Blue Yonder Demand and Supply Problem

Team

Team Name

Deku

College

Thapar Institute of Engineering and Technology

Team Members

Raghav Kumar

Problem Solving and Approach

Problem : Demand and Supply

Given a list of stores, a list of products, a time period, a demand range and probability distribution for each store, product, time defined on the demand range find the optimal trucking that minimise empty trucks and maximises throughput by utilising the DC and factory intelligently.

Purposed Solution and Approach:Monte Carlo Tree Search

At any day t, we receive an order for each store, product in the range (demand min, demand max). And then we make a choice of what products to generate from the factory, what to put into DC, and what to ship via trucks.

This a classic example of a zero-sum two player game, where the demand is stochastic, and our decisions are optimal i.e like in the game 2048.

However, unlike 2048 the branching factor is exponentially high

$$O(\sqrt{(Demand\ Rannge)^{|products|*|stores|} * 2^{|stores|} * (|products|)^{DC\ Capacity}})$$

Because of this, even at depth of 2 the number of possible states will become practically impossible. But algorithms like expecimax require 4-6 depth trees before they start giving meaningful solutions.

For this specific problem one of the most practical and used solution is a Monte Carlo Tree Search.

It works by exploring only a subset of the meaningful region.

Benefits of my solution

Trivially parallelizable

Although the current implementation is a single threaded C++ object, that is only for experimenting and prototyping. MCTS is a tree traversal and tree expansion algorithm, and as such is the most trivial example of a parallelizable algorithm. It can be parallelised along GPU using CUDA programming and HPC using languages like Chapel.

Compute Once, infer many times

Once the tree has been generated, you can serialise the tree and store it. For the next day, you don’t have to regenerate the entire tree, the current situation would be a grandchild of the root node. Making that grandchild the root-node, all the children of this grandchild will be useful, and only a small portion of the original tree will need to be generated.

Non presumptuous algorithm

MCTS make no assumptions about the problem it solves. All it needs are 2 things, a function that generates a random next node from the current node that is capable of generating all possible nodes(lets call this function smart_generator), and a fitness function that can evaluate leaves, nothing else is needed. So in the future if there are added constraints, only and only smart_generator will have to be re-written, the rest of the code will remain exactly the same.

Heuristic Intelligence can be added in the future

The MCTS algorithm loop consists of 4 steps. Selection, Expansion, Simulation and Back propagation. During the expansion phase the following formula is used to evaluate a nodes fitness (assuming Upper Confidence Bound 1 algorithm is used to solve the Multi Armed Bandid Subproblem as opposed to epsilon-greedy, which is the case in my implementation)

$$\frac{success}{my\ simulations} + \sqrt{\frac{c_1 * \ln{(parent\ simulations)}}{my\ simulations}}$$

where c1 is the exploitation constant, the higher it is the more we exploit. However it can be extended as

$$\frac{success}{my\ simulations} + \sqrt{\frac{c_1 * \ln{(parent\ simulations)}}{my\ simulations}} + \frac{c_2 * P_i}{my\ simulations + 1}$$

Where Pi is domain specific knowledge, so if the future some heuristic measure are created to prefer specific nodes (i.e we wanna make it so that raining days have lesser number of trucks and we have a probability distribution of rain, we can use this factor to explore those nodes more) we can extend the algorithm to work with knowledge.