashboard

We created a dashboard called **Transportation Assignment** and published the Optimizer action to the dashboard so that an end-user, such as a Regional Supply Chain Manager, can create 'what-if' scenarios with **Input Data**.

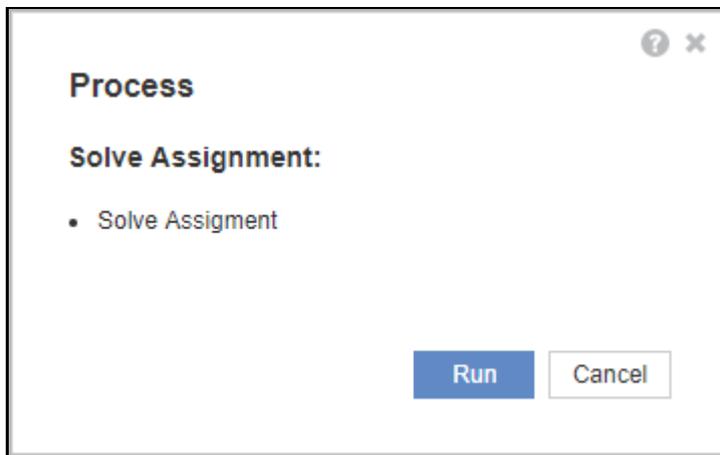
1. Adjust the input variables (at the top of the dashboard).



2. Click **Solve Assignment** in the dashboard to launch the action.



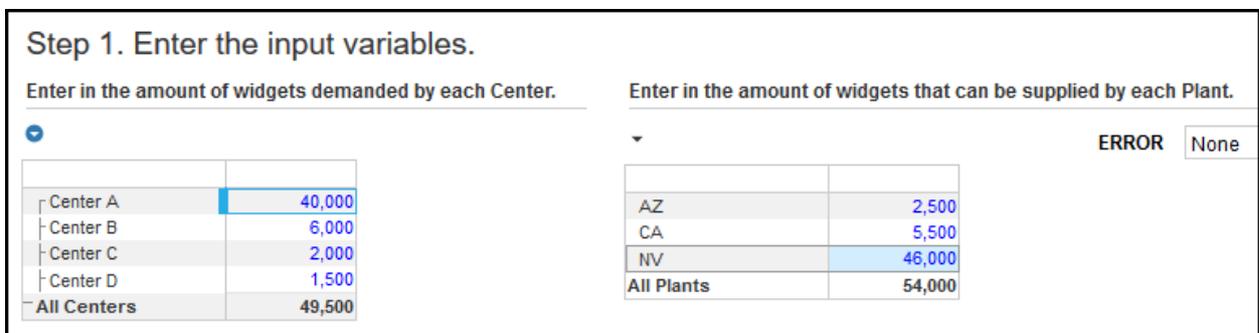
3. Click **Run** in the **Process** window to run the Optimizer.



4. Click **Close** when the process completes. The results are displayed on the dashboard.

Ensuring feasibility in the end-user dashboard

In the **Transportation Assignment** example, we have a constraint to fully meet the demand at each retail center. The following dashboard shows an error status of **None** because supply can meet demand and a solution is feasible.



The dashboard can alert the user if an adjustment to input values compromises the feasibility of solving the problem. If we increase total demand beyond the total supply, no solution is feasible because supply can't meet demand. The following shows the **Supply Does Not Meet Demand** error.