Enterprise Wide Optimization Using Mixed Integer Linear Programming

May 31, 2018

Abstract

Supply chains whose decisions consist of inventory holding, production, transportation, sales, and materials acquisitions can be modeled and optimized for revenue using mixed integer linear programming (MILP). A "moving-horizon" model optimizes the revenue over the time-steps included within the horizon; possible and optimal decisions are limited by the system's state, which can be updated at each time-step, or horizon "shift". Many decisions are integer-constrained or binary, greatly increasing the computational cost of optimization, but the cost in the model presented is "minimal" in the sense that only valid decisions are allowed. Appropriate cutting planes can vastly decrease the cost of making binary-constrained decisions, even in high dimensional systems, and parallelization can divide the computational time-cost by the number of available processors. Model extensions related to specific types of supply chains are discussed.

1 Introdution

A supply chain whose decisions consist of inventory holding, production, transportation, sales, and the acquisition of raw materials at various price tiers can be optimized for profit using the following model. The number of decision variables scales approximately proportional to the product of the number of locations, the number of products, and the number of time-steps modeled.

Depending on the system being modeled, the number of price tiers available for acquisitions may be a coefficient to the total number of decision variables. The number of decisions that can be carried out at each location greatly influences the number of decision variables as well. If all or most locations can buy, sell, produce, hold, and ship almost all possible goods, the number of decisions will be several orders of magnitude greater than if most locations can only carry out one or two of those activities.

The model below includes an objective function linear in each decision variable, and each constraint is linear in each decision variable, making this model solvable by any mixed integer linear programming (MILP) algorithm.

The number of decision variables constrained to the integers is proportional to the total number of decision variables. Most integer-constrained decisions, however, are constrained to binary variables, which greatly limits the computational cost to explore the solution space. The computation is still expensive using a branch-and-bound linear programming algorithm, but the number of nodes that need to be explored is far less than in a general MILP due to the binary nature of the decisions. Branch-and-bound is highly parallelizable, meaning that the total time to compute can be divided by the number of processors available for computation, to a nearly arbitrary extent, as long as each processor is able to compute a (non-integer-constrained) linear program with a large number of variables. The optimal parallelization scheme probably would store the constraint matrix in a database that can be accessed by several nodes simultaneously, but each node would also need its own significant memory capacity to solve its linear programming sub-problem.

If all the following constraints are met, and the given objective function is optimized, then the system-wide revenue will be maximized, and the decision variables' values will correspond intuitively to their stated meanings.

This model amounts to a "moving horizon optimization" problem. In practice, an implementation will have to be parameterized one time to correspond to a given system, and then can be updated with the new system state at each time-step. If the optimization is truly driving the real system, then most changes made to the system at each time-step will simply correspond to the decisions made at the previous calculated time-step. Thus, an expeditious implementation of the below model will assume that previous decisions have been carried out, and will only required manual updating of the system-state when a decision *fails to be carried out*. That would mean that late arrivals of orders, incorrect shipments, and other inconsistencies will require manual system-state updating, but otherwise updating can be automatic. If decisions are consistently carried out differently from the model's planning, then the parameters of the system should be adjusted to reflect reality.

Uncertainty can be accommodated as discussed, but is not explicitly planned for in this model. To account for uncertainty, the same general treatment of system modeling can be used, but the objective function would need to reflect something different from deterministic revenue. The most obvious way to account for uncertainty in the given implementation is to optimize the model with slightly different parameters related to uncertain system characteristics, such as demand, production time, delivery time, and even costs. After running the optimization many times, the decisions that are optimal or near-optimal across a range of parameters could be considered "robust" to uncertainty.

Most MILP solvers can record any non-optimal, near-optimal, or general feasible solutions encountered during optimization. By recording near-optimal solutions, the robustness of certain decisions can be analyzed as discussed in the previous paragraph.

Decisions can be manually constrained to desired values or vetoed by a user, to accommodate needs that fall outside of revenue optimization. In the end, all decisions are made by the user, and an implementation and

user interface tailored to actual usage will be most appropriate.

2 Variables

2.1 System Scale Constants

Constant	Meaning	Example Entry = value	Example Meaning
n_z	Number of products in system	`nz = 5;`	5 products and components in system
n_t	Number of time-steps in system	`nt = 4;`	4 time-steps in system
n_{loc}	Number of locations in system	`nloc= 6;`	6 locations (factories, warehouses, retailers) in system
n_t^0	Number of initialized time-steps	`nt0 = 2;`	The first two time-steps are defined as initial conditions (decision variables hard-bound)
n_{tiers}	Maximum number of price-tiers for a raw material	`ntiers = 3;`	Some raw materials have 3 pricing tiers for acquisition
$dimTotal$	Total number of decision variables	`dimTotal = (4+nloc)*nz*nloc*nt + (nloc^2)*nt + nt*nloc;`	See: Decision Variables
$dimInt$	Total number of INTEGER-VALUED decision variables	`dimInt = nz*nloc*nt + (nloc^2)*nt + nt*nloc;`	See: Decision Variables

2.2 Parameters

Parameter	**Number Parameters**	**Element =**	**Explanation**
Demand	$n_z * n_t * n_{loc}$	$dem_{i,j,k} = c^w_{i,j,k} =$ `dem(i,j,k) =`	demand for product i at time-step j at location k
B.O.M.	$n_z * n_z$	$B_{i,I} =$ `Bom(i,I) =`	Number of units of item I DIRECTLY required to produce 1 unit item i
Time to Produce	$n_z * n_{loc}$	$R^z_{i,k} =$ `R_z(i,k) =`	Number of time-steps in production cycle for product i at location k
Time to Transport	n^2_{loc}	$R^t_{k,L} =$ `R_t(k,L) =`	Number of time-steps to send a truck from location k to location L
Product Space Requirements	n_z	$s_i =$ `s_y(i) =`	Number of "units" (of space, load, etc.) required to hold or truck one unit of product i
Capacity to Hold	n_{loc}	$c^y_k =$ `c_y(k) =`	Capacity (in "units") to hold at location k
Capacity to Truck	1	$c_{truck} =$ `c_truck =`	Capacity of one truck (in "units")
Capacity to Produce	$n_z * n_{loc}$	$c^z_{i,k} =$`c_z(i,k) =`	Capacity to produce product i at location k (in "number/amount produced")
Minimum Order Size	$n_z * n_{tiers}$	$c^{wm} =$`c_w(m,i) =`	Minimum order size to purchase SKU i at price tier m.
Cost at Sale	$n_z * n_t * n_{loc}$	$f^{ws}_{i,j,k} =$ `f_ws(i,j,k) =`	Revenue gained from a sale of one unit of product i at time-step j at location k.
Cost at Acquisition	$n_z * n_t * n_{loc} * n_{tiers}$	$f^{wm}_{i,j,k} =$ `f_w(i,j,k,m) =`	Cost to acquire SKU i at time-step j at location k at price tier m.
Cost to Produce (per-unit)	$n_z * n_t * n_{loc}$	$f^z_{i,j,k} =$ `f_z(i,j,k) =`	Cost to produce one unit of product i at time-step j at location k
Cost to Produce (fixed)	$n_z * n_t * n_{loc}$	$f^{z0}_{i,j,k} =$ `f_z0(i,j,k) =`	Fixed cost of running a production cycle of product i at time-step j at location k
Cost to Hold	$n_t * n_{loc}$	$f^y_{j,k} =$ `f_y(j,k) =`	Cost to hold one unit (in "units") at time-step j at location k
Cost to Truck by Route	n^2_{loc}	$f^t_{k,L} =$ `f_t(k,L) =`	Cost to send one truck from location k to location L (can easily make time-variable)
Characterization of Location for Sales	$n_z * n_{loc}$	$c^{w0}_{i,k} =$ `wc(i,k) =`	$\begin{cases} 1 & \text{Product } i \text{ can be sold at location } k \\ -1 & \text{Product } i \text{ can be bought at location } k \\ 0 & \text{otherwise} \end{cases}$
Characterization of Location for Production	$n_z * n_{loc}$	$c^{z0}_{i,k} =$ `zc(i,k) =`	$\begin{cases} 1 & \text{Product } i \text{ can be produced at location } k \\ 0 & \text{otherwise} \end{cases}$
Characterization of Route	n^2_{loc}	$c^{t0}_{k,L} =$`tc(k,L) =`	$\begin{cases} 1 & \text{Trucks can/do travel from location } k \text{ to location } L \\ 0 & \text{otherwise} \end{cases}$
Initial Conditions	$n_z * n^0_t * n_{loc} +$ ...	`init(n,i,j,k,L) =`	Defined for time-steps $j \leq n^0_t$, n is the decision index, and i, j, k, and L may or may not all be used to index to product/time/location/location whose initial conditions are hard-set

2.3 Decision Variables

Variable	Indexed in decision vector =	Meaning	Max Number of Variables	Lower Bound	Upper Bound
$y_{i,j,k}$	`x(J(1,i,j,k,1))` =	inventory of product i at location k at time-step j	$n_z * n_t * n_{loc}$	0	c^y_k, ∞^*
$z_{i,j,k}$	`x(J(2,i,j,k,1))` =	amount produced of product i at location k at time-step j	$n_z * n_t * n_{loc}$	0	$c^z_{i,k}, \infty^*$
$t_{i,j,k,l}$	`x(J(3,i,j,k,L))` =	amount of product i transferred from location k to location l at time-step j	$n_z * n_t * n^2_{loc}$	0	∞
$w^s_{i,j,k}$	`x(J(4,i,j,k,1))` =	amount of product i SOLD at location k at time-step j	$n_z * n_t * n_{loc}$	$-\infty, 0^{**}$	$0, \infty^{**}$
$z^0_{i,j,k}$	`x(J(5,i,j,k,1))` =	$\begin{cases} 1 & \text{Machine at location } k \text{ goes into production for product } i \text{ at time-step } j \\ 0 & \text{otherwise} \end{cases}$	$n_z * n_t * n_{loc}$	0	1
$t^0_{j,k,l}$	`x(J(6,1,j,k,L))` =	(INTEGER ≥ 0) Number of trucks sent from location k to location l at time-step j	$n_t * n^2_{loc}$	0	∞
$z^{00}_{j,k}$	`x(J(7,1,j,k,1))` =	$\begin{cases} 1 & \text{Machine at location } k \text{ is "in-production" at time-step } j \\ 0 & \text{otherwise} \end{cases}$	$n_t * n_{loc}$	0	1
$w^m_{i,j,k}$	`x(J(8,i,j,k,m))` =	amount of product i BOUGHT at location k at time-step j at price-tier m	$n_z * n_t * n_{loc} * n_{tiers}$	0	∞
$w^{m0}_{i,j,k}$	`x(J(9,i,j,k,m))` =	$\begin{cases} 1 & \text{Product } i \text{ is purchased at tier } m \text{ at time-step } j \text{ at location } k \\ 0 & \text{otherwise} \end{cases}$	$n_z * n_t * n_{loc} * n_{tiers}$	0	1

*Note that where two bounds are given, the more restrictive bound may be included to accelerate optimization, but the theoretical optimization problem will be satisfied given the less restrictive constraint. This is usually because one of the constraints will limit the given variable under any circumstances.

**Note that decision variables corresponding to the sale of product at a given location are non-negative and unbounded above, whereas decision variables corresponding to the purchase of product/component at a given location are non-positive and unbounded below.

3 Formal Optimization Problem

If the following function is maximized given all the listed constraints, then all the decision variables will correspond to possible decisions at each time-step. **Objective:**

$$Maximize \sum_{i,j,k} f^{ws}_i w^s_{i,j,k} - \sum_{i,j,k,m} f^{wm}_i w^m_{i,j,k} - \sum_{j,l,k} f^t_{l,k} t^0_{j,l,k} - \sum_{i,j,k} f^z_{i,k} z_{i,j,k} - \sum_{i,j,k} f^{z0}_{i,k} z^0_{i,j,k} - \sum_{i,j,k} f^y_k s_i y_{i,j,k} \tag{10}$$

SUBJECT TO:
Evolution:

$$y_{i,j,k} = y_{i,(j-1),k} + z_{i,(j-R^z_{i,k}),k} - \sum_I z_{I,(j-1),k} B_{I,i} + \sum_l t_{i,(j-R^t_{l,k}),l,k} - \sum_l t_{i,(j-1),k,l} - w^s_{i,(j-1),k} + \sum_m w^m_{i,(j-1),k} \tag{1}$$

$$\forall i \in [1, n_z], j \in [2, n_t], k \in [1, n_{loc}]$$

Final Time-Step Evolution:

$$w^s_{i,n_t,k} + \sum_I z_{I,n_t,k} B_{I,i} + \sum_l t_{i,n_t,k,l} \leq y_{i,n_t,k} \; \forall \, i \in [1, n_z], k \in [1, n_{loc}] \quad (2)$$

Inventory Holding:

$$\sum_i s_i y_{i,j,k} \leq c^y_k \; \forall \; j \in [1, n_t], k \in [1, n_{loc}] \quad (3)$$

Transportation:

$$\sum_i s_i t_{i,j,k,l} \leq c_{truck} t^0_{j,k,l} \; \forall \; j \in [1, n_t], k \in [1, n_{loc}], l \in [1, n_{loc}] \quad (4)$$

Sales:

$$w^s_{i,j,k} \leq dem_{i,j,k} \; \forall \; i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}] \quad (12)$$

Acquisitions:

$$c^{wm}_i w^{m0}_{i,j,k} \leq w^m_{i,j,k} \leq \infty w^{m0}_{i,j,k} \; \forall \, i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}], m \in [1, n_{tiers}] \quad (6)$$

Production Capacity:

$$z_{i,j,k} \leq c^z_{i,k} z^0_{i,j,k} \; \forall \; i \in [1, n_z], j \in [1, nt], k \in [1, n_{loc}] \quad (7)$$

Production Site Occupation:

$$z^{00}_{j,k} + \sum_i z^0_{i,j,k} \leq 1 \; \forall \; j \in [1, nt], k \in [1, n_{loc}] \quad (8)$$

Production Site PRE-Occupation:

$$\underbrace{z^{00}_{j,k}}_{0 \text{ or } 1} \geq \sum_{i=1}^{n_z} \sum_{\tau=1}^{R_z(i,k)-1} \underbrace{z^0_{i,j-\tau,k}}_{0 \text{ or } 1} \quad \forall \quad j \in [1, n_t], k \in [1, n_{loc}] \quad (9)$$

3.1 Evolution Equation: $n_z * n_t * n_{loc}$ equality constraints

$$y_{i,j,k} = y_{i,(j-1),k} + z_{i,(j-R^z_{i,k}),k} - \sum_I z_{I,(j-1),k} B_{I,i} + \sum_l t_{i,(j-R^t_{l,k}),l,k} - \sum_l t_{i,(j-1),k,l} - w^s_{i,(j-1),k} + \sum_m w^m_{i,(j-1),k} \quad (1)$$

$$\forall i \in [1, n_z], j \in [2, n_t], k \in [1, n_{loc}]$$

Explanation:

$$\underbrace{y_{i,j,k}}_{\text{A}} = \underbrace{y_{i,(j-1),k}}_{\text{B}} + \underbrace{z_{i,(j-R^z_{i,k}),k}}_{\text{C}} - \sum_I \underbrace{z_{I,(j-1),k} B_{I,i}}_{\text{D}} + \sum_l \underbrace{t_{i,(j-R^t_{l,k}),l,k}}_{\text{E}} - \sum_l \underbrace{t_{i,(j-1),k,l}}_{\text{F}} - \underbrace{w^s_{i,(j-1),k}}_{\text{G}} + \sum_m \underbrace{w^m_{i,(j-1),k}}_{\text{H}}$$

$$\underbrace{\forall i \in [1, n_z], j \in [2, n_t], k \in [1, n_{loc}]}_{n_z * n_t * n_{loc} \text{ possible indices}}$$

Equation Label	Term	Explanation
A	$y_{i,j,k}$	Inventory of product i at time-step j at location k
B	$y_{i,(j-1),k}$	Inventory of product i **held over** from time-step $(j-1)$ at location k
C	$z_{i,(j-R^z_{i,k}),k}$	$R^z_{i,k}$ is the number of time-steps required to produce product i at location k. Therefore, $z_{i,(j-R^z_{i,k}),k}$ denotes the quantity of product i that is **coming out of production at time-step** j
D	$z_{I,(j-1),k}B_{I,i}$	$B_{I,i}$ is the number of units of product/component i required to produce one unit of product I, and $z_{I,(j-1),k}$ denotes the number of units of product I produced in the $(j-1)$'th time-step. Therefore, $z_{I,(j-1),k}B_{I,i}$ denotes how many units of product/component i **went into production** for product I at time-step $(j-1)$.
E	$t_{i,(j-R^t_{l,k}),l,k}$	$R^t_{l,k}$ is the number of time-steps required to transport a truck from location l to location k. Therefore, $t_{i,(j-R^t_{l,k}),l,k}$ denotes the number of units **transported to** location k from location l that arrive at time-step j.
F	$t_{i,(j-1),k,l}$	The quantity of product i **transported from** location k to location l in time-step $(j-1)$
G	$w^s_{i,(j-1),k}$	The quantity of product i **sold** at location k during time-step $(j-1)$.
H	$w^M_{i,(j-1),k}$	The quantity of product i **acquired** at location k during time-step $(j-1)$ at price tier m.

3.2 Final-Timestep Evolution: $n_z * n_{loc}$ constraints

$$w_{i,n_t,k} + \sum_I z_{I,n_t,k}B_{I,i} + \sum_l t_{i,n_t,k,l} \leq y_{i,n_t,k} \; \forall \; i \in [1,n_z], k \in [1,n_{loc}] \quad (2)$$

Explanation:

$$w_{i,n_t,k} + \underbrace{\sum_I z_{I,n_t,k}B_{I,i}}_{\text{Amount of product } i \text{ used for production at location } k \text{ in time-step } n_t} + \underbrace{\sum_l t_{i,n_t,k,l}}_{\text{Amount of product } i \text{ shipped from location } k \text{ in time-step } n_t} \leq y_{i,n_t,k} \; \forall \; \underbrace{i \in [1,n_z], k \in [1,n_{loc}]}_{n_z * n_{loc} \text{ possible indices}}$$

The evolution equation ensures that all goods present at a given location in a given time-step are either sold, transported away, used for production, or held for the next time-step. That equation also allows for goods to be added to the location through the corresponding decisions. Another way to consider the evolution equation, however, is that the decisions in a given time-step determine the held-over inventory for the *next* time-step. For this reason, the evolution equation cannot govern the final time-step included in the model. In the final time-step, the decisions cannot be constrained to result in the held-over inventory of the next time-step, but rather than leave them un-constrained, they must be constrained to involve only available inventory. Un-constrained, the final time-step's decisions to sell would be maximized to demand, possibly beyond the inventory available at that time-step.

3.3 Holding: $n_t * n_{loc}$ constraints

$$\sum_i s_i y_{i,j,k} \leq c_k^y \; \forall \; j \in [1, n_t], k \in [1, n_{loc}] \tag{3}$$

Explanation:

$$\overbrace{\sum_i \underbrace{s_i}_{\substack{\text{Space taken up} \\ \text{by product } i}} \underbrace{y_{i,j,k}}_{\substack{\text{quantity of} \\ \text{product } i \text{ held} \\ \text{at time-step } j \\ \text{at location } k}}}^{\substack{\text{Total space taken up} \\ \text{by inventory at time-} \\ \text{step } j \text{ at location } k}} \leq \underbrace{c_k^y}_{\substack{\text{Capacity to} \\ \text{hold inventory} \\ \text{at location } k}} \quad \forall \underbrace{j \in [1, n_t], k \in [1, n_{loc}]}_{n_t * n_{loc} \text{ possible indices}}$$

This constraint ensures that the quantity of inventory held at a given location at a given time-step does not not exceed that location's capacity to hold inventory. Note that the "space taken up by product i" s_i is measured in "units", which can refer to volume, load, etc. If several different types of capacities to hold products need to be taken into account, several capacity constraints can be included as s_i^n, and an additional set of equally many constraints would be required for each type of capacity.

3.4 Transportation: $n_t * n_{loc}^2$ constraints

$$\sum_i s_i t_{i,j,k,l} \leq c_{truck} t_{j,k,l}^0 \; \forall \; j \in [1, n_t], k \in [1, n_{loc}], l \in [1, n_{loc}] \tag{4}$$

Explanation:

$$\overbrace{\sum_i \underbrace{s_i}_{\substack{\text{Space taken up} \\ \text{by product } i}} \underbrace{t_{i,j,k,l}}_{\substack{\text{Quantity of} \\ \text{product } i \\ \text{shipped from} \\ \text{location } k \text{ to} \\ \text{location } l \text{ at} \\ \text{time-step } j}}}^{\substack{\text{Total quantity of in-} \\ \text{ventory shipped from} \\ \text{location } k \text{ to location } l \\ \text{at time-step } j}} \leq \underbrace{c_{truck}}_{\substack{\text{Capacity of one} \\ \text{truck to hold in-} \\ \text{ventory}}} \underbrace{t_{j,k,l}^0}_{\substack{\text{Number of} \\ \text{trucks sent} \\ \text{from location } k \\ \text{to location } l \text{ at} \\ \text{time-step } j}} \quad \forall \underbrace{j \in [1, n_t], k \in [1, n_{loc}], l \in [1, n_{loc}]}_{n_t * n_{loc}^2 \text{ possible indices}}$$

Note that the number of trucks sent from location k to location l in time-step j, $t_{j,k,l}^0$, is constrained to be a non-negative integer, meaning that it is denoted an integer, and its lower and upper bounds are 0 and ∞, respectively.

3.5 Sales: $n_z * n_t * n_{loc}$ constraints

$$w^s_{i,j,k} \leq dem_{i,j,k} \; \forall \; i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}] \tag{5}$$

Explanation:

$$\underbrace{w^s_{i,j,k}}_{\substack{\text{quantity of product } i \text{ sold at time-step } j \text{ at location } k}} \leq \underbrace{dem_{i,j,k}}_{\substack{\text{Demand for product } i \text{ at time-step } j \text{ at location } k}} \quad \forall \; \underbrace{i \in [1, n_z], j \in [1, nt], k \in [1, n_{loc}]}_{n_z * n_t * n_{loc} \text{ possible indices}}$$

3.6 Acquisitions: $n_z * n_t * n_{loc} * n_{tiers}$ constraints

$$c_i^{wm} w^{m0}_{i,j,k} \leq w^m_{i,j,k} \leq \infty w^{m0}_{i,j,k} \; \forall \, i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}], m \in [1, n_{tiers}] \tag{6}$$

Explanation:

$$\underbrace{c_i^{wm} w^{m0}_{i,j,k}}_{\substack{\text{Product } i \text{ purchased at tier } m \text{ at time-step } j \text{ at location } k}} \leq w^m_{i,j,k} \leq \underbrace{\infty w^{m0}_{i,j,k}}_{\substack{\text{Product } i \text{ purchased at tier } m \text{ at time-step } j \text{ at location } k}} \quad \forall \; \underbrace{i \in [1, n_z], j \in [1, nt], k \in [1, n_{loc}], m \in [1, n_{tiers}]}_{n_z * n_t * n_{loc} * n_{tiers} \text{ possible indices}}$$

$$= \begin{cases} c_i^{wm} & : \\ 0 & : \end{cases} \qquad = \begin{cases} \infty & : \\ 0 & : \end{cases}$$

Note that the binary decision variable $w^{m0}_{i,j,k} \in \{0, 1\}$ represents the binary decision to either purchase SKU i at price tier m at time-step j at location k or not.

If $w^{m0}_{i,j,k} = 0$, then this equation states that

$$0 \leq w^m_{i,j,k} \leq 0,$$

or simply

$$w^m_{i,j,k} = 0,$$

meaning that *none* of SKU i is purchase at time-step j at location k at price-tier m, as desired.

Alternatively, if $w^{m0}_{i,j,k} = 1$, then this equation states that

$$c_i^{wm} \leq w^m_{i,j,k} \leq \infty,$$

which means that the minimum acquisition is c_i^{wm}, which is the minimum order size for product i at price tier m, as desired. The maximum order size is defined as infinity, because the upper bound of $w^m_{i,j,k}$ already dictates that orders cannot exceed the holding capacity of their destination.

Purchasing at the highest possible price tier will allow for equal acquisition with the least penalty to the objective function. Thus, there is no need to restrict that the maximum order size at a given tier be less than or equal to the minimum order size for the subsequently higher tier. If an order size were to exceed the minimum order size at a higher tier, the decision variable w^{m0} corresponding to the higher tier would be nonzero due to optimality.

3.7 Production Capacity: $n_z * n_t * n_{loc}$ constraints

$$z_{i,j,k} \leq c^z_{i,k} z^0_{i,j,k} \ \forall \ i \in [1, n_z], j \in [1, nt], k \in [1, n_{loc}] \tag{7}$$

Explanation:

$$\underbrace{z_{i,j,k}}_{\substack{\text{Quantity of product } i \\ \text{produce in time-step } j \\ \text{at location } k}} \leq \underbrace{c^z_{i,k}}_{\substack{\text{Capacity to produce} \\ \text{product } i \text{ at location } k}} \underbrace{z^0_{i,j,k}}_{=\begin{cases} 1 & : \text{ product } i \text{ produced in time-step } j \text{ at location } k \\ 0 & : \text{ otherwise} \end{cases}} \forall \underbrace{i \in [1, n_z], j \in [1, nt], k \in [1, n_{loc}]}_{n_z * n_t * n_{loc} \text{ possible indices}}$$

This equation states that if $z^0_{i,j,k} = 1$, then the quantity of product i produced at time-step j at location k is less than or equal to $c^z_{i,k}$, which is the production capacity for product i at location k.

If, on the other hand, $z^0_{i,j,k} = 0$, then the quantity of product i produced at time-step j at location$k$ is zero.

Note that $z^0_{i,j,k} \in \{0, 1\} \ \forall i, j, k$, no matter what because those decision variables are constrained to be integers, and their lower and upper bounds are 0 and 1, respectively.

3.8 Production Site Occupation: $n_t * n_{loc}$ constraints

$$z^{00}_{j,k} + \sum_i z^0_{i,j,k} \leq 1 \ \forall \ j \in [1, nt], k \in [1, n_{loc}] \tag{8}$$

Explanation:

$$\underbrace{z^{00}_{j,k}}_{=\begin{cases} 1 & : \text{ location } k \text{ is "in-production" at time-step } j \\ 0 & : \text{ otherwise} \end{cases}} + \sum_i \underbrace{z^0_{i,j,k}}_{=\begin{cases} 1 & : \text{ product } i \text{ produced starting in time-step } j \text{ at location } k \\ 0 & : \text{ otherwise} \end{cases}} \leq 1 \ \forall \underbrace{j \in [1, nt], k \in [1, n_{loc}]}_{n_t * n_{loc} \text{ possible indices}}$$

This inequality ensures that the sum of all these binary decision variables is at most 1, meaning that **at most one of these variables can be equal to** 1. That means that, for a given time-step j and location k, there is one of three possibilities:

- No production starts, nor is any product "in production". "No production starts" means that for all products i, $z^0_{i,j,k} = z_{i,j,k} = 0$, and "no product is in-production" means that $z^{00}_{j,k} = 0$.
- Production "starts" on product i. This means that $z^0_{i,j,k} = 1$, and, in this case, this constraint ensures that no other products go into production simultaneously. It also ensures that $z^{00}_{j,k} = 0$, which means that the factory is not "in-production" for anything, but has just "started" production.
- Product i is "in-production". In this case, $z^{00}_{j,k} = 1$ is ensured by equation (9), regardless of which product i. In this case, this constraint ensures that no other products go into production at this time-step and location.

3.9 Production Site PRE-Occupation: $n_t * n_{loc}$ constraints

$$\underbrace{z^{00}_{j,k}}_{0 \text{ or } 1} \geq \sum_{i=1}^{n_z} \sum_{\tau=1}^{R_z(i,k)-1} \underbrace{z^{0}_{i,j-\tau,k}}_{0 \text{ or } 1} \quad \forall \quad j \in [1, n_t], k \in [1, n_{loc}] \tag{9}$$

which is equivalent to

$$\underbrace{z^{00}_{j,k}}_{0 \text{ or } 1} \geq \sum_{i=1}^{n_z} \sum_{\tau=j-R_z(i,k)+1}^{j-1} \underbrace{z^{0}_{i,\tau,k}}_{0 \text{ or } 1} \quad \forall \quad \underbrace{j \in [1, n_t], k \in [1, n_{loc}]}_{n_t * n_{loc} \text{ possible indices}}.$$

This ensures

$$z^{00}_{j,k} = \begin{cases} 1 & \text{Machine at location } k \text{ is} \\ & \text{“in-production” at time-step } j \\ 0 & \text{otherwise} \end{cases}$$

given

$$z^{0}_{i,j,k} = \begin{cases} & \text{Machine at location } k \\ 1 & \text{goes into production for} \\ & \text{product } i \text{ at time-step } j \\ 0 & \text{otherwise} \end{cases}$$

$R^z_{i,k}$ = Number of time-steps in production cycle for product i at location k (an integer ≥ 1)

This constraint demands that no product goes into production while another product's production cycle is ongoing.

Suppose that at a given location k, product I went into production at time-step $j - R^z_{I,k}$. By definition, its production cycle extends from time-step $j - R^z_{I,k}$ to time-step j, with j being the final time-step during the production cycle. We would like to say

$$\begin{aligned} z^{00}_{j-R^z_{I,k}+1,k} &= 1 \\ z^{00}_{j-R^z_{I,k}+2,k} &= 1 \\ \vdots \qquad & \quad \vdots \\ z^{00}_{j-1,k} &= 1 \\ z^{00}_{j,k} &= 1 \end{aligned}$$

Since these decision variables are bounded above by 1, it is sufficient to show that $z^{00}_{\tau,k} \geq 1$ for the appropriate indices τ.

Now let

$$t = j - R^z_{I,k} + 1,$$

which is one time-step AFTER product I went into production at location k, and consider $z^{00}_{t,k}$, which is the “in production” variable at the time-step t. Using this change of variable, we want to prove

$$
\begin{aligned}
z^{00}_{t,k} &= 1 \\
z^{00}_{t+1,k} &= 1 \\
\vdots &\quad \vdots \\
z^{00}_{j-1,k} &= 1 \\
z^{00}_{j,k} &= 1
\end{aligned}
$$

Product I went into production at time-step $t-1$, so we know

$$
\begin{aligned}
z^{0}_{I,t-1,k} &= 1 \\
&\text{and} \\
z^{0}_{i,t-1,k} = z_{i,t-1,k} &= 0 \;\forall\; i \neq I
\end{aligned}
$$

due to the production capacity constraint (7), and the production site occupation constraint (8), respectively.

Since we know the production-time for product I at location k, $R_z(I,k) \geq 1$, and since all these decision variables are non-negative, we can confidently say that

$$
\begin{aligned}
z^{00}_{t,k} &\geq \sum_i \sum_{\tau=t-R_z(i,k)}^{t-1} \underbrace{z^{0}_{i,\tau,k}}_{0 \text{ or } 1} \\
&\geq \sum_{\tau=t-R_z(I,k)}^{t-1} \underbrace{z^{0}_{I,\tau,k}}_{0 \text{ or } 1} \\
&\geq \underbrace{z^{0}_{I,t-1,k}}_{\text{The final term in summation}} \\
&= 1.
\end{aligned}
$$

Now consider time-step $t+1$, and note

$$
\begin{aligned}
z^{00}_{t+1,k} &\geq \sum_i \sum_{\tau=t-R_z(i,k)+1}^{t} \underbrace{z^{0}_{i,\tau,k}}_{0 \text{ or } 1} \\
&\geq \sum_{\tau=t-R_z(I,k)+1}^{t} \underbrace{z^{0}_{I,\tau,k}}_{0 \text{ or } 1} \\
&\geq \underbrace{z^{0}_{I,t-1,k}}_{\text{The second-to-last term in summation}} \\
&= 1.
\end{aligned}
$$

This logic holds until the first time-step at which $z^{0}_{I,t-1,k}$ is not included in the given summation, which is the time-step $t + R^{z}_{I,k}$, at which

point we notice

$$z^{00}_{t+R^z_{I,k},k} \geq \sum_i \sum_{\tau=t-R_z(i,k)+R^z_{I,k}}^{t+R^z_{I,k}} \underbrace{z^0_{i,\tau,k}}_{\text{0 or 1}}$$
$$= \sum_i \sum_{\tau=t}^{t+R^z_{I,k}} \underbrace{z^0_{I,\tau,k}}_{\text{Doesn't include } z^0_{I,t-1,k}}.$$

By examining the indices, we notice we've proved

$$\begin{aligned} z^{00}_{t,k} &= 1 \\ z^{00}_{t+1,k} &= 1 \\ \vdots &\quad \vdots \\ z^{00}_{t+R^z_{I,k}-1} &= 1 \end{aligned}$$

and since $t = j - R^z_{I,k} + 1$, we've shown

$$\begin{aligned} z^{00}_{t,k} &= 1 \\ z^{00}_{t+1,k} &= 1 \\ \vdots &\quad \vdots \\ z^{00}_{j,k} &= 1. \end{aligned}$$

Note that it is useful to constrain equality here, because while there is no *cost* to letting $z^{00}_{j,k} = 1$ outside of a production cycle, and such a decision can be easily disregarded, the correspondence of this model to the system which it models would necessitate $z^{00}_{j,k} = 0$ whenever production is *not happening* at location k at time-step j. This constraint would become:

$$\underbrace{z^{00}_{j,k}}_{\text{0 or 1}} = \sum_{i=1}^{n_z} \sum_{\tau=1}^{R_z(i,k)-1} \underbrace{z^0_{i,j-\tau,k}}_{\text{0 or 1}} \quad \forall \;\; j \in [1, n_t], k \in [1, n_{loc}].$$

3.10 Objective Function

$$Maximize \sum_{i,j,k} f^{ws}_i w^s_{i,j,k} - \sum_{i,j,k,m} f^{wm}_i w^m_{i,j,k} - \sum_{j,l,k} f^t_{l,k} t^0_{j,l,k} - \sum_{i,j,k} f^z_{i,k} z_{i,j,k} - \sum_{i,j,k} f^{z0}_{i,k} z^0_{i,j,k} - \sum_{i,j,k} f^y_k s_i y_{i,j,k} \tag{10}$$

Explanation:

$$Maximize \overbrace{\sum_{i,j,k} \underbrace{f^{ws}_i w^s_{i,j,k}}_{\text{A}} - \sum_{j,l,k} \underbrace{f^t_{l,k} t^0_{j,l,k}}_{\text{B}} - \sum_{i,j,k} \underbrace{f^z_{i,k} z_{i,j,k}}_{\text{C}} - \sum_{i,j,k} \underbrace{f^{z0}_{i,k} z^0_{i,j,k}}_{\text{D}} - \sum_{i,j,k} \underbrace{f^y_k s_i y_{i,j,k}}_{\text{E}} - \sum_{i,j,k,m} \underbrace{f^{wm}_i w^m_{i,j,k}}_{\text{F}}}^{\text{G}}$$

Equation Label	**Term**	**Explanation**
A	$f_i^{ws} w_{i,j,k}^s$	The parameter f_i^{ws} denotes the sales price of one unit of product i. The decision $w_{i,j,k}^s$ denotes how much of product i is sold at time-step j at location k. The product of these two terms is the amount of revenue accrued by selling product i at time-step j at location k.
B	$f_{l,k}^t t_{j,l,k}^0$	The parameter $f_{l,k}^t$ denotes the cost of sending a truck from location l to k, and the decision variable $t_{j,l,k}^0$ denotes the number of trucks sent on that route at time-step j. The product of these two terms is the money spent trucking along that route at time-step j.
C	$f_{i,k}^z z_{i,j,k}$	The parameter $f_{i,k}^z$ denotes the cost-per-unit of producing product i at location k, and the decision variable $z_{i,j,k}$ denotes how much of product i goes into production at time-step j at location k. The product of these two terms is the variable cost expended on producing product i at time-step j at location k.
D	$f_{i,k}^{z0} z_{i,j,k}^0$	The parameter $f^{z0})i,k$ denotes the fixed-cost of a production cycle for product i at location k. The binary ($\in \{0,1\}$) decision variable $z_{i,j,k}^0$ denotes whether or not a production cycle for product i begins at time-step j at location k. The product of these terms is either zero or $f_{i,k}^{z0}$, and is equal to the fixed-cost of producing product i expended at time-step j at location k. Note that fixed costs are considered to be accrued instantaneously at the beginning of a production cycle (arbitrary re: accounting).
E	$f_k^y s_i y_{i,j,k}$	The parameter f_k^y denotes the cost per-time-step of holding one "unit" of inventory at location k. The decision variable $y_{i,j,k}$ denotes how much of product i is held at time-step j at location k. The product of these terms is the cost expended for holding product i at time-step j at location k.
F	$f_i^{wm} w_{i,j,k}^m$	The parameter f_i^{wm} denotes the cost to purchase one unit of product i at price tier m. The decision variable $w_{i,j,k}^m$ denotes how much of product i is purchased at time-step j at location k at price tier m. The product of these terms is the cost expended for purchasing product i at time-step j at location k at tier m.
G	Equation	This equation describes the "profit" over all the time-steps considered. That is, for each time-step, the revenue minus the purchasing costs, minus the transportation costs, minus the variable/proportional production costs, minus the fixed production costs, minus the holding costs is considered the profit, and the revenue over each time-step is added. This may be more easily visualized given the presentation given in (11).

$$Maximize:$$

$$\overbrace{\sum_{j}\left(\underbrace{\sum_{i,k} f_i^w w_{i,j,k}}_{\text{Revenue/Sales Costs at time-step } j} - \underbrace{\sum_{l,k} f_{l,k}^t t_{j,l,k}^0}_{\text{Transport costs at time-step } j} - \underbrace{\sum_{i,k} f_{i,k}^z z_{i,j,k}}_{\text{Proportional production costs at time-step } j} - \underbrace{\sum_{i,k} f_{i,k}^{z0} z_{i,j,k}^0}_{\text{Fixed production costs at time-step } j} - \underbrace{\sum_{i,k} f_k^y s_i y_{i,j,k}}_{\text{Holding costs at time-step } j}\right)}^{\text{Profit at time-step } j} \tag{11}$$

4 Initial Conditions

The initial conditions can define the values of all the decision variables for time-steps that have ALREADY passed, so that the optimization begins "during" system operations.

Decisions for ANY time-step, though, can be hard-set. There are other ways to ensure orders come through, such as by adding constraints that ensure the sum of production quantities exceeds a given quantity by order date, or by ensuring inventory exceeds the order amount, etc. There are many ways to meet additional constraints. However, it's worth noting that, if external effort is put into determining some decisions, the corresponding decision variables can easily be hard-set to the values determined to be optimal.

5 Indexing Decision Variables (using the matrix J)

All decision variables will be stored in a vector after $MATLAB$ optimizes. The index of the decision variables in this final vector may be unclear, considering that the decision variables all span multiple indices, and are most intuitively stored in 3- or 4-dimensional arrays. The most straightforward way to find the optimal values in $MATLAB$'s output is to create an "indexing matrix", which I've called J.

The first dimension of J refers to the type of decision variable being referenced. That is, all entries of the form `J(1,i,j,k,L)` are, by arbitrary convention, the indices of *holding* decision variables of the form $y_{i,j,k}$. Since the indices of the holding variables run over 3 indices, the fifth dimension of matrix J is unused, and, when the first component of the index in J is 1, the fifth component must be 1 as well.

The indices of decision variables in the decision vector are described in section 2.3.

6 Pruning Unused Variables

Not every location produces every product. Not every location sells or buys every product. In fact, since a "location" can be a retailer, a warehouse, or a production site, it's quite likely that *most* product-location pairs correspond to either sales, production, or neither. This can be described in the system by constraining the decision variables corresponding irrelevant decisions (production or sales of a given product at a given location (at every time-step)) to be zero. However, decreasing the overall number of decision variables is desirable given the problem's computational complexity. Therefore, it is more useful to simply remove irrelevant decision variables from the problem entirely.

This may produce an indexing headache in implementation, but it is conceptually very simple: consider those variables as simply non-existent, and whenever they are referenced in a summation over their corresponding indices, consider nothing to be referenced.

Transportation variables (continuous and integer-valued) corresponding to transportation routes that are un-travelled can also be "pruned".

In this way, the dimension of the problem (number of decision variables) can be vastly reduced from $dimTotal$ to what can be called $dimFinal$.

7 Extensions

Possible extensions of this technique, to include more functionality or make more efficient the processing of current goals.

7.1 Stochasticity and Fully Time-Varying Parameters

Note that in the objective function (11), each decision variable's coefficient has a subscript j corresponding to the time-step in which the decision is being made. That is, all the parameters associated with costs, revenues, and actions in this model are time-varying. As a result, the effects of changes in these system parameters can be investigated using this model.

Even time-varying parameters are *not* decisions made within this model. On one hand, this means that a system parameter can be modeled as varying arbitrarily from time-step to time-step without adding computational complexity to a solution, barring possible identification of symmetries within the model by a sophisticated solver. On the other hand, this requires separate analysis to choose time-varying parameters that are within the user's control, such as price.

All parameters may follow a deterministic or stochastic relationship, and can be adjusted according to a separate model, such as a price-demand curve. Time-varying parameters are no different, and so this supply-chain model can accommodate arbitrarily complex time-varying relationships between parameters, limited only by the granularity of the time-steps.

Uncertainty and stochasticity are not included in this model as such, and parameters are modeled as "hard-set" and "known", whether time-varying or static. If parameters vary stochastically, a user can sample

parameters throughout their domains and generate optimal solutions for each. This standard technique for stochastic optimization can describe the stability of solutions, find regions of bifurcation in parameter space, and give a user a more descriptive idea of what to do next. Time-varying parameters give many more degrees of freedom in modeling stochasticity, but the idea is the same.

7.2 Backlog in Demand

If demand in a given time-step is not met, does the demand for the product in a subsequent next time-step understate the total consumer need/desire for that good? Does demand "roll over" to the next time-step?

Suppose demand *did not* roll over. Then, un-met demand would represent a loss in revenue, and optimization would meet as much of that demand as possible in every time-step. Unfortunately, if an optimal production schedule failed to meet demand at a given time-step, no further action would be taken by the optimization scheduler to increase supply at subsequent time-steps.

While meeting demand at specified time-steps is important, so is adjusting to production limitations.

The current demand-fulfilling constraint in this model enforces that the amount of a given product sold at a given time-step (at a given location) is less than or equal to the demand at that time-step (at that location):

$$w_{i,j,k} \leq \underbrace{c^w_{i,j,k}}_{dem_{i,j,k}} \quad \forall\ i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}]. \tag{12}$$

To allow the system to fulfill *all* backlogged demand at full price, we can allow the more lenient constraint

$$\sum_{\tau \leq j} w_{i,\tau,k} \leq \sum_{\tau \leq j} \underbrace{c^w_{i,\tau,k}}_{dem_{i,\tau,k}} \quad \forall\ i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}]. \tag{13}$$

It is likely, however, that while backlogged demand does imply potential for sales, *not all* backlogged demand is sale-able. To extend this limitation to account for a diminishing return on backlogged demand, define a sequence

$$(a_n)_{n=1}^{\infty}$$

such that

$$\lim_{n \to \infty} a_n = 0$$

and

$$1 \geq a_1 > a_2 > \ldots \geq 0$$

Then, replace the constraint in (13) with

$$\sum_{\tau=1}^{j} a_{(j-\tau+1)} w_{i,\tau,k} \leq \sum_{\tau=1}^{j} a_{(j-\tau+1)} c^w_{i,\tau,k} \ \forall\ i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}]. \tag{14}$$

Note that over the summation in (14), the index of a_n runs from j to 1, with the most recent demand term $c^w_{i,j,k}$ having coefficient a_1, the

greatest coefficient of any demand term. In this way, backlog in demand is seen to "decay", and maximum demand is met. That is, it is always still possible for $w_{i,j,k} = c^w_{i,j,k}$ for all i, j, and k, but if a future time-step is to fill some backlogged demand, it can only fill the fraction a_τ of it, where $a_\tau < 1$ in general.

A convenient choice of a_n would be $a_n = (\mu^w_{i,k})^{n-1}$, where $\mu^w_{i,k} \in [0, 1]$ is a "decay constant" for demand backlog of a product at a given location. Modelling fully non-decaying demand backlog then amounts to letting $\mu^w_{i,k} = 1$, which allows demand to be filled "at any time-step" after the demand presents itself (and $a_n = 1 \ \forall \ n$). A "fully decaying", backlog-discarding model is described by letting $\mu^w_{i,k} = 0$, which allows only demand at the current time-step to be filled (and $a_1 = 1$, $a_n = 0 \ \forall \ n \neq 1$, assuming the convention $0^0 = 1$).

The resulting backlogged demand constraint can be expressed as a convolution of the decay sequence a_n as in (15).

$$(a_n)^j_{n=1} * (w_i, n, k)^j_{n=1} \leq (a_n)^j_{n=1} * (c^w_{i,n,k})^j_{n=1} \ \forall \ i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}]. \tag{15}$$

Consider demand backlog being a function of time as well as location and sku. Alternatively, consider demand backlog to be a function of forecasted demand for a sku at a location and time-step. Perhaps (

Note again that there is still only one demand-sales constraint per product, time-step, and location (and fewer still after pruning). This demand backlog adjustment *does not* increase the number of constraints of the problem or, presumably, the complexity of finding the solution. This does reduce the sparsity of the final constraint matrix, but it isn't clear that adding non-zero entries to that matrix makes the problem more computationally intensive, althought it does, in general, increase the memory required to store the (sparse) constraint matrix. If sparsity in the constraint matrix is seen to in fact yield computational advantages, this note will be amended.

7.3 Deteriorating/Decaying Inventory

Given variables representing inventory at each time-step, any decay (or growth) function corresponding to the reality of the system can be implemented at no computational cost. This can be accomplished by multiplying the coefficient of the expression for $y_{i,j,k}$ in the evolution equation (1) by the constant $\mu^y_{i,j,k} < 1$, which can be called the decay constant of product i at location k at time-step j. If all products decay at the same rate, the decay constant can be simply μ^y. However, for example, peaches decay in Georgia in June (within a day) faster than apples decay in Massachussets in November (in a cellar, over the course of months). Thus, a parameter matrix can be more appropriate than a single real number, at no computational cost.

If this model were used to optimize a small system, such as a commercial kitchen, in which each "location" might be an employee at their station, then the deteriorating good extension might be very valuable. In

such a system, transportation would likely occur between a walk-in refrigerator and a prep table, limited by the "truck size" of what can be carried in a person's arms, with time-steps of just a few minutes. In this case, different deterioration rates might lead to the decision to bring a tray of spinach back to the cooler immediately, rather than leaving it on the table, whereas the mushrooms can be left on the table and carried back to the cooler in the same armful as the fish sauce, after prepping both sliced mushrooms and their eventual sauce. Note that in a situation such as this, the number of total "trucks" in any one time-step would be limited to the number of available "factories" (employees), and a constraint would need to be added:

$$\sum_{j,k,L} t^0_{k,L} + \sum_{i,k} z^0_{i,j,k} + \sum_{k} z^{00}_{j,k} \leq n_{\text{employees}} \tag{16}$$

Modelling decay consists of replacing the evolution equation (1) with

$$y_{i,j,k} = \mu^y_{i,j,k} \left(y_{i,(j-1),k} + z_{i,(j-R^z_{i,k}),k} - \sum_I z_{I,(j-1),k} B_{I,i} + \sum_l t_{i,(j-R^t_{l,k}),l,k} - \sum_l t_{i,(j-1),k,l} - w_{i,(j-1),k} \right). \tag{17}$$

For more information about decaying inventory, but nothing about mixed-integer linear programs, see: `http://www.m-hikari.com/ams/ams-2010/ams-69-72-2010/mishravkAMS69-72-2010.pdf`

7.4 Expiring Inventory

While decaying inventory has been explored previously in the field of supply chain management, hard-window "expiration" or "expiry" has not been incorporated into the models described in academic literature (as far as I can tell). To enforce hard-set expiry of goods within the current model, only a small computational cost is incurred. The necessary constraints will actually limit the solution space and thus may accelerate convergence of an algorithm. Newly-defined decision variables can be included to account for expiry, but, perhaps more easily, the existing framework can be extended to describe product expiry.

For each component i, define time of expiry to be R^y_i. Then, define a location k_{waste} and let the cost to transport to k_{waste}, $f^t_{(k,k_{waste})} = 0 \; \forall \; k \in [1, n_{loc}]$. Here, n_{loc} continues to describe the number of locations *not* including k_{waste} for convenience of documentation. Documentation will be updated when this model extension is incorporated into the model's computation. Allowing for shipment to k_{waste} as to other locations at each time-step allows for shipment of goods to k_{waste} at zero cost to the system (except, of course, opportunity cost - which will be minimized!). The locational parameters describing k_{waste} should include $c^y_{k_{waste}} = \infty$, $f^y_{j,k_{waste}} = 0 \; \forall \; j \in [1, n_t]$, and $R^t_{k,k_{waste}} = 1$, meaning that infinite product can be held indefinitely at no charge at k_{waste}, and waste arrives there as soon as possible after it is shipped. Finally, and crucially, it must be enforced that $c^{t0}_{k_{waste},k} = 0 \; \forall \; k \in [1, n_{loc}]$ and that $c^{t0}_{k,k_{waste}} = 1 \; \forall \; k \in [1, n_{loc}]$, meaning every location can produce waste by shipping to k_{waste}, but no locations can "reclaim" waste of any kind.

The following constraint will ensure that the appropriate quantity of expired goods is in fact shipped to k_{waste} at time of expiry. SCRAP ALL PREVIOUS NOTES. (Make sure it isn't possible to put goods on a truck so that their expiry passes un-noticed while en-route. Account for this!) NEW NOTE: To account for inventory expiry, consider incorporating inventory variables $y^m_{i,j,k}$ where m indicates the number of time-steps that have passed since beginning of lifespan of said good. The evolution equation then decouples into several equations, where production or purchasing adds to a given "shelf-age" inventory level, while holdover of inventory at a given age adds to the subsequent inventory age. The total number of decision variables (for all possible decisions) corresponding to a given product i would then be multiplied by approximately R^y_i, increasing the problem complexity by several times. Assuming the shelf-life for a good is either very short (just a few time-steps) or very long (many, many time-steps for commodities or fully shelf-stable goods), it may be wise to omit long shelf lives by compressing differently shelf-aged inventories into a single inventory, while short-lived goods would be represented by R^y_i multiples of the original model's decision variables, but R^y_i would be small, as stated.

7.5 Accounting for Equity

The current problem formulation optimizes for the profit earned by the final time-step, referred to as n_t. Profit is defined by the objective function (10) to be:

Profit	=	+Sales Revenue
		−Transportation Costs
		−Production Costs
		−Holding Costs

It would not add any computational complexity to expand the objective function to include the value of inventory that is "held", "in-transit," and "in-production" at the final modeled time-step. The objective function would then be:

Profit	=	+Sales Revenue
		−Transportation Costs
		−Production Costs
		−Holding Costs
		+Value of Stored/In-Transit/In-Production Goods

Formally, recall that $f^w_{i,j,k}$ denotes the sales price of product i at time-step j at location k. Analogously, let $f^{w_f}_{i,n_t,k}$ denote the "value" of product i at location k at the final time-step n_t. Let val_{n_t} denote the final "value" of *all* the unsold inventory produced in the system by the final time-step. The value of val_{n_t} can be defined as

$$val_{n_t} = \sum_{i,k} f^{w_f}_{i,n_t,k} y_{i,n_t,k} + \sum_{i,k} \sum_{j=n_t - R^z_{i,k}+1}^{n_t} f^{w_f}_{i,n_t,k} z_{i,j,k} + \sum_{i,k,l} \sum_{j=n_t - R^t_{l,k}+1}^{n_t} f^{w_f}_{i,n_t,k} t_{i,j,l,k}. \tag{18}$$

This can be explained as:

$$val_{n_t} = \underbrace{\sum_{i,k} f^{w_f}_{i,n_t,k} y_{i,n_t,k}}_{\text{unsold, held inventory}} + \underbrace{\sum_{i,k} \sum_{j=n_t - R^z_{i,k}+1}^{n_t} f^{w_f}_{i,n_t,k} z_{i,j,k}}_{\text{“in-production” inventory}} + \underbrace{\sum_{i,k,l} \sum_{j=n_t - R^t_{l,k}+1}^{n_t} f^{w_f}_{i,n_t,k} t_{i,j,l,k}}_{\text{“in-transit” inventory}}.$$

Then let the objective function (10) become:

$$Maximize \sum_{i,j,k} f^w_i w_{i,j,k} - \sum_{j,l,k} f^t_{l,k} t^0_{j,l,k} - \sum_{i,j,k} f^z_{i,k} z_{i,j,k} - \sum_{i,j,k} f^{z0}_{i,k} z^0_{i,j,k} - \sum_{i,j,k} f^y_k s_i y_{i,j,k} + val_{n_t} \tag{19}$$

Valuing equity incentivizes an optimal solution to store value in inventory that is “in-production” or “in-transit” at the final time-step. This is an important fact of a supply chain, but will incentivize production of a potentially huge quantity of product in the final time-steps. Constraints could be added so that the “end-time” production and shipping at a given location do not exceed some aggregate demand forecast over the next several time-steps at that location. If many time-steps are simulated, then it would add a comparatively small computational burden to create different “tiers” of end-time inventory variables, to represent non-linear value. That is, perhaps end-time inventory is valuable, but is modeled most realistically with a non-linear value. Then “end-time” production and shipping orders (which will not be completed by n_t can be enforced to be represented by “bottom-tier” decision variables until their quantity exceeds a threshold at which point they are represented by “top-tier” decision variables, whose value is lower (a separate analogue of $f^{w_f}_{i,n_t,k}$ would be introduced). The quantity of each tier can be limited to the available storage capacity at each location and by forecasted demand.

7.6 Shipping Options

If several shipping options are available, decision variables corresponding to those shipping modes, with possibly several shipping cost/distance matrices $R^{t_1}_{k,L}, \ldots, R^{t_{n_{trucking}}}_{k,L}$.

7.7 Capacity in “Units” vs. Size and Weight (and potentially other quantifiers)

An object does not take up “x units” in a warehouse or on a truck. Rather, a product has a weight and a volume, and they are relevant at different times. Consider storing products’ weights and volumes. The volume would affect storage as well as shipping requirements needed to transport

that good, whereas the weight might affect primarily the shipping requirements. An extra set of "transportation capacity" constraints would be required. Possibly the same number of "holding capacity" requirements would be needed (unless weight was in fact relevant). This would not add to complexity of the MILP, and might in fact limit the number of possible solutions.

7.8 Horizon Stabilization

Presumably, the model will produce un-realistic artifacts of horizon shifting, mostly concerning decisions corresponding to late time-steps. Due to the inability to fully accurately quantify the value of a held-over piece of inventory at the end of a modeled timeframe, optimization may instruct preparing unwise quantities of held-over inventory. Ideally, updates in the system state each time the model is iterated to the next time-step will mean that "next step" instructions are always well-founded, and instructions near the horizon will never be fully relevant. Ideally, however, many instructions will be well founded, so that communication regarding orders from producers and suppliers that are not incorporated into the system can be made as early and accurately as possible. Of course orders that have already been placed can be hard-constrained into the model and incorporated into an optimal plan, but in order to ensure that the artifacts close to the horizon do not compromise the legitimacy of instructions closer to the current time-step, the horizon should be placed as far outward as possible - past the earliest desirable point at which any orders would be need to be placed "ahead of time".

Interestingly, given information regarding the demand function, such as its periodicity (inner product with respect to Fourier bases), total weight (e.g. the L1 norm throughout the product's life-cycle corresponds to "life-cycle sales benchmarks"), certain periodic or long-term optimal system-wide behavior may be analytically determinable. In short, certain types of long-term "planning ahead" would only be possible with a horizon of a certain depth (such as pre-emptively producing a subcomponent at full order size, at the largest production site); but at some "horizon stabilizing" depth, presumably, *all* possible "planning-ahead" strategies would be considered by this model. Ideally, this "horizon stabilizing" depth should be the depth included in the model for optimization of the system to fully manifest.

Can this depth be derived analytically from the demand functions and parameters? Does it perhaps need to be determined by simulation? This question warrants further exploration.

7.9 Linking Adjacent Supply Chains

The computational cost of expanding any dimension of a supply chain is seen to be relatively high. Given new access to parallel processing, advances in processor speed, and advances in integer linear programming, such as cutting plane automation, this work may have recently become feasible despite computational limitations. However, there will always

be a computational advantage to disaggregating sub-supply chains, and optimization may still be possible.

One way to do this is to divide a supply chain into smaller supply chains, possibly dictated by the individual hubs. Then, to purchase any good from a separate supply chain, there will be an "acquisition lead time" to purchase product i from location k at time-step j denoted $R^w_{i,j,k}$. This lead time will not be equivalent to the production lead times familiar to SCM. Instead, $R^w_{i,j,k}$ will be defined by the number of time-steps necessary to ship product i from *the nearest facility in the wider supply chain network that holds that product/component/subassembly/material.*

To calculate $R^w_{i,j,k}$ during pre-processing, all supply chains involving SKU i will be subjected to planning optimization (through the very model in this paper) with the parameter $c^w_{i,j',k'}$ (the demand for SKU i, which may be zero by default if it is a sub-assembly) set to the proposed cross-supply chain demand, or, alternatively, with an enforced minimum production order or sales allocation/benchmark before a certain time-step. The sub-supply chain whose lead time is lowest may dictate the value of $R^w_{i,j,k}$ for inter-SC purchasing.

Iteratively optimizing over different supply chains to generate different lead times (and even equity valuations) may disaggregate the computation sufficiently to allow for robust system-wide solution navigation. By iteratively improving inter-SC decisions, a gradient-descent for the overall supply chain is carried out. Depending on the efficiency of disaggregated MILP computation, a database could be constantly updated governing inter-SC commerce by keeping track of the minimum lead times for subassemblies across optimization borders.

By dis-aggregating and optimizing over smaller systems, "benchmark descriptors" can be defined and established meaningfully, such as the equity valuation mentioned in 7.5. These descriptors can govern inter-SC commerce, which can be utilized optimally by purchasing subject to lead times R^w, at cost governed by the equity descriptor. By optimizing subject to all hard-constrained orders for the various subassemblies that could be utilized inter-SC, and maintaining a database of the resulting valuations (determined by profit reduction in the system that produces the SKU) and corresponding lead times, inter-SC commerce would be effortless. Furthermore, if each SC in a given "SC net" optimized at each time step subject to inter-SC orders, system-wide optimization would be achieved, or nearly achieved, as though the system had not been disaggregated at all.

Different inter-SC subassembly trade costs can be associated with different lead-times. In fact, this common platform will be a novel way to consistently account for that most basic of facts: that different lead times have different costs. There will be no guesswork in setting the prices for different lead times, and there will be no need for competition to whittle away surplus value on any end - all participating SCs will presumably agree to share system-wide savings.

!!!

8 Symbol Guide

Hi Theo, here is a guide to some of the math symbols I use all the time.

Symbol	Meaning	Example	Example Meaning
$\in$	"is in" or "is an element of"	$x \in [0, 1]$	Variable x is in the interval from 0 to 1. Equivalent to $0 \leq x \leq 1$.
$[a, b]$	The interval including all real numbers between a and b (including a and b)	$x \in [a, b]$	Variable x is in the interval from a to b
$\{a, b, c, d\}$	The set containing the numbers/vectors/elements a, b, c and d	$x \in \{a, b, c, d\}$	Variable x is equal to a, b, c, or d
$\sum_{x=a}^{b} f(x)$	The sum of the function $f(x)$ evaluated at all integer values of x between a and b	$\sum_{x=1}^{4} x^2 = \sum_{x \in \{1,2,3,4\}} x^2$	$1^2 + 2^2 + 3^2 + 4^2$
		$\sum_{i=1}^{2} \sum_{x=1}^{5} x^i$	$(1^1 + 2^1 + 3^1 + 4^1) + (1^2 + 2^2 + 3^2 + 4^2)$
		$\sum_{i,x} x^i = \sum_{i} \sum_{x} x^i$	If the only possible values of x and i are obvious, this notation suffices
$\forall$	"for all"	$w_{i,j,k} \leq dem_{i,j,k} \ \forall \ i, j, k$	Sales of product i at time-step j at location k are less than the demand for product i at time-step j at location k, and this goes FOR ALL products i, time-steps j, and locations k.
		$\forall \ i \in [1, n_z], j \in [1, n_t], k \in [1, n_{loc}]$	For all i in the interval $[1, n_z], \forall j$ in the interval...Sometimes it is useful to specify ALL the values that a variable can take on.
		$\forall \ i \ \neq I$	For all values i NOT EQUAL TO I. This is a useful way to refer to all indices EXCEPT a given one.