Supermarket Sweep

Presented by:
Sri Ravi Teja Kolipakula
Cassidy Gasteiger
Kalyan Kumar Vulchi

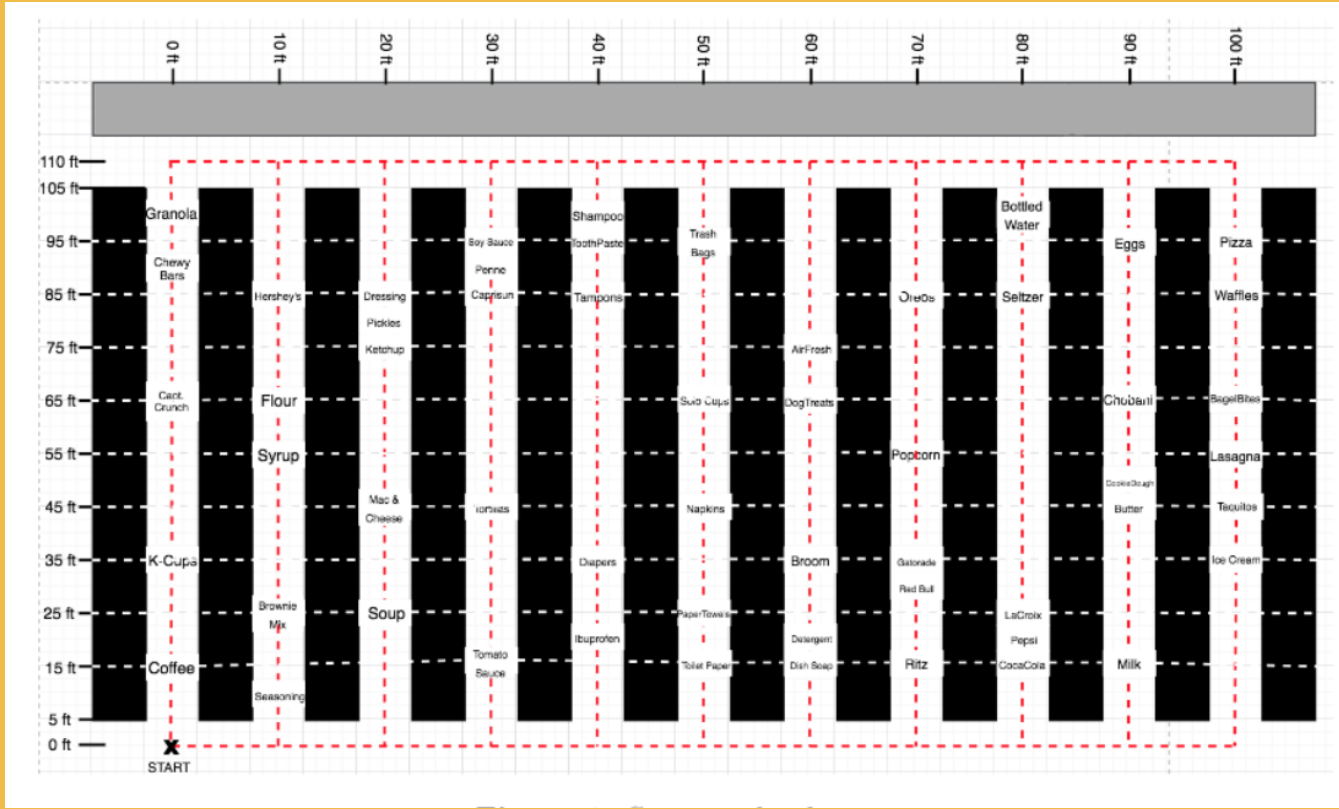
Problem Statement

Supermarket Sweep was a TV game show where contestants had a short amount of time to grocery shop and pick up items, with the goal of maximizing the monetary value of their cart to win prize money. Our goal is to formulate the Supermarket Sweep as an optimization problem.

Problem Parameters:

- 15-item maximum
- 90 seconds
- 56 possible grocery items (can only select each item once)
- Shopper moves at 10 ft/s
- Shopper takes 2s to pick up each item
- Shopper begins at same start and end point

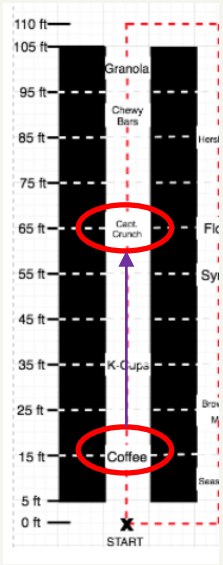
Supermarket Layout



Time to go from *i* to *j*

First, we stored the values of the shortest amount of time it would take for the shopper to go from every item *i* to every item *j* as a parameter d_{ij} for use in our problem formulation.

For items in the same aisle:

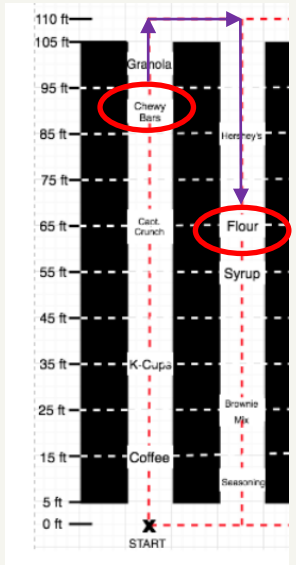


$|i_{ycoord} - j_{ycoord}|$ / 10 feet per second

Example:

$|coffee_y - captcrunch_y|$ / 10
$|15 - 65|$ / 10
50 / 10
$d_{coffee,captcrunch}$ = 5 seconds

For items in different aisles:



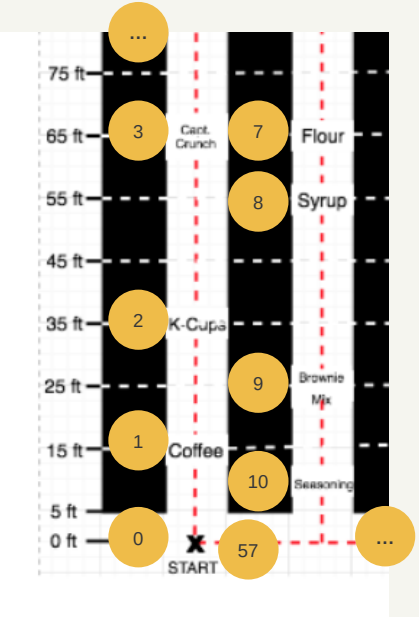
$(\min((i_{ycoord} + j_{ycoord}), 220 - (i_{ycoord} + j_{ycoord}))$ / 10 feet per second) + $|i_{xcoord} - j_{xcoord}|$ / 10 feet per second

Example:

$(\min((chewy_y + flour_y, 220 - (chewy_y + flour_y))$ / 10) + $|chewy_x - flour_x|$ / 10
$\min((90 + 65), 220 - (90 + 65)$ / 10) + (10 / 10)
$(\min(155, 45)/10) + 1$
$d_{chewy,flour}$ = 7.5 seconds

Our Approach

To formulate the Supermarket Sweep optimization problem, we adapted the Traveling Salesman problem formulation with “lazy constraints” to create a maximization problem.



We assigned an index to each item and split the start and end node to create a Hamiltonian path problem:

- Each item *i* = (1-56)
- Start node = 0
- End node (same location as start node) = 57 (n+1)

Problem Formulation

Decision Variables:

X_{ij} = binary variable to decide if a path between two items i, j is selected $\forall\ i, j \in [0,57]$
t_{ij} = total time taken to reach product j including time for (i,j) $\forall\ i, j \in [0,57]$
0 if (i,j) does not belong to the optimal path
y_j = total time taken to reach product j during the journey $\forall\ j \in \{0,\dots,57\}$
u_{ij} = time value of items picked until item j $\forall\ i, j \in [0,57]$ (used to update z_j)
0 if (i,j) does not belong to the optimal path
z_j = total value of items picked until item j $\forall j \in [0,57]$

Given:

d_{ij} = time taken to travel from product i to product j $\forall\ i, j \in [0,57]$
v_i = value of item i $\forall\ i \in [1,56]$

Objective Function:

maximize z_{57}

Path Constraints

PART B.

1	$\sum_{j=1}^{57} x_{ij} \leq 1 \; \forall \; i \in [0, 56]$	The shopper can leave each item only once (end point is excluded)
2	$\sum_{i=0}^{56} x_{ij} \leq 1 \; \forall \; j \in [1, 57]$	The shopper can arrive at each item a maximum of one time (start point is excluded, but end point is included)
3	$\sum_{i=0}^{56} x_{ij} = \sum_{i=1}^{57} x_{ji} \; \forall \; j \in [1, 56]$	If the shopper arrives at an item, they must leave the item (start point and end point are excluded)
4	$\sum_{j=1}^{56} x_{0j} = 1$	Shopper must leave from the starting point
5	$\sum_{i=1}^{57} x_{i57} = 1$	Shopper must end at the end point
6	$\sum_{i=1}^{56} x_{i0} = 0$	Shopper should not go to the starting point at any time during the sweep
7	$\sum_{j=0}^{56} x_{57j} = 0$	Shopper should not start from the end point at any time during the sweep
8	$x_{ii} = 0 \; \forall \; i \in [0, 57]$	Shopper should not go to the same item twice

Time Constraints

PART B.

9 $t_{ij} \leq M x_{ij} \; \forall (i, j) \in [0, 57]$ where M is a large number

If $x_{ij} = 0$, then t_{ij} must also be 0

10 $y_0 = 0$

Initial movement time = 0

11 $\sum_{i=0}^{56} t_{ij} = y_j \; \forall j \in [1, 57]$

Time until item j is equal to the sum of t_{ij}

12 $\sum_{k=1}^{57} t_{jk} = y_j + \sum_{k=1}^{57} d_{jk} x_{jk} \; \forall j \in [1, 57]$

Time until item k is equal to time required to move to item j plus time taken from j to k if path (j,k) is selected

Value Constraints

PART B.

13 $u_{ij} \leq M x_{ij} \; \forall \; i, j \in [0, 57]$
where M is a large number

If $x_{ij} = 0$, then u_{ij} must also be 0

14 $z_0 = 0$

Initial value of all items in the cart = 0

15 $z_j = \sum_{i=0}^{56} u_{ij} \; \forall \; j \in [1, 57]$

z_j is equal to the running value of all items in the cart

16 $z_j + \sum_{k=1}^{57} v_k x_{jk} = \sum_{k=1}^{57} u_{jk} \; \forall \; j \in [1, 57]$

Update u_{ij} to calculate running value of items picked in cart

Problem Constraints

PART B.

17 $y_{57} + (\sum_{i=0}^{56} \sum_{j=1}^{57} x_{ij} - 1) * 2 \leq 90$

Total time must be less than or equal to 90 seconds

18 $\sum_{i=0}^{56} \sum_{j=1}^{57} x_{ij} \leq 16$

Total number of items picked less than or equal to 15
(Added 1 to include end point)

Non-Negativity and Integrality Constraints

19 $t_{ij} \geq 0 \quad \forall i \in [1, 57] \quad \forall j \in [1, 57]$ Non-negativity constraint

20 $u_{ij} \geq 0 \quad \forall i \in [1, 57] \quad \forall j \in [1, 57]$ Non-negativity constraint

21 $x_{ij} \in [0, 1] \quad \forall i \in [1, 57] \quad \forall j \in [1, 57]$ Integrality constraint

An Optimal Solution

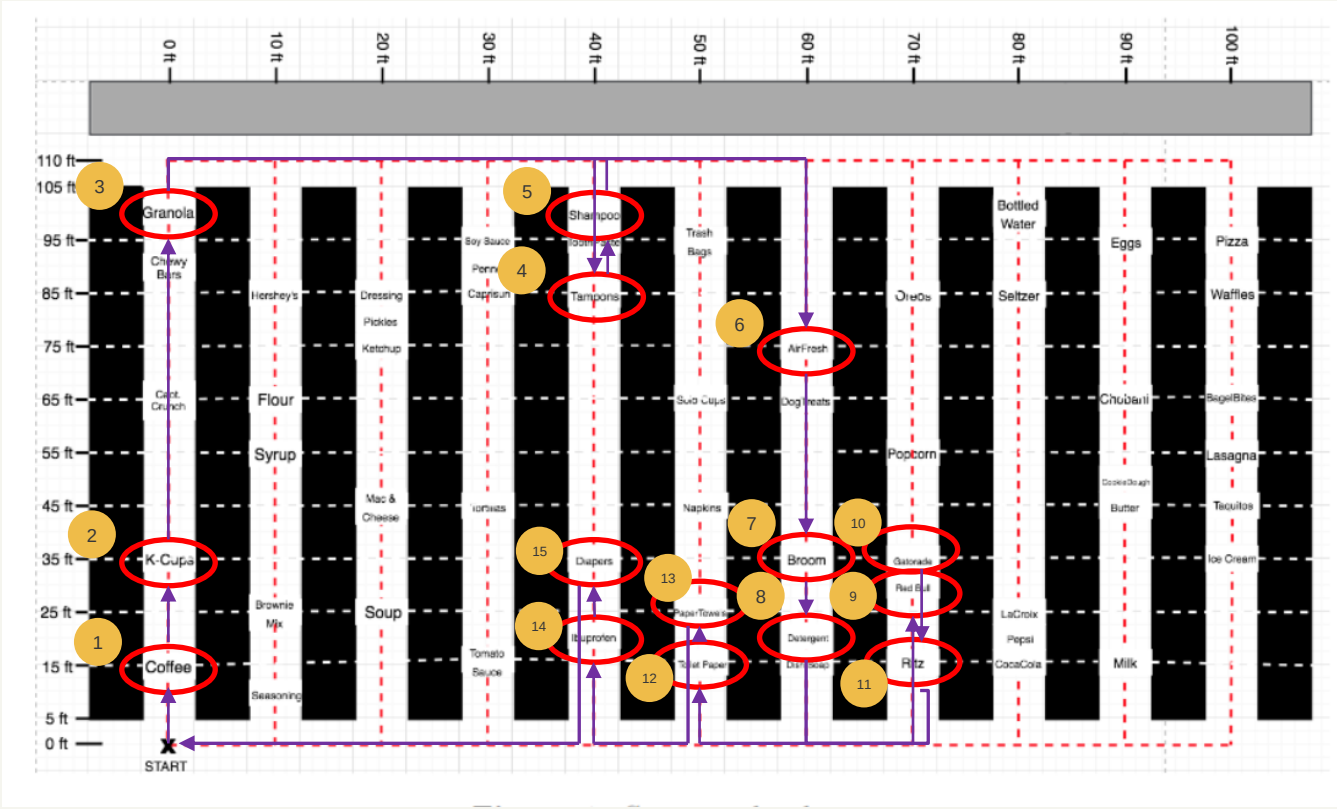
Total Cart Value

$143.85

Items Selected

- Coffee Beans
- K-cups
- Granola
- Tampons
- Shampoo
- Air Freshener
- Broom
- Detergent
- Redbull
- Gatorade
- Ritz
- Toilet Paper
- Paper Towels
- Ibuprofen
- Diapers

Optimal Path

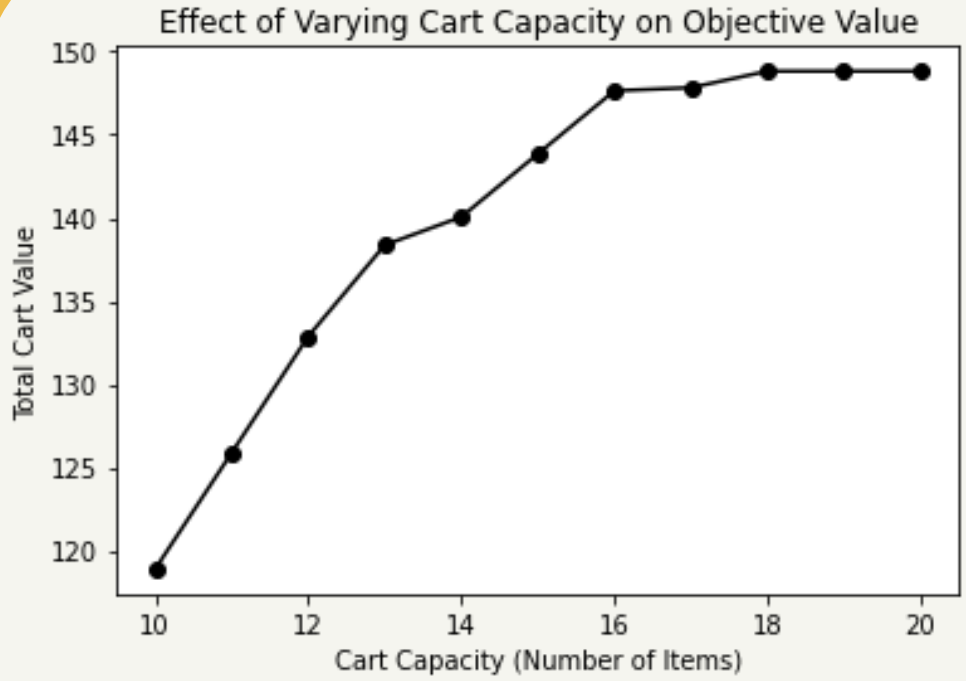


Total Time Available

- Increasing the time available to shop intuitively increases the total cart value; with more seconds available, the shopper has more time to place higher-value items in their cart.
- Our chart levels out at 110 second. Past that point, increasing the time available to shop does not increase the total value of items in the cart.
- Presumably, with 110 seconds, the shopper has enough time to place the 15 highest-value items in their cart, and the maximum objective function value is constrained by cart capacity rather than time.
- The highest possible value for 15 items from this grocery store is $154.85.

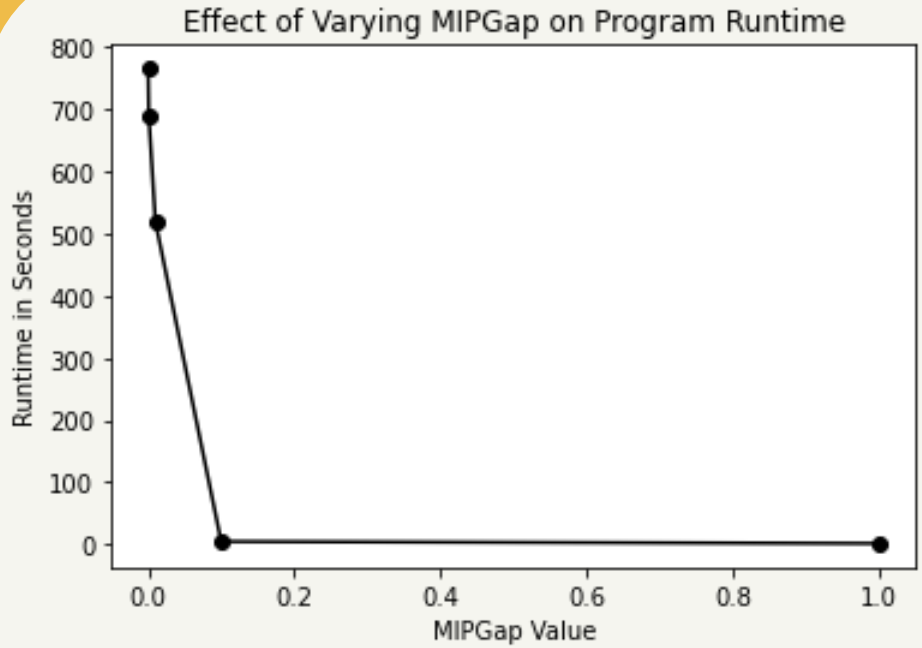
PART E.

Cart Capacity



- Increasing the cart capacity (total number of items allowed to be selected as part of the sweep) intuitively increases the total cart value; the shopper can now select more items to assemble a higher-value cart.
- However, our chart levels out at 18 items. Past that point, increasing the number of items allowed in the cart does not increase their total value.
- Presumably, the shopper runs out of time to place more items in their cart. Now the maximum objective function value is constrained by the time constraint rather than cart capacity.
- The highest possible value for items reachable within 90 seconds is $148.82.

MIPGap



- MIPGap is a parameter in Gurobi that dictates how precise the solution should be. The default parameter is .0001, meaning we receive a solution once the objective value is within .01% of the optimal value of the problem.
- The graph demonstrates how much more quickly Gurobi output a solution by decreasing MIPGap parameter.
- The default .01% took nearly 13 minutes. Increasing MIPGap to .1% still took about 12.5 minutes. A value of 1% took only 5.45 seconds to run, and 10% took 1.36 seconds. However, 1% and 10% did not find the optimal solution.
- If we have a model with many constraints and don't need a very precise solution, increasing the MIPGap parameter is a good option to decrease runtime.

Thank you!