The Trains problem is a modified version of “Traveling salesman problem”. Problems of this type should not be solved by classical graph search algorithms, because for a small number of vertices it may be difficult or even impossible to cross all possible routes. The number of such routes will be equal to n!, where n - is the number of vertices. That’s why this problem is usually solved using Heuristic algorithms. These are algorithms that give a very good result but not the best.

It is uncomfortable to store data for the train task in the form of a table, because two stations can have many different trains between them. That’s why I chose the ant algorithm.

This algorithm simulates the work to find the best way to eat an ant swarm in nature. Ants leave their pheromones every time they go through the route. The next ant is most likely to follow a route close to the marked one, or choose it accidentally. Pheromones gradually wear out. After a while, one of the fairly good routes will be the most visited by ants, which means this route will have the most pheromones.

So I used the following mathematics to implement this phenomenon programmatically:

- ants is the number of iterations of a route selection
- the number of the next train in the route depends on the ant’s desire to go there
- $desire_i = ph_i^a * w_i^b$, where ph - pheromone on i-th train; w - 1/weight of i-th train(price or time); a, b - pheromone and weight factors respectively
- $P_i = \frac{desire_i}{\sum_{k=0}^{trains\ amount} desire_k}$, where P_i is is the probability of choosing i-th train
- a number from 0 to 1 is generated for probabilistic selection. The train whose P_i was closer to this number would be selected.
- initial pheromones are arranged equally, then change after each repetition of the ant run.
- $ph_{newi} = ph_i * (1 - e) + \Delta ph_i$, where e - is evaporation for pheromone, Δph_i - added to i-th trains pheromones
- $\Delta ph_i = \frac{Q}{W}$, where Q - It’s some coefficient for convenience, W - total weight (sum of prices or times) of the train track.

Structures:

- **train.Train**

  Represents a structure with fields for storing all the information about the train. All fields are exported.

  methods:

  NewTrain(string) (train.Train, error)

  return new train instance and errors, use string to initialize

  (*Train)String()string

  implement Stringer

- **graph.Graph**

  Inherits map[int][]int, represents key-value object, where key is number of station, value - slice of the train indexes that start at this station

  methods:

  NewGraph([]train.Train)*graph.Graph

  create and return pointer to new graph for given slice of trains

  (*graph.Graph)AddNode(int, int)

  add for key-station slice of train indexes new index

  (*graph.Graph)GetNodes()map[int]bool

  return all stations

  (*graph.Graph)String()string

  implement Stringer

- **ants_algorithm.AntsSwarm**

  Represents a set of data and settings necessary for the work of the ant algorithm. All fields are not exported.

  methods:

  NewAntsSwarm([]train.Train, *graph.Graph, antsAmount, a, b, evaporation, q) *AntsSwarm

  create and return pointer to new ants swarm for given data and settings

  (*AntsSwarm) ChangeSetting(antsAmount , a, b, evaporation, q)

  change algorithm settings

  (*AntsSwarm) FindShortestRoute(repeat) ([]int, time.Duration)

  find and return a short route and its duration with a given repeat.

  (*AntsSwarm) FindCheapestRoute(repeat) ([]int, float32)

  find and return a cheap route and its result price with a given repeat.

  (*AntsSwarm) pheromoneInitialization(pheromone) [][2]float64

  initializes and returns the pheromone table.

  (*AntsSwarm) runTime(pheromones, bestRoute, bestTime)

  implements an ant’s move to find a short route

  (*AntsSwarm) runPrice(pheromones, bestRoute, bestPrice)

  implements an ant’s move to find a cheap route

  FindNearestT(keys, den, key) int

  return value from keys where key is nearest. den is denominator (route weight)

  getDurationBetween(start, end) time.Duration

  return duration between two times.

Other function:

UploadData(string, string) ([]train.Train, error)

Read data in .xlsx file by given file name and sheet name, then create and return slice of trains

How to use:

Cope repository and run trains_problem.exe