Appendix A: Genetic Algorithm

Appendix A.1: Overview

The underlying mechanism of store assignment to putwalls is guided by an optimization meta-heuristic called a genetic algorithm. As an optimization algorithm, it will be able to minimize a given function, in this case, the distance traveled in the pick module, and the number of store reassignments made. The genetic algorithm is so named due to its biologically inspired mating and mutation functions to "select" the "fittest" solutions. The parallel is further observed as the process rests on a level of randomness, but overall, this algorithm is more efficient than random search and exhaustive search algorithms.

There are several components to the search process:

1. Fitness function to be optimized
2. Constraints
3. Initial arrangement
4. Initial solution generation
5. Selection of which solution will move forward
6. Crossover to produce next generation of solutions
7. Random mutation of solutions in the new generation

Appendix A.2: Fitness Function to be Optimized

The fitness function is the rationale by which certain assignments are deemed "better" or "fitter" than others. The fitness function is defined as follows where i represents the index of the solution generated:
Fitness of solution_i =

$$\text{Pick module distance for solution}_i + 300 * (\text{Number of putwall moves to arrange solution}_i)$$

1. **Distance travelled in the pick module:** The distance estimate is calculated for each putwall within each of the nine zones of the pick module. The distance is traveled is calculated by replicating the S-path and the distance between each location to the next.
2. **The number of putwall moves made:** Each change in the putwall setup carries with it a systemic cost of approximately 1 minute per move. Therefore, a solution that swaps 200 locations is less desirable than a solution that changes just 20. The 300 multiplier enforces that each move must save a minimum 300 feet of picking distance.

Appendix A.3: Constraints

Solutions are immediately disqualified if they violate any of the following constraints:

1. **The number of order lines per putwall:** If the number of order lines on a putwall exceeds the set parameter.
2. **The number of stores per putwall:** If the number of stores on a putwall exceeds the maximum of capacity of the putwall.
3. **The maximum number of moves:** If the solution has more moves than the maximum number of moves input for a given day.

Appendix A.4: Initial Arrangement

The initial arrangement of the putwall is used as a starting point for potential solutions. A simple example of a putwall arrangement is shown below, using only six customers (1, 2, 3, 4, 5, 6) and three putwalls (1, 2, 3).

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 3 \end{bmatrix}$$

In this matrix, the first index corresponds to the first customer. The number value of that index corresponds to the putwall assignment for that customer. In this example, customers one, two, and three are all assigned to putwall one, customers four and five are assigned to putwall two, and finally, customer six is assigned to putwall three.

Appendix A.5: Initial Solution Generation

Once the initial arrangement of store assignments to a putwall is known, a random number of putwall swaps are made to the initial assignment to create 100 new potential solutions.

Appendix A.6: Selection of which Solutions will Move Forward

Each of the 100 new solutions is then run through the fitness evaluator to judge its distance savings and cost of implementation. To carry the most "fit" solutions forward to the next generation, i.e. the most return on distance savings with the least number of moves, selection of new solutions is defined as the probability relative to the objective value. This process is done by first normalizing the objective values with the equation:

$$\text{Normalized Fitness of solution}_i = \text{Max}(f(\text{i})) - f(\text{i}) + 1$$

Then selection probability is completed with the equation:

$$\text{Probability that solution}_i \text{ is selected} = \frac{\text{Normalized Fitness Solution}_i}{\sum \text{Normalized Fitness Solutions}}$$

Solution Number	Solution	Fitness Value	Normalized Fitness Value	Selection Prob
1	1 1 1 2 2 3	25	1	0.019
2	1 2 2 2 1 3	10	16	0.296
3	3 2 1 2 2 3	10	16	0.296
4	3 3 1 1 2 3	5	21	0.389

Table 8: Example of calculation of selection probability of four solutions

Appendix A.7: Crossover to Produce next Generation of Solutions

After solutions are selected to undergo crossover, crossover randomly creates new solutions by parts of solutions with one another. A simple example of this is shown with two solutions with six customers and three putwalls.

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & | & 2 & 2 & 3 \\ 2 & 2 & 3 & | & 1 & 1 & 1 \end{bmatrix}}_{\text{Parent solutions}} \rightarrow \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 3 & 2 & 2 & 3 \end{bmatrix}}_{\text{Offspring solutions}}$$

This process continues until there are 70 new cross-over solutions.

Appendix A.8: Random Mutation of Solutions in the New Generation

The 70 new cross-over solutions undergo mutation at a fixed probability. A store has a random chance of being assigned a new putwall. An example of mutation is shown below.

$$\underset{\text{Original solution}}{\begin{bmatrix}1 & 1 & \boxed{1} & 2 & 2 & 3\end{bmatrix}} \rightarrow \underset{\text{Mutated solution}}{\begin{bmatrix}1 & 1 & 3 & 2 & 2 & 3\end{bmatrix}}$$

After mutation, all the solutions are evaluated by fitness function and the process is repeated until the number of iterations has been completed.

Appendix A.9: Current Parameters

The current model parameters were determined by hypertuning parameters with cross validation.

Parameter	Setting
Maximum Number of Iterations	100
Maximum Number of Putwall Moves	30
Mutation Probability	.01
Crossover Probability	0.3
Crossover Type	Uniform

Table 9: Model parameters