Delivery Scheduling

Metaheuristics for Optimization/Decision Problems

- Delivery system composed of **one truck** with **constant speed**.
- For a given set of packages, each with their own delivery coordinates, design algorithms to **optimize** the **delivery order of packages**.
- Packages may be of different types: **normal**, **fragile** or **urgent**.
- Must **minimize travelling costs**.
- Must maximize reputation, by **minimizing** both **damage to fragile packages** and **delayed deliveries of urgent packages**.

João Ramos Marco Costa Tiago Viana

Problem Formulation

$\mathbf{PackageSet} = \{p_0, \ldots, p_{n-1}\}, \text{where } |\mathbf{PackageSet}| = n$

$\mathbf{Solution} = (s_0, \ldots, s_{n-1}) \in \mathbf{PackageSet}^n, \text{ where } \forall_{i \neq j}\ (0 \leq i < j < n),\ s_i \neq s_j, \text{ and } |\mathbf{Solution}| = n$

$neighbour(\mathbf{S}) \rightarrow \text{For a given } s_i, s_j \in \mathbf{S}, \text{ swap } s_i \text{ and } s_j$

$mutation(\mathbf{S}) \rightarrow \text{ For } i = 0 \text{ to } \frac{n}{2},\ s_{2i}, s_{2i+1} \in \mathbf{S}\ , \text{swap } s_{2i} \text{ and } s_{2i+1} \text{ with probability } P_{swap}$

$crossover(\mathbf{S_1}, \mathbf{S_2}) \rightarrow \text{Iterate through } \mathbf{S_1} \text{ and } \mathbf{S_2}, \text{picking a package from either one with equal probability}$
$\text{Add it to the child if not already present}$

Hard Constraints

- Delivery truck **starts at origin**.
- Only **one location** can be visited at a time.
- **Routes** between **all** delivery locations are **available**.
- The driver drives at **60km per hour** and takes **0 seconds** to deliver the goods.
    - Both these constants are **parameters** of the problem and can be changed for each instance.

Problem Formulation | Evaluation Function

For a given **Solution**, $TotalCost = w_{TravellingCost} \cdot TravellingCost + w_{DamageCost} \cdot DamageCost + w_{DelayCost} \cdot DelayCost$

$$TravellingCost = C_{km} \cdot \sum_{i=0}^{n-2} d(s_i, s_{i+1}), \text{ where: } s_i, s_{i+1} \in \textbf{Solution}$$

C_{km} is the travelling cost per km

$d(s_i, s_j)$ is the distance between delivery locations of packages s_i and s_j

$$DamageCost = \sum_{i=0}^{n-1} d_i \cdot Z_i, \quad \text{where: } Z_i \text{ is the cost of damaging package } s_i, \begin{cases} Z_i > 0, & \text{if } s_i \text{ is fragile} \\ Z_i = 0, & \text{otherwise} \end{cases}$$

$$\begin{cases} d_i = 1, & \text{with probability } P_{damage} \\ d_i = 0, & \text{otherwise} \end{cases}$$

$$P_{damage} = 1 - (1 - X)^{d_{s_i}}$$

d_{s_i} is the distance travelled in kms by package s_i

X is the probability of a fragile package being damaged per each km travelled

$$DelayCost = C_{delay} \cdot \sum_{i=0}^{n-1} delay_{s_i} \cdot u_i, \quad \text{where: } s_i \in \textbf{Solution}, \begin{cases} u_i = 1, & \text{if } s_i \text{ is urgent} \\ u_i = 0, & \text{otherwise} \end{cases}$$

C_{delay} is the cost per minute of delay

$delay_{s_i}$ is the delay of package s_i in minutes

Implementation

- **Programming Language** - Python
- **Development Environment** - Python scripts in VSCode. Considering migrating to Jupyter Notebooks later.
- **Data Structures**
  - **Package** - Represents a package to be delivered. Stores the coordinates of the package's delivery location and the package type. For fragile packages, the breaking change and breaking cost are set. For urgent packages, a maximum delivery time is set.
  - **Delivery Schedule** - Represents an instance of the problem to be solved. Stores the set of packages to be delivered and the constants related to the evaluation function.
- **Libraries** - NumPy, MatplotLib and PyGame.

Optimization Algorithms

- Hill Climbing
  - First Accept *(Already implemented)*
  - Best Accept *(Already implemented)*
- Simulated Annealing
- Tabu Search
- Genetic Algorithms

Bibliography

- Stuart Russel, Peter Norvig - Artificial Intelligence: A modern Approach.
- Delivery Scheduling. Retrieved from https://drive.google.com/file/d/1-A85i8haeQQSYkRILF0uYZOZ0Niba0zJ/view. Accessed March 3, 2024.