Google HashCode 2016 Drone Delivery Optimization Problem

A. Abreu, L. Tavares, M. Duarte

Faculdade de Engenharia da Universidade do Porto
Informatics and Computing Engineering Department

{*up201800168, up201809679, up201909936*}*@fe.up.pt*

March 19, 2021

Overview

Work Specification

The Google HashCode 2016 Delivery problem is an extension of a Vehicle Routing Problem (VRP). Thus, an optimization problem. The problem consists of routing drones to deliver orders, taking products from warehouses to customers.

An order has a single customer and a list of products. Drones can carry multiple products from warehouses to customers or between warehouses as long as their total weight does not exceed the maximum load capacity of the drone. Hence, products from the same order do not need to be delivered by the same drone.

Work Specification

Drones take one turn to load or deliver a specified number of items of a specific product type, and take d turns to go from a place A to B, where d is the lowest integer greater or equal than the euclidian distance between A and B. In addition, drones have the option to wait for w turns to actually move.

Finally, the objective is to deliver as many offers as possible within n turns, where n is an arbitrary number.

Solution Representation

Each order is split into products, defining an identifier for each product. Each gene/operation represents a path taken by a product P made by a drone D to a destination F, (P, D, F).

This representation is used for local search and genetic algorithms, explained after.

Example of a list of operations:

(P1,D1,W1)	(P2,D3,H2)	(P3,D0,H1)	(P4,D2,H0)

P_i - product i
D_j - drone j
W_k - warehouse k
H_l - house l

The initial solution takes orders in ascending order of size and indicates the necessary products that are in the nearest warehouse. Then, for each product, a random drone is selected, and the order in which these products are delivered is also calculated at random.

Neighbourhood/Mutation

Given the similarity between the problem definition for both solution approaches, the neighbourhood (used in local search) and the mutation (used in genetic algorithmics) functions, are similar and as follows:

- Exchange of position of two operations, as long as the order of operations for the same drone is amended as well;
- Assignment of an operation to a different drone.

Crossover Functions

Since the problem can be interpreted as a more specific VRP, classical crossover functions [4] that take the order into account can be applied, such as:

- **One Point Crossover** Given two parents, a random cut point is selected in the same place for both. The left part is taken from the first parent, completing the rest with non repeated genes from the second parent. Vice versa for a second offspring;
- **Order Crossover** builds offspring by choosing a subtour (between two random cut points) of a parent and preserving the relative order of bits of the other parent. Vice versa for a second offspring. [2, 3]

A more in-depth review on these crossovers is recommended and can be checked clicking here.

Chromosomes and Crossover Restrictions

For a correct and logical use of the genetic algorithms, some restrictions are required:

1. A gene that transports a product to a warehouse cannot appear after a gene that transports the same product to its final destination by the same drone;
2. A chromosome cannot have genes that transport a product to a warehouse it has been before;
3. Two genes are deemed repeated if their allele presents the same product and destination;
4. A chromosome must have one and only one gene that transports a product to its final destination.

Rigid Constraints

1. Products can only be taken from warehouses;
2. The sum of the products P delivered at a client C must be less or equal than the products ordered by the client C;
3. The total weight of the drone's cargo can not exceed the drone maximum capacity. Let $S_d = (P_1, P_2, \ldots, P_n)$ be the set of products assigned to a drone d, the following expression must be verified,

$$\sum_{p=1}^{|S_d|} w(P_p) \leq c_d,$$

where $|S_d|$ is the number of products assigned to the drone d, $w(P_p)$ is the weight of the product P_p, and c_d the maximum capacity of the drone d.

Fitness Function

The fitness function takes into consideration orders completed before t turns. For an order to be completed, all its items have to be delivered to the customer's house.

$$\text{maximize } f(t) = \sum_{i=1}^{N} g(t,i)$$

$$g(t,i) = \begin{cases} ceil(\dfrac{t - t_{taken}(O_i)}{t} \times 100), & \text{if } t < t_{taken}(O_i) \\ 0, & \text{if } t \geq t_{taken}(O_i) \end{cases}$$

where t is the total time of the simultation, $t_{taken}(O_i)$ is the time taken to deliver the order i, and N the number of orders that have been completed.

Local Search Implementation

Two different heuristics were used for local search, Hill Climbing and Simulated Annealing, both of them receive the stopping condition which is the number of iterations. Apart from that, the Simulated Annealing metaheuristic can be customized with the following configurations:

- **Initial temperature:** with 1000 as default;
- **Temperature schedule:** the decreasing rate of the temperature, with 0.9 as default.

Genetic Algorithm Implementation

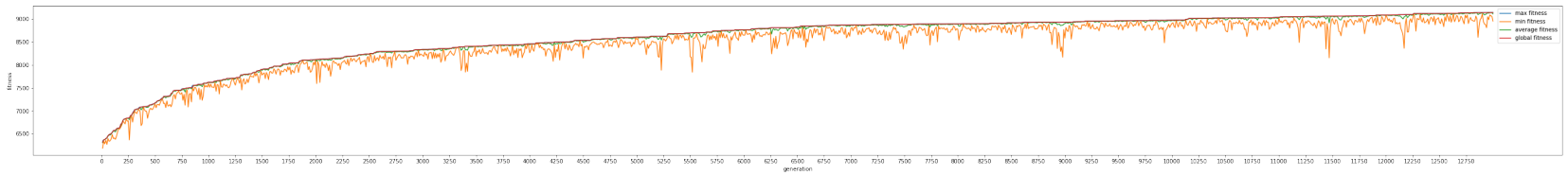
The genetic algorithm is also configurable upon calling. Its stopping conditions are either the total number of iterations, the number of iterations without improving, or a given time. The algorithm can also be customized with the following configurations:

- **Population size:** with 30 as default;
- **Generational:** if it generates enough offsprings to replace the previous population, or if only 2 are generated and the weakest existent chromosomes are dismissed, generational as default;
- **Mutation probability:** the probability of mutation upon the generation of a offspring. Mutations modify a random number of genes below 2% of the size of the chromosome. The default mutation rate is 20%;
- **Selection method:** either tournament selection, where the number of participants per tournament can be specified, or roullete selection. The default is tournament selection with a third of the population per tournament;
- **Crossover:** either one point or order crossover, both specified above. The default is order crossover.

Experimental Results for the Genetic Algorithm

The results below were obtained with the file **custom.in** with the following configurations:

- **Selection method:** Tournament with 15 competitors
- **Crossover:** Order
- **Generational:** Default (True)
- **Population size:** Default (30)
- **Mutation rate:** 0.15
- **Max improveless generations:** 150

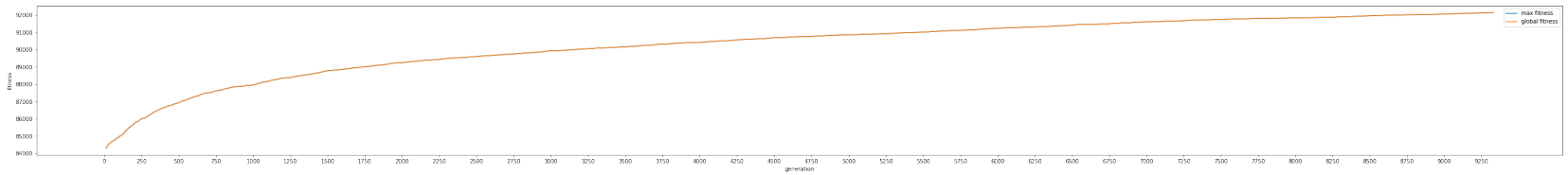


Click here for high resolution.

Experimental Results for the Genetic Algorithm

The results below were obtained with the file **busy_day.in** with the following configurations:

- **Selection method:** Tournament with 15 competitors
- **Crossover:** Order
- **Generational:** Default (True)
- **Population size:** Default (30)
- **Mutation rate:** Default (0.20)
- **Max improveless generations:** 200



Click here for high resolution.

Experimental Results for the Genetic Algorithm

A more in-depth study on these problemas and algorithms is made here and its reading is highly advised for a better understading of their implementation and results.

Conclusion

Given the implementation and the implementation of the algorithmic solution, some conclusions can be extrapolated:

- Ideally, the local search should algorithms generate all neighbours before making conclusions, but this can be very power consuming thus, setting a maximum number of neighbours per iteration, returns faster and overall better results;
- Simulated annealing returns better results if the temperature decreases slowly, otherwise it is very similar to hill climbing;
- The crossover and mutation functions depend a lot on the problem. For problems such as TSP and VRP (such as this one), crossovers based on order return better results than other traditional crossovers;
- These optimization algorithms depend a lot on the initial solution, if it is really optimized, the algorithms may find it difficult to escape local optima, on the other hand, if it is too far off the good solutions, the algorithm may take a long period to get to good solutions, and when time is an important factor, it generates not so good solutions, especially on very large problems.

References

Raphael Reitzig (2018)

Google Hashcode 2016: Our approach

link

Abid Hussain, Yousaf Shad Muhammad, M. Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry, and Showkat Gani5 (2017)

Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator

link

Lawrence Davis

Applying Adaptive Algorithms to Epistatic Domains

link

Varun Kumar S G, Ramasamy Panneerselvam (2017)

A Study of Crossover Operators for Genetic Algorithms to Solve VRP and its Variants and New Sinusoidal Motion Crossover Operator

link