A GRASP-based search technique for scheduling travel-optimized warehouses in a logistics company.

Abstract

In many transport companies, one of the main objectives is to optimized the travel cost of their fleet through develop scheduling procedures. Other objectives are related to delivery time, fuel savings, etc. However warehouse stock management is not properly considered during the scheduling process due to the high number fo constraints that must be satisfied. In this paper, we combine the warehouse stock management problem and the routing problem to be applied in a real company that allows negative stocks in their warehouses. The proposed multi-objective problem is modeled and solved by a greedy randomized adaptive search procedure (GRASP). In this case, we made some tests to tune the proposed algorithm to achieve a good solution according to the company specification. The results show that the proposed GRASP outperforms current search greedy technique used in the company. Concretely, the stock balancing is improved up to 82% and the computational time is decreased between 50% and 62%.

Introduction

Nowadays, transport companies focus on obtaining automatic forecasts and order planning within a given time frame. Many scheduling techniques of different nature can be found in literature to solve this kind of problems. There exist a set of problems that bring together all cases of graph-related problems in the context of transport design. The Multiobjective transportation network design provides a framework that lists all types of transportation problems together. For this context a first level taxonomy is developed in which methods and techniques are grouped by mathematical structure or purpose of the problem formulation (Current and Marsh 1993). The Multiobjective Transportation Network Design is a very extensive group of problems that try to address everything from transportation problems. In this paper we will focus on two of them, the vehicle routing problems (VRP) and the assignment problems (AP).

VRPs can be represented as theoretical problems in graphs. Given a complete network $G = \{V, A\}$ in which V is a list of vertices and A is a list of arcs. Most problems represent the zero vertex as the output vertex and the rest of the vertices as customers to be passed through to deliver an order. The list of arcs is made up of i, j pairs that connect two vertices, these arcs have a cost associated with going from the i vertex to the j (Eksioglu, Vural, and Reisman 2009). From this point on, the wide development of solutions for diverse problems has generated a very wide literature. Mainly, VRP problems are multi-target problems in which there are certain features that combine with each other to solve problems. However, there is a more specific set of real life problems that traditional VRPs cannot solve, such as: the Open VRP, the Dynamic VRP and the Time-Dependent VRP.

In the Open VRP (OVRP), the main feature that interests us is that the vehicles are not forced to return to the starting point, so this type of problems seeks to minimize the number of vehicles used and the total distance traveled (e.g., (Subramanian, Uchoa, and Ochi 2013; Cao, Lai, and Yang 2014)). Some real life problems can be modelled as OVRP (López-Sánchez et al. 2014; Salari, Toth, and Tramontani 2010).

Taking into account the rapid evolution of technologies, we can obtain a large amount of data in real time, not only the status of the vehicle or the order, but also the time of delivery to the customer, the status of the route that will use our vehicle, etc. That is why the development of Dynamic VRP (DVRP) is so important, since it allows us to assume changes in the solutions obtained and modify them with the updated data generating optimal solutions until the last moment (Barkaoui and Gendreau 2013; Pillac et al. 2013).

Another problem affecting VRPs is that the travel time between one vertex and another is deterministic, but in real life the travel time between vertices is not deterministic as there are many variables that can affect it. These problems are defined as Time dependent VRP in which it is assumed that the trip is not deterministic and does not have constant times in the trips between vertices (Li, Leung, and Tian 2012).

The Assignment problems (AP) deals with the question of how to assign n items (vehicles, machines, agents, ...) to m different tasks or procedures. However, it is normal to take into account some limitations that allow to focus the algorithm on a specific case and make it more specific (Burkard, Dell'Amico, and Martello 2009). One of the most used as-

signment problems is called quadratic assignment problem (QAP). This kind of AP allows not all objective functions to be linear, focusing on assigning a set of n elements in m locations. The cost of QAP is composed of a series of smaller costs that are minimized to find the minimum cost.

Considering the existing QAP, this paper is focused specifically on the storage assignment location problem (SLAP). The SLAP involves a group of problems that refers to the allocation of products in a storage space and the optimization of material handling costs, among others. Usually, the SLAP problem depends on parameters such as the distance between source and destination about items and warehouses, the availability of stock, the physical characteristics of the items, the stock refill price, etc. The main optimisation approaches are related to storage space usage and the cycle time of order picking operations, taking into account constraints such as available storage capacity, order picking resource capacities and dispatch policies. (Syed-abdullah, Abdul-rahman, and Mauziah 2018)

In this paper, we focus on a logistic problem provided by a company that focuses its activity on the rental of a wide variety of reusable packaging for the storage and transport of a wide variety of food products. The main priority of this company is to provide containers to allocate fruits, vegetables, fishes, etc in supermarkets that need a wide distribution throughout the country.

The containers are manufactured to simplify the process of transporting, storing and linearly arranging consumer goods (see Fig 1). Today the company has multiple containers, manufactured under the idea of eco-design that complies with the rule of the three Rs: reduction, reuse and recycling. The company's business model proposes a reduction in the environmental cost of processes through the optimisation and responsible use of materials and waste .

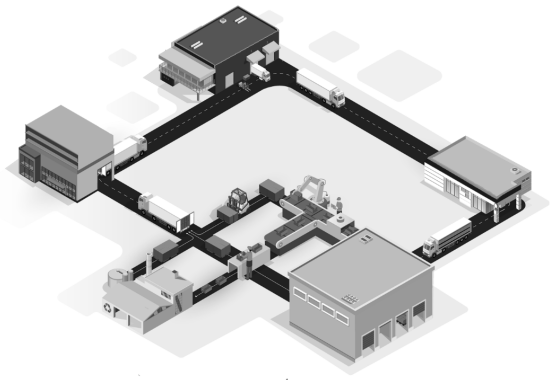


Figure 1: Life cycle of company containers

Currently the company has 13 warehouses strategically distributed in the country to facilitate the attention to their clients. In all the warehouses, there is an amount of containers/items of different types. These containers can be available to be used in a warehouse, they can stacked to be repaired in the same warehouse or they can be sent to a recycling center to make new ones. Each warehouse has a limited number of containers available for each type. However, it is possible that a large amount of containers are needed, so the company has the capacity to acquire more to supply the demand.

The main goal of the company is to transport the containers from warehouses to supermarkets. Currently, the company must optimize the planning of 2000 trips weekly. To this end, the company aims to develop an efficient search algorithm to plan the weekly allocation of a fleet of trucks in the different warehouses spread throughout the country. Due to physical and temporal constraints, the company aims to minimize the cost of transport generated by the trucks and the cost of stock in each of the warehouses. It must be taken into account that the amount of stock in the warehouses may be negative, so it means that the company must acquire them, so the cost of stock is directly associated with the price of items with negative stock.

This paper proposes a new SLAP proposal for balancing stocks taking into account that the problem can be modelled as a VRP, more specifically as an OVRP where the vehicle that transports the order does not return to its point of origin. To this end, A metaheurist search algorithm is proposed (GRASP). It searches for the first solution by applying a series of heuristics. Later, the local search is performed by applying a series of multi-target heuristics to filter the different solutions and choose the most optimized one.

Problem specification

In this section, the problem specification posed by the company is provided for a better understanding of the proposed algorithm, highlighting the most relevant aspects. The main objective is to assign each trip to a warehouse, taking into account all the characteristics of each trip and warehouse. To this end, we seek to minimize the cost of transport and the extra cost of stock. Minimizing the cost of the transport is relatively simple by obtaining the trip with the minimum cost. However, the cost of stock supposes a challenge, due to the fact that the stock of each container type must be balanced. To solve it, a multiobjective function must be determined by assigning weights to transport cost and stock cost to determine the fitness function. This function must maintains the transport cost stable while the stock in the warehouses is distributed homogeneously. Thus, the most important variables involved in the problem must be taken into account.

- **Transport cost** (TC): is the cost associated with an order. This cost tc is based on the distance between the starting point and the loading location, being proportional to the distance. This is taken into account when calculating the final cost in fitness function.
- **Extra stock cost** (SC): if an order composed of N items to be delivered is loaded on a location where the current stock is less than the required, the remaining negative stock quantity must be replaced. Therefore, the cost of the stock will be proportional to the negative stock multiplied by the cost of replacing each of the items.
- **Delay of delivery** (DD): each of the possible locations associated with an order are located at a certain distance

from the origin. Therefore, depending on the location chosen for loading the trip, a different delay will be assigned. This delay affects the amount of stock that is reduced from the loading station from the planned date to the real delivery date.

- **Single load point** (SLP): each order has a number of locations to which it can be uploaded. Each of these locations has a number of important decision features. An order can only be loaded at one location, with the constraint that all items needed in the order must be loaded at the same location. Thus, it is not allowed to load in different locations.

The problem is composed of a dataset (DS) which is based on the combination of three main elements that contains all necessary information to represent a problem instance (see Equation 1). Thus, each problem instance is composed of a set of orders (O) each of them representing a trip to transport items, a set of available warehouses (W) to load the items, and a set of prices (P) of each item.

$$DS = \{O, W, P\} \tag{1}$$

The set $O = \{o_1, o_2, \ldots, o_N\}$ contains all information about the orders, where N is the number of orders. Each order $o_i \in O$ is composed by three parameters $o_i = [D_{o_i}, Ir_{o_i}, W_{o_i}]$:

- D_{o_i} is the delivery date of o_i.
- Ir_{o_i} contains the set of items to be loaded on the order (see Equation 2). Each element $ir_{o_ij} \in Ir_{o_i}$ represents the amount of items to be loaded.

$$Ir_{o_i} = \{ir_{o_i1}, ir_{o_i2} \ldots, ir_{o_iP}\} \tag{2}$$

- W_{o_i} is a set of possible warehouses where order o_i can load all required items (see Equation 3).

$$W_{o_i} = \{W_{o_i1}, W_{o_i1}, \ldots W_{o_iQ}\} \tag{3}$$

  Each element W_{o_ij} is formed by three values:

  - ava_{o_ij} represents the availability of platform j to load all items of order o_i. Thus, $ava_{o_ij} = 1$ means the warehouse is available, and $ava_{o_ij} = 0$ otherwise.
  - pr_{o_ij} contains the transport cost of order o_i from platform j. If $ava_{o_ij} = 0$ this value is null.
  - dl_{o_ij} is the delay to load the order o_i from platform j. If $ava_{o_ij} = 0$ this value is null.

To control the amount of stock at the beginning of each week, the company provides a tridimentional matrix with the current status of each warehouse W (see Equation 4).

$$W = \{w_1, w_2, w_3, \ldots w_P\} \tag{4}$$

where each $wst_i \in W$ is composed of a bidirectional matrix (item x date of week). Due to the model of the company, it is allowed to generate negative stock in the warehouses associated to an order $wst_{ijk} \in \mathbb{N}$. This is allowed because each of the logistic centers (warehouse) can replace the lack of containers by shipping them.

Finally, P is the set of unitary price for each items/article (see Equation 5).

$$P = \{p_1, p_2, p_3, \ldots, p_i, \ldots p_P\} \tag{5}$$

Solving techniques

This section aims to explain the current algorithm working on the company, pointing out the structure and specifications of the implemented greedy algorithm, as well as the characteristics that make it up. Next, the proposed GRASP algorithm, local search and several improvements are detailed.

Greedy algorithm

Currently, the objective of company is the planning of weekly orders based on the demand requested by customers taking into account the available stock expected for each day of the week. The company has developed its own algorithm to obtain an optimized solution based on a greedy technique with a post-processing phase. The stock available in each article model in a week is obtained from a weekly load plan. This stock is obtained from the sum of the remaining containers from the previous week, the containers that have been successfully restored and those obtained by shipments, among others. The greedy algorithm has additional constraints posed from the perspective of the customer, warehouses and geographical areas, in addition to other factors:

- A certain number of clients only allow loading in a certain warehouse due to distance and comfort factors.
- The zone of influence or geographical area has a higher priority in the assignment of orders by each warehouse.
- The loading of an order is restricted to the type of items it contains, so there are certain warehouse that do not serve a particular item.
- The warehouses or loading centres have independent delays for each product at different times of the year, as the quantity of products (fruit, vegetables, fishes) is not constant throughout the year.

These constraints make the generated search tree too large to be addressed in one go. Therefore, the current algorithm developed by the company is split into several layers, which are put together in a post-processing step.

Firstly, the algorithm performs an ordering according to priorities. It uses a heuristic called closeness centrality applied to network theory. This measure allows us to obtain a measure of centrality of a transport network by adding the length of the paths between the different nodes, obtaining the central nodes as those that are closer to the other nodes. Later, the orders are classified according to the article with greater portion of the order, assigning all the available stocks of the nearest warehouses by this article model.

In the second layer, there are unassigned orders to a warehouse because they have failed to meet any of the restrictions of the first layer. These remaining orders are reordered with the centrality closeness heuristics to assign them to less central warehouses with larger stock. The orders that have not been assigned can be renegotiated with the customers to improve the response of the warehouse system or make up for the lack of stock by manufacturing containers needed to meet the customers' needs. However, after this second layer, there may be unassigned orders which are assigned to the most optimal warehouse in a post processing. These

are marked, so that an expert human must review them, looking for combinations that improve the cost of global transport without increasing the extra cost of stock to manufacture containers. The constraints that are joined in the greedy algorithm by layers only take into account the cost that generates the transport and the requests of the clients, forcing an expert to review all the trips to be able to obtain a better solution for the company.

GRASP

Taking into account the structure of the above greedy method and its drawbacks to obtain an good solution, a GRASP algorithm is developed to obtain a balanced solution in terms of travel cost and stock balance. To do this, we focus on obtaining an efficient solution and keeping the stock as balanced as possible among the warehouses.

Algorithm 1 GRASP

```
1: input: All orders O, All stock matrix W, Item price vector P, Size of LR list n
2: output: Optimized solution s*
3: i ⟵ 0
4: t ⟵ [1, 2, 3, ..., |O|]
5: while empty(t) do:
6:    if sort-criterion(i) then:
7:       swap( o_{i-1} , o_i )
8:       t ⟵ t \ i
9:       i ⟵ i + 1
10:   end if
11: end while
12:
13: s ⟵ [ ]
14: LR ⟵ [o_1, ..., o_n]
15: while |s| ≠ |LR| do:
16:   j ⟵ 0
17:   lcr ⟵ [ ]
18:   while j < |LR| do:
19:      lcr ⟵ lcr ∪ LocalSearch(LR_j, P, α, |LR|)
20:      j ⟵ j + 1
21:   end while
22:   s ⟵ s ∪ lcr*
23:   LCR ⟵ LR \ lcr*
24:   LCR ⟵ LR ∪ o_{n+1}
25:   n ⟵ n + 1
26: end while
27: return s*
```

The proposed algorithm 1 is based on two steps. First, the list of orders is sorted to improve the response of the algorithm (from lines 5 to 11). This assumption is based on the idea of constrainedness, analyzing the most restrictive orders first in order to give the algorithm more decision power in the final iterations. And secondly, the iterative part of GRASP (from line 15 to 26) obtains a fraction of the size of the list of requests (LR). This list contains the algorithm's candidates to assign a warehouse. Then. a local search is generated in which the best warehouse is obtained. Once we have the best warehouses for all the trips, we select the warehouse with the lowest cost, eliminating the order from the LR list and adding the next one from the sorting list.

LR ordering The sorting of the LR list is carried out to give more flexibility and to test the behaviour of the algorithm with different inputs. This pre-processing auto-allocates all the trips that have only one available warehouse due to availability or customer's restrictions.

This type of ordering is modular for easier testing. For this purpose, a sorting algorithm is developed based on a criterion (see Line 6. Once all the values in the list are correctly sorted, the list with these values is returned.

- **Sort by number of warehouses (Model 1) :** the list is ordered by the number of possible warehouses, that is, the first order in the list will have less warehouses, allowing the algorithm to assign them with less number of decisions to make (constrainedness).
  - **Sort by number of warehouses and items (Model 2) :** Among the trips ordered by warehouses and with the same value, they are ordered by the least number of items required for the trip.
  - **Sort by number of warehouses and delay (Model 3) :** Among the trips ordered by warehouse, they are ordered according to the number of days of delay in that order. This is because the more days of delay there are, the more blocked stock remains during the week.
- **Sort by Delay and Items (Model 4) :** In this case, orders are classified by the sum of the delays of all their warehouses and by the amount of items required for the trip.
- **Sort by warehouses plus standardized delay and items (Model 5) :** this sorting is carried out by mixing the three criteria previously defined. To do this, we add the standardization of the delay and the number of warehouses and then order by the number of items.
- **Sort by number of weight warehouses and delay (Model 6) :** This sorting is based on the previous one by adding a beta weight to the delay. Then, the sorting is done by number of items.

After carrying out some tests to verify the viability of all sorting types, it was found that sorting by number of warehouses, more specifically, sorting by number of warehouses and items gave the best results without negatively affecting time or computer costs.

Local search

Once we have the order list sorted, we iterate it to execute a local search on each order in order to obtain the best assignation. This local search is based on obtaining the objective function for each of the possible warehouses in a trip, by generating a list of values for each warehouse. Once this list is obtained, we will obtain the minimum value of it and the select warehouse is assigned to the trip.

Algorithm 2 Local Search

1: *input:* A order o, List of item price P, Alpha value for objetive funtion α, Length of LR list n
2: *output:* Best choice of warehouse Q^*
3: $Q \longleftarrow [\ \ ]$
4: $I \longleftarrow Ir_o$
5: $M \longleftarrow W_o$
6: **For** j **in** $\{0, \ldots, |M|\}$:
7: $\quad Cs \longleftarrow 0$
8: $\quad Ct \longleftarrow$ **getTransportCost**(M_j)
9: $\quad$ **For** i **in** $\{0, \ldots, |I|\}$:
10: $\quad\quad Cs_i \longleftarrow P_i * \sum_{k=D_o}^{|wst_{ij}|} wst_{ijk} \mid wst_{ijk} < 0$
11: $\quad\quad Cs \longleftarrow Cs + Cs_i$
12: $\quad$ **end for**
13: $\quad Q_j \longleftarrow \alpha * Ct + (1 - \alpha) * Cs$
14: $\quad Q \longleftarrow Q \cup Q_j$
15: $\quad F_\alpha \longleftarrow \frac{1-\alpha}{n}$
16: $\quad$ **if** $\sum Cs < 0$ **then**:
17: $\quad\quad \alpha \longleftarrow \alpha + F_\alpha$
18: $\quad$ **else:**
19: $\quad\quad \alpha \longleftarrow \alpha - F_\alpha$
20: $\quad$ **end if**
21: **end for**
22: $Q^* \longleftarrow$ **getBest**(Q)
23: **return** Q^*

In each iteration, one of the orders in the LR list is analyzed and the objective function is calculated with all the items that compose the order and the possible warehouses to obtain the best one. The objective function is based on calculating the cost of transport and the cost of stock and weighting them with an alpha value to give more weight to the cost of transport which generates more costs (see Line 15). The transportation cost of a trip is obtained by adding the price of traveling to a warehouse of each order (see Line 8). The cost of the stock is obtained by subtracting the quantity of an article required in a warehouse in a certain day, multiplying by its price in the case that the stock is negative. If the stock remains positive the cost of the stock will be zero (see Line 10). The warehouse assignments for each trip is stored and the best assignment is selected.

Improvements

During the development of the algorithm, several issues appear that negatively affect the optimality of the solution. To overcome these issues, solutions are composed independently of the main code to avoid problems in the efficiency of the algorithm.

The first issue is observed when several GRASP iterations are performed over the same sorting mode, so the algorithm returns identical results because the sorting is so specific that there is no option for certain trips to be sorted differently, i.e. a certain trip in a sorting mode always ends up in the same position in the list. In order to solve this problem, a *shuffle* method is generated which exchanges those trips that have more similar characteristics depending on the chosen sorting mode.

The next issue is related to obtaining an optimized solution. Due to the constant update of the stock matrices during the iterations of the algorithm, it is observed that in problems in which the quantity of stock of the orders is higher than the served by the warehouses, it proves a polarization of the objective function generating an unbalance in the allocation of the warehouses. To overcome this issue, a *alpha scheduler* is generated to update the target function in each of the iterations to avoid this polarization. An alpha parameter is entered in the target function, which increases or decreases its value depending on the assignment in each of the iterations. In case the remaining stock in the selected platform for that trip is positive, alpha grows to give more importance to the transport cost. Otherwise, if the value of the platform stock is negative, the alpha parameter decreases giving more influence to the stock cost.

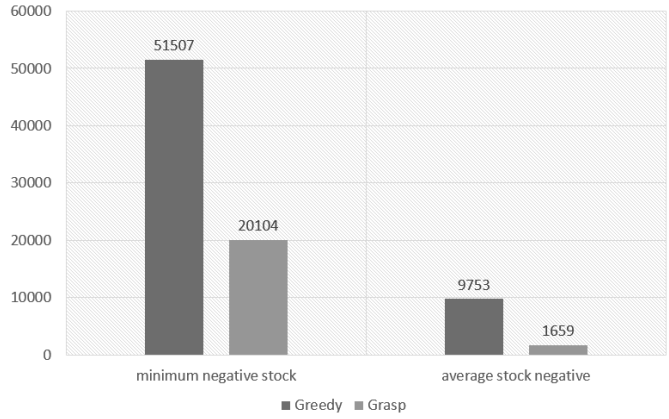
Evaluation

This section presents the results obtained from the comparison of the greedy algorithm implemented by the company and the proposed GRASP algorithm. To this end, several case studies were carried out and the results were split in both objectives: transportation cost and stock balance. The runtime is not included in the next tables due to the fact that the Greedy algorithm had a worse behaviour than the proposed GRASP algorithms in all instances. In average, the runtime for the Greedy algorithm was around 20 minutes for solving each instance, meanwhile the GRASP algorithm took around 50 seconds to solve it. All instances had a different complexity due to the fact that they represents real instances corresponding to different weeks of June and July, 2020. In average, each instance is composed on 1800 trips, each of them involving 5-10 warehouses.

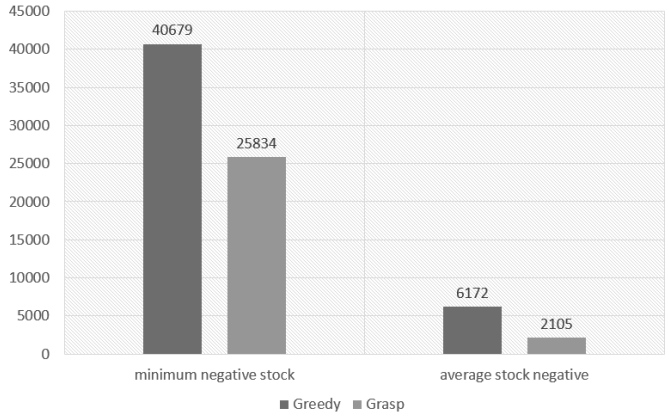
Date	Greedy	GRASP
Week 1	214	233
Week 2	217	244
Week 3	219	242
Week 4	216	235

Table 1: Transportation cost

Table 1 shows the transportation cost of 4 different instances. It can be observed that the transportation cost is lower in the greedy algorithm than the GRASP algorithm. This is due to the fact that the greedy algorithm is developed to obtain the lowest transportation cost by using the centrality heuristic. However it does not tale into account the stock balance.



(a) Week 1 extra stock cost



(b) Week 2 extra stock cost

Figure 2: Extra stock cost

To correctly evaluate the concentration of remaining stock in the warehouses, two metrics are generated. The first one obtains the most negative value in all the warehouses (see Figure 2a). This allows us to determine when the algorithm is wrongly balancing the stock and to be able to compare the stock concentration between the two algorithms. The second metric is based on obtaining the average of negative stock of all warehouses (see Figure 2b). The results obtained by this metric help us to have a more global vision of the stock balancing by the two algorithms. The results of the GRASP algorithm regarding the amount of remaining stock in the different weeks indicate that the load balancing was being effective, with up to 82% improvement in negative average stock and 60% in negative minimum stock.

Conclusions and future work

Transport companies need to optimize their logistics infrastructures and strategies to be more efficient in a more competitive word. The proposed problem tries to merge two different problems, the warehouse stock management problem and the routing problem to minimize both the negative stock and transport cost. To this end, a GRASP-based metaheuristic has been developed to improve the greedy algorithm that is currently managed by the company. The results on several case studies show that although the greedy algorithm had a better behaviour in the transport cost, due to it is specially guided by a heuristic. However the proposed GRASP algorithm overcomes the results obtained by the greedy algorithm in the stock balancing, improving the negative average stock up to 82%.

In further works, we will improve the proposed algorithm by combining the proposed GRASP algorithm with a genetic algorithm (GA). Thus, the solutions obtained by the GRASP algorithm will participate as a subset of the initial population for the genetic algorithm. It could improve the quality of the solutions due to the high capability of the GA to combine previous solutions and avoid trapping in local optima.

References

Barkaoui, M., and Gendreau, M. 2013. An adaptive evolutionary approach for real-time vehicle routing and dispatching. *Computers & Operations Research* 40(7):1766–1776.

Burkard, R.; Dell'Amico, M.; and Martello, S. 2009. *Assignment Problems*. SIAM - Society of Industrial and Applied Mathematics, 1 edition. 382 Seiten.

Cao, E.; Lai, M.; and Yang, H. 2014. Open vehicle routing problem with demand uncertainty and its robust strategies. *Expert Systems with Applications* 41(7):3569–3575.

Current, J., and Marsh, M. 1993. Multiobjective transportation network design and routing problems: Taxonomy and annotation. *European Journal of Operational Research* 65(1):4–19.

Eksioglu, B.; Vural, A. V.; and Reisman, A. 2009. The vehicle routing problem: A taxonomic review. *Computers and Industrial Engineering* 57(4):1472–1483.

Li, X.; Leung, S. C. H.; and Tian, P. 2012. A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem. *Expert Systems with Applications* 39(1):365–374.

López-Sánchez, A. D.; Hernández-Díaz, A. G.; Vigo, D.; Caballero, R.; and Molina, J. 2014. A multi-start algorithm for a balanced real-world Open Vehicle Routing Problem. *European Journal of Operational Research* 238(1):104–113.

Pillac, V.; Gendreau, M.; Guéret, C.; and Medaglia, A. L. 2013. A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225(1):1–11.

Salari, M.; Toth, P.; and Tramontani, A. 2010. An ILP improvement procedure for the Open Vehicle Routing Problem. *Computers & Operations Research* 37(12):2106–2120.

Subramanian, A.; Uchoa, E.; and Ochi, L. S. 2013. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research* 40(10):2519–2531.

Syed-abdullah, S. S.; Abdul-rahman, S.; and Mauziah, A. 2018. Solving Quadratic Assignment Problem with Fixed Assignment (QAPFA) using Branch and Bound Approach Solving Quadratic Assignment Problem with Fixed Assignment (QAPFA) using Branch and Bound Approach.