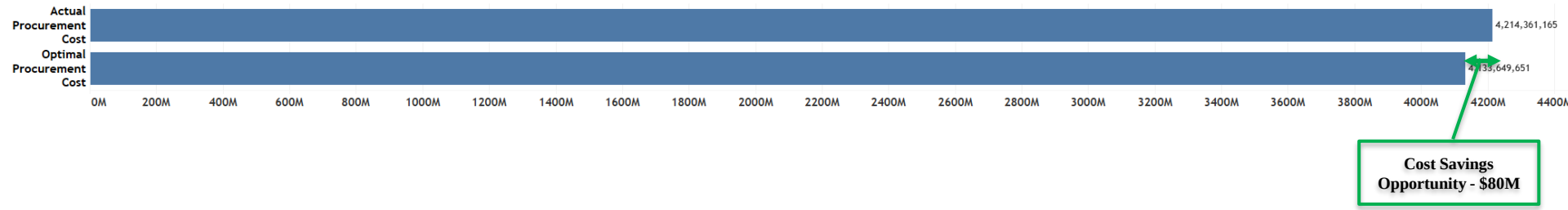
ABC Procurement Optimization – Problem Formulation

ABC Procurement Optimization – Overview

- Project Goal – Find potential cost savings by minimizing the procurement costs of buying fungible (interchangeable) products.
- POC created successfully – Built optimization model to identify any cost savings
  - Captured all applicable rebate programs offered by vendors and applied the discounts from these programs to get net-net costs of purchasing products
  - Ran the optimization model to meet supply-demand, etc. and find the best products to buy in order to achieve minimum procurement costs
  - Used year 2021 to compare the procurement costs
- Results - For 2021, the actual spend on products across ABC was $4.214 B, and the model recommends that the same purchases can be done with $4.133 B with a potential opportunity of ~$80 M costs reduction.

Branch Num	Brand Line Number	Vendor	Actual Quantity	Actual Procurement Cost	Composite Unit Cost	Optimal Quantity	Optimal Procurement Cost
001	5116	ALCO NVC INC			12	3,880	46,133
	5114	TARCO INC	11,520	133,594	13	22,120	257,184
	4098	RESISTO			14	15,400	172,018
	6814	OWENS CORNING	21,478	424,697	21		
	1021	GAF BUILDING MATERIALS CORP	3,358	69,166	25		
	3916	MALARKEY ROOFING PRODUCTS	839	18,133	26		
	1011	CERTAINTEED	4,197	107,122	40		
002	4098	RESISTO			16	12,040	144,341
	5114	TARCO INC	7,860	100,751	18	1,440	16,827
	6814	OWENS CORNING	839	18,957	24		
	1021	GAF BUILDING MATERIALS CORP	1,999	41,354	25		
	1010	OWENS CORNING	1,499	32,060	26		
	1011	CERTAINTEED	1,276	27,098	27		
004	5114	TARCO INC			14	3,660	42,888
	4098	RESISTO	1,296	17,768	16	7,168	84,741
	1021	GAF BUILDING MATERIALS CORP	3,278	69,602	27		
	3415	IKO MANUFACTURING INC	4,310	110,168	29		
	1020	GAF BUILDING MATERIALS CORP	800	24,241	38		
	1011	CERTAINTEED	1,137	47,209	56		
005	4098	RESISTO	3,168	38,868	16	26,272	328,115
	5114	TARCO INC	12,960	176,076	18	3,780	44,138
	6814	OWENS CORNING	1,620	35,946	23		
	3416	IKO MANUFACTURING INC	3,342	68,294	25		
	1021	GAF BUILDING MATERIALS CORP	3,157	65,770	29		
	1010	OWENS CORNING	4,499	100,410	30		
	1020	GAF BUILDING MATERIALS CORP	700	21,921	38		
	1011	CERTAINTEED	599	24,467	52		

Purchase recommendations



Assumptions & Important Business Logic

- Fungible groups were manually identified by Chris Rossignol based on item profile and price.
- The Fungible Product Groups in Scope are -
  - STANDARD STARTER SHINGLE
  - PREMIUM ICE & WATER
  - SHINGLE MFG DISCRETIONARY ICE&WATER
  - NON-SHINGLE MFG DISCRETIONARY ICE&W
  - HIGH TEMP ICE & WATER

- In-scope programs - All programs which are applicable to the above 5 fungible product groups.

- Non-Fungible Products – All products which are impacted by the in-scope programs but are not part of fungible groups.
  These products are involved in the analysis because the discounts from in scope programs can change the net-net costs for these products.
  - Examples – PREMIUM STARTER SHINGLES, TAPE ICE & WATER, etc.

Assumptions & Important Business Logic

- Replacement Cost / Unit Cost for Items
  - For baseline we get the replacement cost for the item-branch-vendor-month as the monthly average unit cost from the receipt costs.
  - If Unit cost for any item at a branch from a vendor in a month is not available, then it is checked if the item is purchased at any branch from the vendor in the region in that month and if yes, then we fill it with the median unit cost of all receipts from that month.
  - An item should be at least purchased 10 times in the year in any district or region from a vendor in order to use the unit costs for filling other branches in the district or region.
  - If the receipt cost is not available then the supplier replacement cost from Inventory Dataset is used.
- Unit Cost (Composite) for Brand lines
  - The unit cost at brand line level is a weighted unit cost prepared using the receipts value proportions of the items in that brand line.
  - This is required in order to provide the Brandline a representative item cost, while our costs are at the SKU level. Rollup to Brandline level is required in order to execute the calculations required of the POC.
- Program Weighted Reporting UOM-Program UOM
  - Any program which is not specified in reporting UOM (gate limits) then this program is converted to reporting UOM using the receipts quantity proportions of the items in that program
- Branch Vendor Freight
  - If there is no freight available for a branch vendor combination then we fill it with 6%, prompt pay and additional discount as 0%

Model – Inputs

Problem Definition:

Minimize Merchandise Procurement Costs across all ABC branches while satisfying the demand of all products

Inputs:

- f – set of product groups
- i – set of brand lines
- b – set of branches
- g – set of gates
- t – time period - month
- v – set of vendors
- p – set of programs

Model – Inputs Parameters

- $unit\_cost_{fivbt}$ – unit cost (composite) of product group f and brand line i from vendor v at branch b at time t
- $unit\_cost\_discount_{fivbpgt}$ – unit cost discount in dollars calculated* for program p at gate g applicable to product group f and brand line i purchased from vendor v at branch b at time t

  *The program chaining will be applicable to the applicable programs to get this discount for the unit cost
- $demand_{fbt}$ - forecasted in-scope discretionary demand of product group f at branch b at time t. This excludes the customer excluded qty (qty sold to excluded customers). **The non- discretionary demand has to be subtracted from forecast to get this demand**
- $actual\_ptd\_qty_{fivbpt}$ - the actual quantity purchased to date of product group f and brand line i from vendor v at branch b for program p at time t. The short time-based programs like programs only applicable in a month will have 0 value e.g. programs applicable only in May can't use April's quantity.
- os_qty_{fivbpt} - the out-of-scope quantity of out-of-scope product group f and brand line i purchased from vendor v at branch b and time t following under program p at time t
- $qty\_out\_scope_{p}$ - Quantity purchased for all out-of-scope products which follow under in scope programs program p
- $ptd\_cost\_discount_{fivbptg}$ - cost discount in dollars for the actual_ptd quantity of product group f purchased from vendor v at branch b for program p at gate g at time t
- $os\_cost\_discount_{fivbptg}$ - The cost discount in dollars for the out-of-scope quantity of product group f purchased from vendor v at branch b for program p at gate g at time t
- $jobquote\_prop_{fb}$ - the proportion of forecasts of product group f at branch b in last 12months which will be job quotes
- $directs\_prop_{fb}$ - the proportion of forecasts of product group f at branch b in last 12 months which will be direct shipments
- $products\_program\_prop_{fpv}$ - the proportion of sales of product group f among all the total products sales which belongs to a program p from vendor v
- **$non\_disc\_qty_{fivbpt}$ - the non – discretionary quantity of product group f and brand line i which is fixed to be purchased from vendor v at branch b and time t following under program p. *Need to decide the proportions.**
- **$non\_disc\_cost_{fivbpt}$ - the non – discretionary cost of product group f and brand line i which is fixed to be purchased from vendor v at branch b and time t following under program p.**
- **$non\_disc\_cost\_discount_{fivbpgt}$ - The cost discount in dollars for non – discretionary quantity of product group f purchased from vendor v at branch b for program p at gate g at time t**

Model – Decision Variables

Decision variables:

Continuous variables

$Q_{fivbpgt}$ - In-Scope Quantity of product group f and brand line i purchased from vendor v at branch b under program p at gate g and time t

Binary variables

PG_{pg} 1, If gate g is selected for program p;

0, o.w.

Model – Objective Function

Objective Function:

minimize Procurement Costs of all product groups purchased at all ABC branches from vendors in the calendar year

Minimize Obj = Procurement Cost– Program Discounts

Procurement Cost

$$= \sum_{fivbt,p\_list} \sum_{g \in p\_list[0]} ((unit\_cost_{fivbt} * \ Q_{fivbp\_list[0]gt}) + (actual\_ptd\_cost_{fivbtp\_list[0]}) + (out\_scope\_cost_{fivbtp\_list[0]}) + (\boldsymbol{non\_disc\_cost_{fivbtp\_list[0]}})) * (1 + \text{Freight\_Percent}_{vb} - \text{Prompt\_Pay\_Percent}_{vb})$$

sum over all combinations of $fivbt, p\_list$; *where p_list is the programs chaining at* $fivbt$ *and g is gates for first program p_list[0]*

Program Discounts

= *total discount from all programs for non job quotes and non direct forecast +*

*total discount for job quote forecast quantity (PO_TYPE=J) **+*

*total discount for direct shipmentforecastquantity (WAREHOUSE = D)***

$$= \sum_{fivbpgt} ((unit\_cost\_discount_{fivbpgt} * \ Q_{fivbpgt}) + (actual\_ptd\_cost\_discount_{fivbpgt} * PG_{pg}) + (out\_scope\_cost\_discount_{fivbpgt} * PG_{pg}) + (\boldsymbol{non\_disc\_cost\_discount_{fivbpgt} * PG_{pg}})) * (1 - jobquote\_prop_{fb} - directs\_prop_{fb} - jobquote\_directs\_prop_{fb})$$

$$+ \sum_{fivbpgt;po\_type=J} ((unit\_cost\_discount_{fivbpgt} * \ Q_{fivbpgt}) + (actual\_ptd\_cost\_discount_{fivbpgt} * PG_{pg}) + (out\_scope\_cost\_discount_{fivbpgt} * PG_{pg}) + (non\_disc\_cost\_discount_{fivbpgt} * PG_{pg})) * (jobquote\_prop_{fb})$$

$$+ \sum_{fivbpgt;warehouse=D} ((unit\_cost\_discount_{fivbpgt} * \ Q_{fivbpgt}) + (actual\_ptd\_cost\_discount_{fivbpgt} * PG_{pg}) + (out\_scope\_cost\_discount_{fivbpgt} * PG_{pg}) + (non\_disc\_cost\_discount_{fivbpgt} * PG_{pg})) * (directs\_prop_{fb})$$

**the po_type=J means that for the unit cost discount calculations we assume that this quantity will be Job Quotes and any programs which do not provide rebates for job quotes will not provide any discount. So, we will simply do not chain such programs to other programs applicable. Same for Directs - warehouse = D case.

Model – Constraints

1. Supply should be equal to **discretionary** demand

   Quantity of product group f purchased from all vendors v at branch b for all gates g should be equal to the discretionary demand for that product group at that branch at all time periods t

   $$\sum_{f'ivb't',p\_list} \sum_{g \in p\_list[0]} Q_{fivbp\_list[0]gt} == demand_{fbt};$$

   $$\forall\ fbt;\ where\ p\_list\ is\ the\ programs\ chaining\ at\ fivbt\ and\ g\ is\ gates\ for\ program\ p\_list[0];\ f' = f;\ b' = b;\ t' = t;$$

2. All programs chaining for a product group f purchased from vendor v at branch b at any time t will have same quantity for all gates

   $$\sum_g Q_{fivbtp\_list[0]g} == \sum_g Q_{fivbtpg};$$

   $$for\ all\ combinations\ of\ fivbt, p\_list;\ for\ all\ programs\ p\ in\ p\_list[1:];\ for\ all\ gates\ in\ g \in p\_list[0]$$

3. Each program should only be applied to one gate

   $$\sum_g PG_{pg} == 1; \qquad \forall p;$$

4. For any program p at any gate g for remaining time t, the quantity of all applicable product groups f purchased from vendor v at all applicable branches b at all remaining time t should be 0 or less than the max demand

   $$\sum_{fivbt}((Q_{fivbpgt}) \leq BigM * PG_{pg};\ \forall p, \forall g;\ BigM = \sum_{f \in pbt} demand_{fbt}\ \forall t;$$

5. For any program p to qualify any gate g the quantity of all applicable product groups f purchased from vendor v at all applicable branches b for all applicable time t should be within the gate limits

   max_original_gate_limit[p] = original gate lower limit of the program p (not the new created gates limit)

   only write constraint if for a program p and gate g : Lower or Upper_limit$[p, g]$ *<= max_original_gate_limit[p];* as after max limit there is no constraint on program and gates created after that do not matter here in this quantity lower-upper limits constraints then

   I. Lower limit

   $$\sum_{fivbt}((Q_{fivbpgt}) + (actual\_ptd\_qty_{fivbpt} * PG_{pg}) + (\boldsymbol{non\_disc\_qty_{fivbpt} * PG_{pg}}) + (out\_scope\_qty_p * PG_{pg})) \geq Lower\_limit[p, g] * PG_{pg};$$

   $\forall p, \forall g,;$ *only write constraint if* $Lower\_limit[p, g]$ <= max_original_gate_limit[p]

   II. Upper limit

   $$\sum_{fivbt}((Q_{fivbpgt}) + (actual\_ptd\_qty_{fivbpt} * PG_{pg}) + (\boldsymbol{non\_disc\_qty_{fivbpt} * PG_{pg}}) + (out\_scope\_qty_p * PG_{pg})) \leq Upper\_limit[p, g] * PG_{pg};$$

   $\forall p, \forall g,;$ *only write constraint if* $Upper\_limit[p, g]$ <= max_original_gate_limit[p]

Model – Constraints – Baseline Mapping

For model in [optimal, business_as_usual]

Only write constraint → If model == 'business_as_usual':

6. Supply should be >= equal to discretionary forecast for the branches

Notes

$\sum_{g \in p\_list[0]} Q_{fivbtpg} >= DISC\_Forecast_{fivbt};$

each constraint for all combinations of $fivbt, p\_list$; *for all programs p in p_list ;*

Model – Example 1

Product Group - f	Brand Lines - i	Vendor - v	Program - p	Branch - b	Month - m	Gate - g
	3180	OC	1. 5714-21-OC-NE 2. 004-21-OC 3. 1011-21-OC-NE-JAN			
SSS	2664	OC	1. 004-21-OC 2. 1011-21-OC-NE-JAN	b = 584	m = 1	
	3932	ATL	1. 010-21-ATL 2. 401-21-ATL-NE			

gate_id	limit	org_limit
0	0	0
1	214,854	0
2	234,386	0
3	1,220,000	0
4	1,320,000	0
5	1,420,000	0

gate_id	limit	org_limit
0	0	0
1	215,083	0
2	726,161	0
3	792,176	0
4	858,192	0
5	1,147,228	0
6	1,180,005	0
7	1,223,709	0
8	1,267,414	0
9	1,311,117	0

Objective Function:

UC[SSS, **3180**, **OC**, 584 ,1] * Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] + … for all gates to 5
- UCD[SSS, **3180**, **OC**, 584 ,1] * Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] - … for all gates to 5
- UCD[SSS, **3180**, **OC**, 584 ,1] * Q[SSS, **3180**, **OC**, 584 ,1 **004-21-OC**, 0] - … for all gates to 5
- UCD[SSS, **3180**, **OC**, 584 ,1] * Q[SSS, **3180**, **OC**, 584 ,1, **1011-21-OC-NE-JAN**, 0] - … for all gates to 5
+ OS_COST [*, *, **OC**, 584 ,1, **5714-21-OC-NE,0**] * PG[**5714-21-OC-NE, 0**] + … for all gates to 5

+ UC[SSS, **2664**, **OC**, 584, 1] * Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] + … for all gates to 5
- UCD[SSS, **2664**, **OC**, 584, 1] * Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] - … for all gates to 5
- UCD[SSS, **2664**, **OC**, 584, 1] * Q[SSS, **2664**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] - … for all gates to 5
+ OS_ COST[*, *, **OC**, 584, 1, **004-21-OC ,0**] * PG[**004-21-OC ,0**] + OS_COST[**004-21-OC ,1**] * PG[**004-21-OC ,1**] + … for all gates to 5

+ UC[SSS, **3932**, **ATL**, 584, 1] * Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] + … for all gates to 9
- UCD[SSS, **3932**, **ATL**, 584, 1] * Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] - … for all gates to 9
- UCD[SSS, **3932**, **ATL**, 584, 1] * Q[SSS, **3932**, **ATL**, 584, 1, **401-21-ATL-NE**, 0] - … for all gates to 9
+ OS_ COST[*, *, **ATL**, 584, 1, **010-21-ATL, 0**] * PG[**010-21-ATL, 0**] + OS_ COST[**010-21-ATL ,1**] * PG[**010-21-ATL ,1**] + … for all gates to 9

UC = Unit Cost; UCD = Unit Cost Discount (program discount on a unit after chaining with other programs); OS_ COST = Out Of Scope products net net Costs for p @ g

Model – Example 1

Product Group - f	Brand Lines - i	Vendor - v	Program - p	Branch - b	Month - m	Gate - g
	3180	OC	1. 5714-21-OC-NE 2. 004-21-OC 3. 1011-21-OC-NE-JAN			
SSS	2664	OC	1. 004-21-OC 2. 1011-21-OC-NE-JAN	b = 584	m = 1	
	3932	ATL	1. 010-21-ATL 2. 401-21-ATL-NE			

gate_id	limit	org_limit
0	0	0
1	214,854	0
2	234,386	0
3	1,220,000	0
4	1,320,000	0
5	1,420,000	0

gate_id	limit	org_limit
0	0	0
1	215,083	0
2	726,161	0
3	792,176	0
4	858,192	0
5	1,147,228	0
6	1,180,005	0
7	1,223,709	0
8	1,267,414	0
9	1,311,117	0

Constraints:

1. Demand_SSS_584_1 == 1000

Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] + … for all gates to 5
+ Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] + … for all gates to 5
+ Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] + … for all gates to 9 == 1000

2. Same Quantity for Programs Chaining

Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] + … for all gates to 5
== Q[SSS, **3180**, **OC**, 584 ,1, **004-21-OC**, 0] + … for all gates to 5
== Q[SSS, **3180**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] + … for all gates to 5

Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] + … for all gates to 5
== Q[SSS, **2664**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] +… for all gates to 5

Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] + … for all gates to 9
== Q[SSS, **3932**, **ATL**, 584, 1, **401-21-ATL-NE**, 0] + … for all gates to 9

3. One gate for each program

PG[**5714-21-OC-NE**, 0] + … for all gates to 5 == 1
PG[**004-21-OC**, 0] + … for all gates to 5 == 1
PG[**1011-21-OC-NE-JAN**, 0] + … for all gates to 5 == 1
PG[**010-21-ATL**, 0] + … for all gates to 9 == 1
PG[**401-21-ATL-NE**, 0] + … for all gates to 9 == 1

Model – Example 1

Product Group - f	Brand Lines - i	Vendor - v	Program - p	Branch - b	Month - m	Gate - g
	3180	OC	1. 5714-21-OC-NE 2. 004-21-OC 3. 1011-21-OC-NE-JAN			
SSS	2664	OC	1. 004-21-OC 2. 1011-21-OC-NE-JAN	b = 584	m = 1	
	3932	ATL	1. 010-21-ATL 2. 401-21-ATL-NE			

gate_id	limit	org_limit
0	0	0
1	214,854	0
2	234,386	0
3	1,220,000	0
4	1,320,000	0
5	1,420,000	0

gate_id	limit	org_limit
0	0	0
1	215,083	0
2	726,161	0
3	792,176	0
4	858,192	0
5	1,147,228	0
6	1,180,005	0
7	1,223,709	0
8	1,267,414	0
9	1,311,117	0

Constraints:

4. BigM – sum of all quantity for a program in a gate <= BigM

Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] <= BigM * PG[**5714-21-OC-NE**, 0] for all gates to 5

Q[SSS, **3180**, **OC**, 584 ,1, **004-21-OC**, 0] + Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] <= BigM * PG[**004-21-OC**, 0] for all gates to 5

Q[SSS, **3180**, **OC**, 584 ,1, **1011-21-OC-NE-JAN**, 0] + Q[SSS, **2664**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] <= BigM * PG[**1011-21-OC-NE-JAN**, 0] all gates

Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] <= BigM * PG[**010-21-ATL**, 0] for all gates to 9

Q[SSS, **3932**, **ATL**, 584, 1, **401-21-ATL-NE**, 0] <= BigM * PG[**401-21-ATL-NE**, 0] for all gates to 9

5. Program Gate Limits

Max Gate Limit = 0 for all programs

Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] + OS_ QTY[**5714-21-OC-NE**] * PG[**5714-21-OC-NE**,**0**] >= 0; ONLY FOR GATE 0
Q[SSS, **3180**, **OC**, 584 ,1, **004-21-OC**, 0] + Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] + OS_ QTY[**004-21-OC**] * PG[**004-21-OC**,**0**] >= 0; ONLY FOR GATE 0
Q[SSS, **3180**, **OC**, 584 ,1, **1011-21-OC-NE-JAN**, 0] + Q[SSS, **2664**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] + OS_ QTY[**1011-21-OC-NE-JAN**] * PG[**1011-21-OC-NE-JAN** , 0] >= 0; ONLY FOR GATE 0
Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] + OS_ QTY[**010-21-ATL**] * PG[**010-21-ATL**,**0**] >= 0; ONLY FOR GATE 0
Q[SSS, **3932**, **ATL**, 584, 1, **401-21-ATL-NE**, 0] + OS_ QTY[**401-21-ATL-NE**] * PG[**401-21-ATL-NE**,**0**] >= 0; ONLY FOR GATE 0

Model – Example 2

Product Group - f	Brand Lines - i	Vendor - v	Program - p	Branch - b	Month - m	Gate - g
SSS	3180	OC	1. 5714-21-OC-NE 2. 004-21-OC 3. 1011-21-OC-NE-JAN	b = 584	m = 1	
	2664	OC	1. 004-21-OC 2. 1011-21-OC-NE-JAN			
	3932	ATL	1. 010-21-ATL 2. 401-21-ATL-NE			
	3933	ATL	1. 5015-21-ATL 2. 010-21-ATL	b = 416	m = 1	

gate_id	limit	org_limit
0	0	
1	214,854	
2	234,386	
3	1,220,000	
4	1,320,000	
5	1,420,000	

gate_id	limit	org_limit
0	0	0
1	215,083	0
2	726,161	0
3	792,176	0
4	858,192	0
5	1,147,228	0
6	1,180,005	0
7	1,223,709	0
8	1,267,414	0
9	1,311,117	0

gate_id	limit	org_limit
0	0	0
1	215,083	215,083
2	726,161	215,083
3	792,176	215,083
4	858,192	215,083
5	1,147,228	215,083
6	1,180,005	215,083
7	1,223,709	215,083
8	1,267,414	215,083
9	1,311,117	215,083

Objective Function:

UC[SSS, **3180**, **OC**, 584 ,1] * Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] + … for all gates to 5
- UCD[SSS, **3180**, **OC**, 584 ,1] * Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] - … for all gates to 5
- UCD[SSS, **3180**, **OC**, 584 ,1] * Q[SSS, **3180**, **OC**, 584 ,1 **004-21-OC**, 0] - … for all gates to 5
- UCD[SSS, **3180**, **OC**, 584 ,1] * Q[SSS, **3180**, **OC**, 584 ,1, **1011-21-OC-NE-JAN**, 0] - … for all gates to 5
+ OS_COST[*, *, **OC**, 584 ,1, **5714-21-OC-NE,0**] * PG[**5714-21-OC-NE, 0**] + … for all gates to 5

+ UC[SSS, **2664**, **OC**, 584, 1] * Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] + … for all gates to 5
- UCD[SSS, **2664**, **OC**, 584, 1] * Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] - … for all gates to 5
- UCD[SSS, **2664**, **OC**, 584, 1] * Q[SSS, **2664**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] - … for all gates to 5
+ OS_ COST[*, *, **OC**, 584, 1, **004-21-OC ,0**] * PG[**004-21-OC ,0**] + … for all gates to 5

+ UC[SSS, **3932**, **ATL**, 584, 1] * Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] + … for all gates to 9
- UCD[SSS, **3932**, **ATL**, 584, 1] * Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] - … for all gates to 9
- UCD[SSS, **3932**, **ATL**, 584, 1] * Q[SSS, **3932**, **ATL**, 584, 1, **401-21-ATL-NE**, 0] - … for all gates to 9
+ OS_ COST[*, *, **ATL**, 584, 1, **010-21-ATL, 0**] * PG[**010-21-ATL, 0**] + … for all gates to 9

+ UC[SSS, **3933**, **ATL**, 416, 1] * Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 0] + … for all gates to 9
- UCD[SSS, **3933**, **ATL**, 416, 1] * Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 0] - … for all gates to 9
- UCD[SSS, **3933**, **ATL**, 416, 1] * Q[SSS, **3933**, **ATL**, 416, 1, **010-21-ATL**, 0] - … for all gates to 9
+ OS_ COST[***, *, ATL**, 416, 1, **5015-21-ATL, 0**] * PG[**5015-21-ATL, 0**] + … for all gates to 9

Model – Example 2

Product Group - f	Brand Lines - i	Vendor - v	Program - p	Branch - b	Month - m	Gate - g
SSS	3180	OC	1. 5714-21-OC-NE 2. 004-21-OC 3. 1011-21-OC-NE-JAN	b = 584	m = 1	
	2664	OC	1. 004-21-OC 2. 1011-21-OC-NE-JAN			
	3932	ATL	1. 010-21-ATL 2. 401-21-ATL-NE			
	3933	ATL	1. 5015-21-ATL 2. 010-21-ATL	b = 416	m = 1	

gate_id	limit	org_limit
0	0	
1	214,854	
2	234,386	
3	1,220,000	
4	1,320,000	
5	1,420,000	

gate_id	limit	org_limit
0	0	0
1	215,083	0
2	726,161	0
3	792,176	0
4	858,192	0
5	1,147,228	0
6	1,180,005	0
7	1,223,709	0
8	1,267,414	0
9	1,311,117	0

gate_id	limit	org_limit
0	0	0
1	215,083	215,083
2	726,161	215,083
3	792,176	215,083
4	858,192	215,083
5	1,147,228	215,083
6	1,180,005	215,083
7	1,223,709	215,083
8	1,267,414	215,083
9	1,311,117	215,083

Constraints:

1. Demand_SSS_584_1 == 1000

Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] + … for all gates to 5
+ Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] + … for all gates to 5
+ Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] + … for all gates to 9 == 1000

Demand_SSS_416_1 == 200

Q[SSS, **3933**, **ATL**, 584, 1, **5015-21-ATL**, 0] + … for all gates to 9 == 200

2. Same Quantity for Programs Chaining

Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] +… == Q[SSS, **3180**, **OC**, 584 ,1, **004-21-OC**, 0] +…
== Q[SSS, **3180**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] +… for all gates to 5

Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] +… == Q[SSS, **2664**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] +… for all gates to 5

Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] +… == Q[SSS, **3932**, **ATL**, 584, 1, **401-21-ATL-NE**, 0] +… for all gates to 9

Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 0] == Q[SSS, **3933**, **ATL**, 416, 1, **010-21-ATL**, 0] … for all gates to 9

3. One gate for each program

PG[**5714-21-OC-NE**, 0] + … for all gates to 5 == 1; PG[**004-21-OC**, 0] + … for all gates to 5 == 1
PG[**1011-21-OC-NE-JAN**, 0] + … for all gates to 5 == 1; PG[**010-21-ATL**, 0] + … for all gates to 9 == 1
PG[**401-21-ATL-NE**, 0] + … for all gates to 9 == 1;
PG[**5015-21-ATL**, 0] + … for all gates to 9 == 1; PG[**010-21-ATL**, 0] + … for all gates to 5 == 1

Model – Example 2

Product Group - f	Brand Lines - i	Vendor - v	Program - p	Branch - b	Month - m	Gate - g
SSS	3180	OC	1. 5714-21-OC-NE 2. 004-21-OC 3. 1011-21-OC-NE-JAN	b = 584	m = 1	
	2664	OC	1. 004-21-OC 2. 1011-21-OC-NE-JAN			
	3932	ATL	1. 010-21-ATL 2. 401-21-ATL-NE			
	3933	ATL	1. 5015-21-ATL 2. 010-21-ATL	b = 416	m = 1	

gate_id	limit	org_limit
0	0	
1	214,854	
2	234,386	
3	1,220,000	
4	1,320,000	
5	1,420,000	

gate_id	limit	org_limit
0	0	0
1	215,083	0
2	726,161	0
3	792,176	0
4	858,192	0
5	1,147,228	0
6	1,180,005	0
7	1,223,709	0
8	1,267,414	0
9	1,311,117	0

gate_id	limit	org_limit
0	0	0
1	215,083	215,083
2	726,161	215,083
3	792,176	215,083
4	858,192	215,083
5	1,147,228	215,083
6	1,180,005	215,083
7	1,223,709	215,083
8	1,267,414	215,083
9	1,311,117	215,083

Constraints:

4. BigM – sum of all quantity for a program in a gate <= BigM

Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] <= BigM * PG[**5714-21-OC-NE**, 0] for all gates to 5

Q[SSS, **3180**, **OC**, 584 ,1, **004-21-OC**, 0] + Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] <= BigM * PG[**004-21-OC**, 0] for all gates to 5

Q[SSS, **3180**, **OC**, 584 ,1, **1011-21-OC-NE-JAN**, 0] + Q[SSS, **2664**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] <= BigM * PG[**1011-21-OC-NE-JAN**, 0] all gates

Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] + Q[SSS, **3933**, **ATL**, 416, 1, **010-21-ATL**, 0] <= BigM * PG[**010-21-ATL**, 0] for all gates to 9

Q[SSS, **3932**, **ATL**, 584, 1, **401-21-ATL-NE**, 0] <= BigM * PG[**401-21-ATL-NE**, 0] for all gates to 9

Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 0] <= BigM * PG[**5015-21-ATL**, 0] for all gates to 9

5. Program Gate Limits

Max Gate Limit = 0 for all programs except 5015-21-ATL

Q[SSS, **3180**, **OC**, 584 ,1, **5714-21-OC-NE**, 0] + OS_ QTY[**5714-21-OC-NE**] * PG[**5714-21-OC-NE,0**] >= 0; ONLY FOR GATE 0

Q[SSS, **3180**, **OC**, 584 ,1, **004-21-OC**, 0] + Q[SSS, **2664**, **OC**, 584, 1, **004-21-OC**, 0] + OS_ QTY[**004-21-OC**] * PG[**004-21-OC,0**] >= 0; ONLY FOR GATE 0

Q[SSS, **3180**, **OC**, 584 ,1, **1011-21-OC-NE-JAN**, 0] + Q[SSS, **2664**, **OC**, 584, 1, **1011-21-OC-NE-JAN**, 0] + OS_ QTY[**1011-21-OC-NE-JAN**] * PG[**1011-21-OC-NE-JAN** , 0] >= 0; ONLY FOR GATE 0

Q[SSS, **3932**, **ATL**, 584, 1, **010-21-ATL**, 0] + OS_ QTY[**010-21-ATL**] * PG[**010-21-ATL,0**] >= 0; ONLY FOR GATE 0

Q[SSS, **3932**, **ATL**, 584, 1, **401-21-ATL-NE**, 0] + OS_ QTY[**401-21-ATL-NE**] * PG[**401-21-ATL-NE,0**] >= 0; ONLY FOR GATE 0

Max Gate Limit for Program **5015-21-ATL = 215,083**

Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 0] + OS_ QTY[**5015-21-ATL**] * PG[**5015-21-ATL,0**] >= 0

Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 0] + OS_ QTY[**5015-21-ATL**] * PG[**5015-21-ATL,0**] <= 215,083 * PG[**5015-21-ATL,0**]

Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 1] + OS_ QTY[**5015-21-ATL**] * PG[**5015-21-ATL,1**] >= 215,083 * PG[**5015-21-ATL,1**] ;

Model – Example 2

Product Group - f	Brand Lines - i	Vendor - v	Program - p	Branch - b	Month - m	Gate - g
SSS	3180	OC	1. 5714-21-OC-NE 2. 004-21-OC 3. 1011-21-OC-NE-JAN	b = 584	m = 1	
	2664	OC	1. 004-21-OC 2. 1011-21-OC-NE-JAN			
	3932	ATL	1. 010-21-ATL 2. 401-21-ATL-NE			
	3933	ATL	1. 5015-21-ATL 2. 010-21-ATL	b = 416	m = 1	

gate_id	limit	org_limit
0	0	
1	214,854	
2	234,386	
3	1,220,000	
4	1,320,000	
5	1,420,000	

gate_id	limit	org_limit
0	0	0
1	215,083	0
2	726,161	0
3	792,176	0
4	858,192	0
5	1,147,228	0
6	1,180,005	0
7	1,223,709	0
8	1,267,414	0
9	1,311,117	0

gate_id	limit	org_limit
0	0	0
1	215,083	215,083
2	726,161	215,083
3	792,176	215,083
4	858,192	215,083
5	1,147,228	215,083
6	1,180,005	215,083
7	1,223,709	215,083
8	1,267,414	215,083
9	1,311,117	215,083

OS_ QTY[**5015-21-ATL**] = 200,000

Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 0] + 200,000* PG[**5015-21-ATL**,**0**] >= 0 -> Works
Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 0] + 200,000 * PG[**5015-21-ATL**,**0**] <= 215,083 * PG[**5015-21-ATL**,**0**] -> constraints the quantity of dec var between 15,083 for gate 0
Q[SSS, **3933**, **ATL**, 416, 1, **5015-21-ATL**, 1] + 200,000 * PG[**5015-21-ATL**,**1**] >= 215,083 * PG[**5015-21-ATL**,**0**] ; constraints the quantity of dec var >= 15,083 for gate 1
Depending on BigM and Demand constraint and discounts, the model will select the gate for program **5015-21-ATL**. If this program gets selected, then it will impact the quantity of program **010-21-ATL** because of equal quantity for programs chaining. Now, for program **010-21-ATL** the gate will be decided based on quantity for brand line 3932 as well in addition to 3933 which got selected automatically because of program **5015-21-ATL.**

Model – Program Maker VS Takers

- Program Takers
  - When we are only considering a very small proportion of all the products which follow under a program and if these considered products also have very small proportion of the total sales then these products will be considered as program takers. This is because these products will utilize/take the gate made/qualified by other products purchases to calculate the discounts
  - So, we take the short term and annual programs and the gates which were applicable in 2021
  - Then we create the optimization model which will use these programs and selected gates to minimize the procurement cost for the in-scope products
  - This means that we are simply assigning the vendors, their programs and the discounts applicable in the model to in-scope products
  - Then the model will tell us what vendor(s) should be chosen so that we get the maximum discounts and the minimum costs
  - We will restrict the total quantity of in-scope plus the out-of-scope products within the selected gates for the programs

- Program Makers
  - We can figure out the proportion of sales of the in-scope products among all the products sales which belongs to a program offered by a vendor
  - Then we basically reduce the program gate limits by multiplying the limits with this proportion
  - So, finally what we are trying to do is that we are trying to meet shortened gates of a program from the quantity recommended for these in-scope product groups and are also assuming that the other out-of-scope products (e.g. Shingles) in the same program will meet the rest of the gate limits (*(1-prop))

Model – Some Points to Consider

- A program can have different discount for each applicable brand line i. A brand line can also come from 2 suppliers/vendors – manufacturer and 2 step vendor. Each product group will then be mapped to all brand lines and the vendors selling these products. This product group will then have a total demand equal to the sum of the demand for all products in this group. So, for a product group f the model will choose which brand line(s) to be bought from which vendor(s) to satisfy the demand of product group f. Also provide the historical proportions of brand line purchased from vendor to be used in non-discretionary demand.
- At any time, t all programs which are applicable to product group f purchased from vendor v at branch b will qualify for the same quantity or product group's cost for discounts calculations. This mean whether a program is national or regional or short term or gated or non-gated, they will all qualify for same quantity to calculate discounts.
- Writing model at program level can be done by calculating the unit cost discount in dollars ($unit\_cost\_discount_{fivbpgt}$) that a program receives. This discount is calculated after all applicable programs are chained from one another.
- For Job Quotes, we will have to predict the proportion of quantity that can be job quotes. Like, we can figure out every month's proportion of job quote demand and apply it to our forecasts. This will result in 2 forecasts – job quotes and non-job quotes. Now, on these job-quotes forecast only the programs which do not qualify for Job Quotes discounts will result in 0 discount and will not chain. However, all the other programs will still chain and will get discounts.
- Similarly, for Directs - as there are program which do not provide discounts on direct shipments.
- For Customer exclusions, we need to calculate the proportion of Customer Excluded Qty and used the remaining proportion as in-scope forecast. The Customer Excluded Qty can be figured out from the list of Customers which follow under non excluded programs.

Model – Unit Cost Discount Calculations

- $unit\_cost\_discount_{fivbpgt}$
  - For the product group f and brand line i from vendor v at branch b and time t
    - Get list of all programs applicable from $program\_group_{fivbt}$ -> $program\_list_{fivbt}$
    - Get chaining level of all programs from $program\_list_{fivbt}$
    - Get dense rank on chaining levels to match programs at same chain level to one rank
      - ❖ Rank 1 - Lowest Chaining Level Number Programs (e.g. Chain Level = 0,0,0)
      - ❖ Rank 2 - Second Lowest Chaining Level Programs (Chain Level = 2,2)
      - ❖ Rank N - Highest Chaining Level Programs (Chain Level = 5,5,5)
      - ❖ Calculate Program Discount or Unit Cost Discount on each dense rank
      - ❖ Programs at any one rank will get the discounts independently of other program at same level meaning they do not chain from itself and hence they won't impact each other's discount.
      - ❖ Programs do not chain if -
        - ❖ chaining Level is 'e' however the program gets discount
        - ❖ or JobQuotes Not included and PO Type 'J' *
        - ❖ or DirectShip not included and PO Shipped to Direct Warehouse*

***** Program which do not provide rebates for Job quote POs and Direct shipments will be handled by Job Quote Proportion and Direct Proportions used in the optimization function. For non-job quote forecast and non-directs forecast, all the applicable programs will provide the discount, whereas for job quote and directs forecast, the programs with JobQuotes not included and DirectShip not included respectively will not chain and not provide any discount as well. We need to think about the cases where a job quote can be direct as well.

Model – Unit Cost Discount Calculations

- $unit\_cost\_discount_{fivbpgt}$
  - For Dense Rank n = 1 to N:
    - If Dense Rank n = 1:

      Total Previous Programs Discount = 0
    - If Chaining Level == 'e': # No chaining for e chains hence no previous programs deduction and accrual:
      - Previous Programs Discount Deductions = 0
    - Else:
      - Previous Programs Discount Deductions = Total Previous Programs Discount

    Cost = $unit\_cost_{fivb}$ (correct UOM especially for cost discount programs or will need merchandising conversion rate)
    - For programs p in Dense Rank = n:
      - $freight\_cost_{fivbpt}$ = Freight Cost = Cost * Is Freight Included * Freight Percent
      - $prompt\_pay\_discount_{fivbpt}$ = Prompt Pay Discount = (Cost - Freight Cost) * Is Prompt Pay Included * Prompt Pay Percent
      - $additional\_discount_{fivbpt}$ = Additional Discount = (Cost - Freight Cost - Prompt Pay Discount) * Additional Discount Percent
      - $self\_chain\_discount_{fivbpt}$ = Self Chain Discount = (Cost - Freight Cost - Prompt Pay Discount - Additional Discount - Previous Programs Discount Deductions) * Self Chain Percent * Guaranteed Discount Percent

      For Quantity Based programs: (non incremental and non-cost based)

      If $program\_group_{fivbt}[program == p \ and \ program\_discount\_type == percent]$:
        - $program\_discount_{fivbpgt}$ = Program Discount[gate = g] = (Cost - Freight Cost - Prompt Pay Discount - Additional Discount - Previous Programs Discount Deductions - Self Chain Discount) * Guaranteed Discount Percent

      Else $program\_group_{fivbt}[program == p \ and \ program\_discount\_type == cost]$:
        - $program\_discount_{fivbpgt}$ = Guaranteed Cost Discount - Freight Cost - Prompt Pay Discount - Additional Discount
      - $unit\_cost\_discount_{fivbpgt} = program\_discount_{fivbpgt}$
      - This level programs discount += Program Discount
    - If Chaining Level != 'e':
      - Total Previous Programs Discount += This level programs discount

Model – Additional Notes

- JobQuotes
  - When for a program Is Job Quotes Included = N and PO Type = J, then that receipt-item do no qualify for the rebates, so we exclude that program from the chain
- Directs
  - When Is Directs Included = N and Warehouse = D, then that receipt-item do no qualify for the rebates, so we exclude that program from the chain
- Cost Discounts
  - Cost based programs neither chain from programs nor have self chaining
  - Need to use conversion rate to convert UOM - Quantity/Conversion Rate
- Percent Discounts
  - Percent based programs chain from programs and can self chain as well