# Comments on the New Optimization Model

Comments in red by D. Gueorguiev, original model in black by Hadi Mohebalizadeh 9/25/23

## Business requirements:

**Force to nodes**: when a volume has inventory and force to node available, it must be used for fulfillment no matter the circumstances, including splits.

**EX1**: Nodes A, B, C have inventory for Order 1 for 2 SKUs. C has force\_to volume of 100, so force C to fulfill Order 1.

**Note 1**: if we cannot fulfill completely by using force\_to nodes we still want to fulfill with as many force\_to nodes as possible:

**EX2**: Node A is a force\_to node and it has an inventory of 1 of SKU1. Node B is a force\_to node and it has an inventory of 1 of SKU2. Node C is not a force\_to node and it has an inventory of 2 for SKU1 and inventory of 2 for SKU2. We have an order with request for 2 units of SKU1 and 2 units of SKU2.

Expected fulfillment: one unit quantity of SKU1 to be fulfilled by Node A, one unit quantity of SKU2 to be fulfilled by Node B, one unit quantity of SKU1 to be fulfilled by Node C and one unit quantity of SKU2 to be fulfilled by Node C. (*Please check with Iwo on this example and its relevancy*).

**EX3**. Node A, Node B and Node C are force\_to nodes. Node D is not force\_to node. Node A has inventory of 2 for SKU1 and inventory of 1 for SKU2. Node B has inventory of 2 for SKU1 and inventory of 1 for SKU2. Node B has lower shipping cost than node A. Node C has an orphan for SKU2. Node D has inventory of 10 for SKU1 and inventory of 10 for SKU2. Node D has lower CO2 emissions and lower shipping cost compared to Nodes A, B, and C.

We have an order with request for 3 units of SKU1 and 3 units of SKU2.

Expected fulfillment: 2 units of SKU1 to be fulfilled by Node B, 1 unit of SKU2 to be fulfilled by Node B, 1 unit of SKU2 by Node C, 1 unit of SKU1 by Node D, 1 unit of SKU2 by node D.

(*Please check with Iwo if having two depriortized nodes with inventory for the same SKUs is realistic scenario*).

**Deprioritized nodes**: only use the node if needed to fulfill the order. However, we can use de-prioritized nodes to reduce splitting. (**EX4**: Nodes A, B, C have inventory for Order 1 for 2 SKUs s. A and B are deprioritized, so use C to fulfill Order 1)

**Soft capacity nodes**: Similar to de-prioritized nodes, do not use over-capacity nodes unless necessary. Cannot use over-capacity to reduce splitting. (**EX5**: Nodes A, B, C have inventory for Order 1 for 2 SKUs s. A and B are over capacity, so use C to fulfill Order 1).

**Note 2:** do split if you need to use nodes over capacity to avoid splitting, which is equivalent to ”*Cannot use over-capacity to reduce splitting.”*

**EX6**: Node A has inventory of 3 units for SKU1. Node A has remaining capacity of 2. Node B has inventory of 3 units for SKU1. Node B is a deprioritized node. Node C has 1 unit of inventory for SKU1.

We have an order with request for 3 units of SKU1.

Expected Fulfillment: 3 units will be fulfilled from Node B.

**EX7**: Node A has inventory of 3 units for SKU1. Node A has remaining capacity of 2. Node B has inventory of 3 units for SKU1. Node B is a deprioritized node. Node C has 1 unit of inventory for SKU1. Node C is a force\_to node.

We have an order with request for 3 units of SKU1.

Expected Fulfillment: 2 units will be fulfilled from Node A, 1 unit to be fulfilled from node C.

**EX8**: Node A has inventory of 3 units for SKU1. Node A has remaining capacity of 2. Node B has inventory of 3 units for SKU1. Node B is a node with capacity of 1. Node C has 3 units of inventory for SKU1. Node C is a positive delay while Nodes A and B have zero delay.

We have an order with request for 3 units of SKU1.

Expected Fulfillment: 2 units will be fulfilled from Node A, 1 unit to be fulfilled from node B.

(**Note 3**: Fulfilling from a node with delay makes it the least desirable option)

**EX9**: Node A has inventory of 3 units for SKU1. Node A has remaining capacity of 2. Node B has inventory of 3 units for SKU1. Node B is a node with capacity of 1. Node C has 1 unit of orphan inventory for SKU1. Node C has a positive delay while Nodes A and B have zero delay.

We have an order with request for 3 units of SKU1.

Expected Fulfillment: 2 units will be fulfilled from Node A, 1 unit to be fulfilled from node B.

(**Note 4**: partially fulfilling from a node with orphan is favored even if it does cause an additional split)

**Minimize nodes per order**: After all of the above constraints, we should minimize the number of nodes per order

**EX10**: Node A has inventory of 1 orphan for SKU1. Node B has inventory of 1 orphan for SKU2. Node C has inventory of 1 orphan for SKU3. Node C has a positive delay. Node D has inventory of 10 for SKU1, inventory of 10 for SKU2, inventory of 10 for SKU3. Node E has inventory of 2 for SKU1, inventory of 2 for SKU2 and inventory of 2 for SKU3. Node E has much lower Shipping Cost and CO2 emission compared to Node D. The shipping costs and CO2 emissions of Node A, Node B, Node C are comparable to that of Node E.

We have an order with request for 5 units for SKU1, 5 units for SKU2 and 5 units for SKU3.

Expected Fulfillment: 1 unit of SKU1 from Node A, 1 unit of SKU2 from Node B, 2 units of SKU1 from Node D, 2 units of SKU2 from Node D, 3 units of SKU3 from Node D, 2 units of SKU1 from Node E, 2 units of SKU2 from Node E and 2 units of SKU3 from Node E. (**Note 5**: fulfilling from nodes with orphans take precedence over minimizing packages per order; fulfilling from nodes with delay makes fulfilling from that node the least desirable option even if it has an orphan)

**Question 1**: Hadi, I did not see the implementation of the orphans set of constraints in the new model. Did you add the sets of constraints regarding the priority treatment of orphans in your most recent code? Could you add the orphan-related constraints to the Confluence page? Notice my Note on **EX10**: *fulfilling from nodes with orphans take precedence over minimizing packages per order; fulfilling from nodes with delay makes fulfilling from that node the least desirable option even if it has an orphan.*

## Parameters

: the demand of SKU in order

: the capacity in node

: the available inventory of SKU in node

: the shipping cost of SKUs in order by carrier and service level to node

: the CO2 emission of SKUs in order by carrier and service level to node

: the delay of SKU in order by carrier and service level to node

: the capacity dual variable in node **Question 2**: Don’t we need a subscript for the order here as well?

: the inventory dual variable of SKU in order in node

: the number of SKUs in order

a parameter for EDD: TBD **Question 3**: Did you update your source code with this parameter? If yes could update the confluence page with it as well?

## Decision Variables

: the fulfilled volume of demand of SKU in order with service level by carrier from node

: the unfulfilled volume of demand of SKU in order

: a binary variable to model the capacity violation in node

: the violation of capacity in node

: a binary variable that is 1 if SKU in order is fulfilled by node ; 0 otherwise

## Constraints

**Demand** – the first constraint to be considered is the demand constraint, where the fulfilled and unfulfilled volumes of a SKU in each order should be equal to its corresponding demand:

(1)

**Force-to node** – once a node is set to be force-to node, we need to utilize its available inventory as much as possible for fulfilling the upcoming demands as it is the highest priority even if it causes split shipments. The condition is formulated as below:

The right-hand side of the above constraint specifies the maximum amount of demand of SKU across all orders that can be somehow potentially fulfilled from all force-nodes. In this case, we can make sure that we’re using up all the available inventory for each SKU in force-to nodes. If total demand of a SKU is greater than the available inventory of that SKU in all force-to nodes, i.e. , then the above constraint uses all inventory in force-to nodes for such SKU. The remaining demand is fulfilled by other types of nodes in the network. Otherwise, if , then force-to nodes fulfill all the demand for a given SKU . The exact fulfillment in these two conditions is specified by the optimization algorithm.

**Regular node** – once the available capacity in all force-to nodes is used up completely by means of the previous constraints, if there is still some demand unfulfilled, it should be allocated to regular nodes. The following constraint satisfies this condition:

It’s worth noting that the above constraint is enforced if and .

**Deprioritized node** – Ultimately, if there is still some demand unfulfilled by force-to and regular nodes, then the remaining demand should be fulfilled by deprioritized nodes, as follows:

It should be noted here that the above constraint is applied if and that . In other words, we make sure that there’s at least one deprioritized node in the network and some remaining unfulfilled demand.

**Capacity** – the capacity constraint is formulated in a soft manner, i.e., it’s allowed to violate this constraint in the program. The soft capacity constraint is given as below:

It should be noted in this constraint that if , then becomes positive to reflect the level of capacity violation. However, if , then must be zero in the lack of any violation. In other words, . This additional condition is formulated as follows:

// is indicator variable

**Inventory** – the total amount of SKU from all orders fulfilled by the node should be less than or equal to the corresponding available inventory which is formulated as follows:

**Split Shipment** – The purpose of this set of constraints is to reduce split shipment. The following constraints assign the SKUs in an order to the lower number of nodes.

// – number of order-lines in order ,

// – number of order-lines involving SKU fulfilled from node in order

// – slack variable for equality constraint conversion for SKU and node on order

// **Question 4**: What is the reason **not** to include as another term in the objective function?

**Variables domain** –

, , , , ,

// **Question 5**: Shouldn’t ?

## Objective Functions

**Cost** – the following objective function minimizes the total shipping cost, the total cost of consuming labors and inventory:

// **Question 6**: Hadi you have decided to include the dual variables as multiplicative factors instead of as simply additive terms in the objective function. I am not sure why you are introducing this change. Also, as we talk about dual variables, we should be subtracting those from the objective function instead of summing them – this negative sign also can be interpreted as a negative feedback which leads to a stable dynamic behavior. If we add them instead of subtracting, we would be implementing positive feedback which will lead to starvation of resources (in this case nodes).

In the current model, the capacity dual variable is a cumulative quantity representing the sum of the dual variables which were introduced in all orders before the current -th order.

For instance, let us assume that when processing the order a decision was made, which involved nodes with indices . The cost incurred on each node in this decision can be calculated as for each .

Let us denote this cost with . To that cost it corresponds its dual which has the semantics of revenue/reward – let us denote it with . Then the capacity dual variable of the online fulfillment problem at the time order is processed computed as: and it should be subtracted from the local objective function when processing order . Here is some tuning parameter which could introduce exponential decay or not. So, we will have: . The same argument applies for the inventory dual variables which also are cumulative quantities accumulated over the previous orders. Does this make sense to you, or I am missing something?

**CO2 Emissions** – The CO2 emission is minimized using the following objective function:

**Delay** – since the delay parameter, i.e., is positive if SKU has a level of delay, and that being early doesn’t have any value for us, the following objective minimizes the summation of delays (or equivalently the average of delay):

In other words, this objective function lead us to fulfillment solutions whose total delay is close to zero. Since fulfillment solutions with negative delay have no value for us (i.e., it suffices to be early, the amount of earliness does not matter), the corresponding delay is substituted by zero in the above objective function.

The min-max delay objective function can be linearized as follows: supposing that , then and . Therefore, the above delay objective function is rewritten as with the latter two inequalities added to the set of constraints.

// makes sense..

## The constraint method

In the developed multi-objective programming model, the goal is to provide a solution compromise among cost, CO2 emissions, and delay. The proposed model, in which cost, CO2 emission, and delay are minimized, can be summarized as follows:

Here is the vector of decision variables, TSC, TCO and DEL are cost, CO2 emissions and delay objective functions, respectively, and denotes the space of feasible solutions. In the constraint method, the original multi-objective problem is converted into a single objective programming model by retaining one of the objective functions as the primary objective function in the model. Accordingly, the remaining objective functions are expressed as the properly defined constraints with enforcing upper and/or lower bounds. Therefore, if delay objective function, i.e. is chosen as the primary objective function, the following single objective programming model is attained where cost and CO2 emission objective functions are treated as model constraints:

where , and . In order to derive the maximum and minimum values of cost and CO2 emission objective functions, the following steps are taken:

* Obtain the global optimum solution of each objective function over . Then, let are the obtained global optimum solutions corresponding to , and objective functions.
* Find the values of the objective functions and at each point of .
* Compute the minimum values of and objective functions as and
* . Compute the maximum values of and objective functions as and .

However, to guarantee an optimum solution of current single objective problem is a Pareto optimal solution of the original multi-objective problem, the modification proposed by (Mavrotas, 2009) is applied here, where the constraints associated with the added functions are transformed to equality by explicitly incorporating the appropriate slack variables and then penalizing the current new variables in the single objective function. Thus we have:

where is an adequately small value that does not affect the objective function.

# Bibliography

Mavrotas, G. (2009). Effective implementation of the ε-constraint method in Multi-objective Mathematical Programming Problems. *Applied Mathematics and Computation*, 455-465.

Relevant links:

*Generation of efficient solutions in Multiobjective Mathematical Programming problems using GAMS: Effective implementation of the ε-constraint method, George Mavrotas*: [here](https://nike.box.com/s/e9wryvk9e0pirlfnp5qcgx7tqud3ce7s)

*Effective implementation of the ε-constraint method for the generation of efficient solutions in Multi-objective Mathematical Programming problems, George Mavrotas, MCDA64, Larisa, 28-30 Sept, 2006:* [here](https://nike.box.com/s/lhx7xqyrz583mbdwuzd4erm6z02cwg7g)