Industrial Engineering
2018; 2(2): 42-51
http://www.sciencepublishinggroup.com/j/ie
doi: 10.11648/j.ie.20180202.11

Time Window and Location Based Clustered Routing with Big and Distributed Data

Mehmet Fatih Yüce[1], Ali Gunes[1, *], Metin Zontul[2], Tuğba Altintas[3]

[1]Department of Computer Engineering, Istanbul Aydin University, Istanbul, Turkey

[2]Department of Computer Engineering, Istanbul Arel University, Istanbul, Turkey

[3]Department of Health Sciences, Uskudar University, Istanbul, Turkey

Email address:

mehmetyuce@stu.aydin.edu.tr (M. F Yüce), aligunes@aydin.edu.tr (A. Gunes), metinzontul@arel.edu.tr (M. Zontul), tugba.altintas@uskudar.edu.tr (T. Altintas)

[*]Corresponding author

To cite this article:

Mehmet Fatih Yüce, Ali Gunes, Metin Zontul, Tuğba Altintas. Time Window and Location Based Clustered Routing with Big and Distributed Data. *Industrial Engineering.* Vol. 2, No. 2, 2018, pp. 42-51. doi: 10.11648/j.ie.20180202.11

Received: September 14, 2018; **Accepted**: October 12, 2018; **Published**: November 7, 2018

Abstract: In this paper, a novel vehicle routing algorithm will be presented. Proposed method will be based on "time windows-based clustering" and "location-based clustering", applied in reversable consecutive order. The method partitions and models the solution space with machine learning technologies, resulting in a better performance for time window and geospatial clustering calculations. Routing process, on the other hand, will be built upon already present open source tools, giving it usability, applicability, manageability, and integration perspectives. The process combines "cluster+cluster+route" units with post process enhancements. Previous works on location-based clustering are proved to be successful, albeit with some disadvantages. On the other hand, routing algorithms have mostly implemented time window calculations as second-class citizens. In this method, time window is a major ingredient of the modelling process. This paper will also differs from some other combinatoric methods used in literature. A history and general description of used methods and tools will also be provided. It is shown that the algorithm can generate good results, some of which are the best values in the recorded literature so far. The method is applied on a big data platform. Horizontal scaling and distributed processing capabilities with the state-of-the-art tooling on such systems are also described.

Keywords: Two Step Clustering, Vehicle Routing, CVRP, VRPTW, Big Data

1. Introduction

Vehicle Routing Problem (VRP) or Truck Dispatching Problem is NP-hard. Its beginning is in the famous Travelling Salesman's Problem (TSP) [1, 2].

The TSP problem can be identified as indicated below.

G: Hamiltonian path or a complete graph.

V: a point or position set,

E: an edge set,

c_{ij}: cost for each edge defined in E.

are given. [4].

cij is the associated cost moving from m ∈ V to k ∈ V [3].

VRP, on the other hand, is first defined by Dantzig and can be described as below [4].

R: a set of routes.

V: a set of vehicles.

P: a vehicle visiting a subset of clients.

C: a set of constraints.

[5]. With this definition, each route, inside the solution set, will become a TSP instance [6]. VRPs can be classified according to their models [7]. Following list, although incomplete, will try to show some of the VRP classes (each, if not defined otherwise, can be taken as another problem class with capacitation and time window features):

Capacitated Vehicle Routing Problem/CVRP [8],

Asymmetric Capacitated VRP/ACVRP) [9],

Arc Routing Problem/ARP) [10],

General ARP (Generalized VRP/GVRP) [11, 12].

Vehicle Routing Problem with Time Windows/VRPTW [13],

Vehicle Routing Problem with Pickup and Delivery/VRPPD [14, 15].

The dial-a-ride problem (DARP) [16],
Two-Echelon Vehicle Routing Problem (2E-VRP) [17, 18],
Vehicle Routing Problem with Backhauls (VRPB) [19],
Green Vehicle Routing Problem GVRP [20],
Distance Constrained VRP [21],
Location Routing Problem [22, 23],
Heterogeneous Vehicle Routing Problems (HVRPs) [24],
Electric Vehicle Routing Problems (E-VRPs) [25].

1.1. Method

The algorithm in this paper is also a heuristic one [26, 27]. The main problem it tries to solve is to enable horizontal scaling on the execution nodes and simplify the calculation. So that, if the problem size gets bigger, a solution would still be found in a similar time and cost. For this to be done, a big data system is assumed to give support for the calculation. With it, calculation distribution is achieved.

As a novel method, merging spatial clustering with time windows-based clustering is introduced. With this method, it will be proved that better timing and less resource usage will be accomplished. Some of best solutions found for a well-known data set is shown at the end.

The problem this paper handles can be generalized best with a capacitated and time windowed VRP model.

1.2. Similar Work

A previously known system makes trials with time windows by Sakalli with a well know try and see model [28]. Using this method, when the route length gets bigger, the time the system takes to generate a single route is increased logarithmically. Because of this, only small route lengths could be generated.

Methods using time window-based clustering is not known. On the other hand, constraint-based time window management is well known [29-32]. In those methods, if a new route is found, time window calculations are done for it. If the cost of the resulting route is better, it is accepted, else it is thrown away. Another interesting method is the time window partitioning [33]. In this method, a time window is partitioned with equal number of parts. After partitioning, a route would start within the limits of these parts. For example, a departure location having a time window of 12:00 am and 18:00 pm is given. The partitions could be arranged as; 12:00 am – 15:00 pm and 15:00 pm – 18:00 pm. According to this model, after partitioning, the routes can start either at 12:00 am or 15:00 pm.

2. Clustering

As input data increases, vehicle routing algorithms slow down. Clustering is a way to get over this performance problem [34]. With clustering, input data are split into smaller sub sets. The resulting structure, then, is taken as multiple routing problems. Another use for clustering is the ability to parallelize the operation. Operations on smaller data will finish in less time. Because each subset can be processed independently, processing finishes even in better times.

Point based, or location-based clustering is the most used clustering method. In that, points near to a pre-selected center location are taken into the same cluster/set. After clustering, routing is done independently for each set [35].

2.1. Advantages of Clustering Methods

Even if an algorithm has an O(n) performance characteristics, when the size of the input data increases, the processing time of the algorithm also increases. When the process is a combinatorial operation, the time it takes to finish will increase with some consideration as indicated next.

If every unit of work in a combination process taken as one, then the operation can be formulated as follows formulated by;

$$C(\mathrm{n},\mathrm{r}) = (C_r^n) = \frac{n!}{r!\,(n-r)!}$$

As it will be shown next, when input data (n: number of input) increase in size, the time it takes increases with ((n+1) *(n)). Above formula is taken as the time it takes to calculate first n element. Substituting n+1 for n in the above equation resuls in the following;

$$C(\mathrm{n}+1,\mathrm{r}) = (C_r^{n+1}) = \frac{(n+1)\,n!}{r!\,((n+1)-r)!}$$

This can also be shown with discrete numbers. If n is 10 and r is 4 (routing 10 locations with each route having a length of 4 location), then the time it takes will be;

$$C(10,4) = (C_4^{10}) = \frac{10!}{4!\,(10-4)!} = \frac{3628800}{24(6)!} = \frac{3628800}{17280} = 210$$

If n is 20 and r is 4; the it becomes.

$$C(20,4) = (C_4^{20}) = \frac{20!}{4!\,(20-4)!} = \frac{2432902008176640000}{24(16)!} = \frac{2432902008176640000}{502146957312000} = 4845$$

Doubling the input data will increase the processing time 23 times. On the other hand, if clustering is done, then it would take 210+210=420 units time. After clustering, if input data increases two folds, then the time to process is also increases two folds. Doubling the input data doubles the processing time. So, a linear time complexity is achieved.

If the process is parallelized, then this increase would not happen. So, it will be done in the same time [36, 37].

2.2. Disadvantages of Clustering

Although clustering has many advantages, a disadvantage

is that it may not always generate an optimal solution. Finding the optimal solution is always a possibility in theory. But, this mostly will require a brute-force method and it can take a very long time. When time is a constraint, brute-force methods only work with small data sets. If data set size increases, these kinds of methods start taking longer and longer times. Instead, clustering will provide better times; but with the (probable) cost of getting away from the optimal solution. Clustering partitions input data, but the optimal solution usually is found when the whole data is taken into the consideration. On the other hand, if there are no additional constraints, clustering can give solutions near the optimal. With constraints coming into the picture, clustering makes it worse in terms of finding the optimal solution. Heuristic algorithms also behave in this manner. They accept getting away from the optimal solution. This problem can be overcome with post processing of routes with some methods such as result elimination and swapping.

Clustering Methods Used in This Work.

In this paper, two clustering methods are used; location based and time windows-based clustering. Location based clustering is done to decrease the time it takes to process a complete schedule. Time windows-based clustering is done over location-based clustering or vice versa. The rationale can be described as following;

Starting from a departure point, a vehicle can traverse a limited number of points. Added to this, each reachable point has its servicing time windows. So, constructing secondary sets upon the first clusters can decrease the processing time even more.

In this work, a clustering process can be described as follow;

Primary clustering (sets).

Secondary clustering (subsets).

Boundary adjustments (sets & subsets).

Solving each subset.

After processing adjustments (sets & subsets).

Accepting the solution (sets & subsets).

3. Big Data

Big data is the data that traditional data management tools cannot handle or have a hard time processing it [38]. In its structure, a “big data” can mean a very broad set of data structures; from CSVs to log files [39]. Also, to be able to handle the massive number of user bases, social media firms made quite investments to big data tools [40]. One of the most important features of big data systems is their ability to scale horizontally.

Traditional data management systems provide similar features, like sharding, as big data systems. But this usually cannot provide the same benefit if the system is not implemented through a big data system.

3.1. “Big Data” Definition

Although big data definition is known from early nineties, it is believed that John Mashey is the first to popularize it [41]. On the other hand, Dug Laney summarizes it with three features [42];

1. Data size.
2. Data Velocity.
3. Data Variety.

Apart from that, each big data system will have the following features;

1. A distributed file system.
2. Distributed processing.

A distributed file system, in image, is not different from any other file system, but, it is very different [43]. One of the preeminent properties of a distributed file system is its fault tolerance.

As for the distributed processing system, Google, in 2003 released a paper on how they do processing on their distributed processing clusters [44]. This became an inspiration on the coming systems. Another paper by Dean and Ghemawat which helped more in this matter, paved the way for Hadoop [45]. MapReduce system, in theory, is taken from these two papers and improved to be used with big data systems developed later.

3.2. Hadoop

Hadoop developers describe it as a distributed processing platform that can work and scale on commodity servers [46]. Mostly written in Java programming language, it is partly implemented with C and with some shell scripts [47]. At first, Hadoop starter, Doug Cutting with Marc Caferalla, thought that an indexing system with one billion web pages could be implemented with a half-million dollars initial investment and thirty thousand dollars operating cost for each month [48]. In 2009, Hadoop publicized its sorting of one terabytes data in sixty-two seconds [49]. The same could be done in sixty-eight seconds by Google [50]. Many companies currently use Hadoop or its derivatives in their systems [51].

3.2.1. Hive™

Hive is one of the most used software in Hadoop ecosystem. It was first developed to query Hadoop data files using an SQL query language. It was first developed by Facebook and later made open source. For a metastore, it needs a relational database. This can be a small database as Derby or a big one Microsoft SQL Server. If an SQL based structure will be used, this metastore is very important. For example, Spark, when introducing its own query language, instead of creating another metastore, it used the one provided by Hive. Nearly all the Hadoop ecosystem tools use this metastore. This provides familiarity operations in all systems.

3.2.2. Spark

Until a couple years ago, MapReduce (M/R) performance was not investigated that much and accepted as the de-facto standard in big data batch computation. Since 2009, M/R started to receive some criticism. After that, Hadoop developers started M/R version 2 and started presenting it in Hadoop. But time has proven to need much more performance gains. Microsoft, Cloudera and Hortonworks firms each started for their on

distributed processing systems. Microsoft's DryAdd [52] is such an example. Building on this, developers started working on Spark. Also, Hortonworks presented Tez and advertised it. Cloudera, on the other hand, backed Spark.

In the following years, Spark gained a lot of momentum [53]. Ease of use made it one of the fundamental tools of data scientists. Unlike other Hadoop ecosystem elements, Spark does not need Hadoop to work on. It can work locally on a single machine or distributed across multiple computers.

Spark MLlib

Previously worked under Mahout, MLlib is the spark's machine learning and data mining solution. In this paper, for example, k-means algorithm is used from this library MLlib has three main parts; Recommendation Systems / Collaborative Filtering, Clustering, Classification.

4. Problem

Two step (or n step) clustering is not a much-used method in vehicle routing. Some constraint-based operations, which can easily be implemented with clustering, is the general method in use. This is applied as it is indicated in Table 1.

Table 1. *Constraint Control Algorithm for each Route.*

Step	Process
1	During optimization, for each route to be checked, constraints are run.
2	If the route passes the constraint, then it is accepted, and its cost is calculated.
3	If the cost is better than the last best cost, it is taken, else thrown away.
4	After that, other route candidates are processed.

The problem in this algorithm is that if input data size increases, the time it takes to process all the constraint increases. To lower this processing time, two methods can be proposed;

1. Algorithm is technically previewed, and optimizations are done at the programming language level.
2. Number of locations are to be lowered so that the algorithm takes less time.

In the first proposal, optimizations will not make that much of a difference asymptotically. On the other hand, lowering number of points available will always lower processing time. The second method is the main objective of this paper.

Let us think in terms of the second method which Table 2 outlines an algorithm for;

Table 2. *Performance İmprovement with Clustering.*

Step	Process
1	First, a single clustering is applied.
2	Clustering method is mostly location / geographical based.
3	Locations that are near to each other are taken to the same cluster.
4	Algorithm step applied in the previous table is executed on each cluster.

Very big systems (like GittiGidiyor.com) are processing millions of requests in a day. A healthy horizontal scaling can also be achieved with this method. Because, parts of data can be executed on different processing sets.

5. Methods Used

5.1. Method

Dataset used in this paper is thought as a four-dimensional matrix. Each dimension is as follows;

1. x,
2. y,
3. start-ti,
4. finish-ti.

"x" and "y" locations/points.

"start-ti" and "finish-ti" are time intervals in each end of the location.

K-Means is used as the clustering algorithm. It can be found in *org.apache.spark.mllib.clustering.KMeans* class.

5.2. MLlib K-Means

K-Means is an unsupervised clustering algorithm [54]. It is one of the most commonly used clustering algorithms.

In MLlib, it a parallel variant of "k-means++" algorithm called "k-means||" [55]. Parameters are;

i. k: Number of clusters.
ii. maxIterations: number of loops the be executed.
iii. initializationMode: random or "k-means||" way.
iv. initializationSteps: How many starting steps in "k-means||".
v. epsilon: convergence value.
vi. seed: Random seed.

5.2.1. K-Means Workings

$X= \{x_1, \ldots, x_n\}$ is a d-dimensional Euclidian space. $\boldsymbol{k}$ is the number of clusters and it is a whole number.

$$||x_i-x_j|.|$$

Is the Euclidian distance between x_i and x_j. Distance between an x point and $Y \subseteq X$ subset

$$\text{smallest}_{y \in Y} ||x - y||.$$

For the subset $Y \subseteq X$, center of mass is;

$$centerofmass(Y) = \frac{1}{Y}\sum_{y \in Y} y$$

Steps [55];

Random *k* central points are taken

Each loop is based on these points.

Center of mass is changed with each point added to the cluster

The system starts converging after some tries.

5.2.2. K-means++

Instead of selecting random starting points, "K-means++" proposes another method [56]. The centers are selected depending on the previous selected location.

5.2.3. K-Means||

K-means|| proposes a stochastic center selection method. A problem indicated is that the number of clusters can be more than the required. After processing, it tries to distribute excess clusters among others. Another feature of this algorithm is that it can work in parallel with multiple machines at the same time.

5.3. Realization

In this paper, Scala is used as the programming language. IntellyJ IDEA Community Edition is chosen as an IDE.

5.3.1. Data Set

Gehring & Homberger data set is used to benchmark the algorithm [57]. The data set is modelled after the famous Solomon Data Set [29].

Modelling Each Primary Cluster

For the actual clustering to take place, a modelling phase must be executed so that each location can be set to a primary cluster value. In this initial clustering, the number of clusters is four and the number of iterations is 5000.

```
val numberOfCenters = 4
val numberOfIterations = 5000
```

Fort the secondary clustering runs, modelling also need to run for each cluster to be generated. If the parameters are the same (which this paper did), the number of models would be sixteen.

5.3.2. Time Windows

A simple input excerpt for modelling is indicated below;

```
c1_10_1

VEHICLE
NUMBER     CAPACITY
 250         200

CUSTOMER
CUST NO.  XCOORD.    YCOORD.    DEMAND   READY TIME  DUE DATE   SERVICE TIME

   0    250       250        0        0       1824        0
   1    387       297       10      200        270       90
   2      5       297       10      955       1017       90
   3    355       177       20      194        245       90
```

In this file, there are 100 jobs to be fulfilled. The first three columns are the client and their locations. The next two columns are the time intervals that the job must be fulfilled in.

For a single row;

1 387 297 10 200 270 90

For a client at (387,387) coordinates

And having a number 1 as client Id

And having an order of 10 unit

Will leave the central depot around 200

At worst, will arrive at the client around 270.

After arriving, handling the order is 90 units time.

5.3.3. Modelling

Each row must be converted to a format so that the algorithm will understand and execute it. Lines like below

```
250      250        0      1824
387      297      200       270
5        297      955      1017
355      177      194       245
```

Will be converted to vector format;

```
sc.textFile(file)
  .zipWithIndex()
    .filter(_._2>9)
    .map(_._1)
    .map(_.replaceAllLiterally("   "," "))
  .filter(_!=null)
  .filter(_.trim.length>0)
  .map { line =>
    line.split(separator).filter(_.trim.length>0)
      .map {
        field => field.toDouble
      }.toSeq.toArray
  }
.drop(1).take(2).union(line.drop(4).take(2))
```

After converting the data, training should start.

```
val kMeansI = new KMeans()
// Setting the parameters
kMeansI.setK(options.numberOfCenters)
kMeansI.setMaxIterations(options.numberOfIterations)
// Return the KMeansModel which is got after running the KMeans
// algorithm on the data gathered by the DataSource component
val model = kMeansI.run(training)
val toWrite = new File(options.TXT_INSTANCE_FOLDER + "models/" + file.getName +"/");
toWrite.mkdirs()
model.save(sc,toWrite.getAbsolutePath +(new Date()).getTime)
```

After this operation, the next step was to save the model. Each cluster/set will have its own model file saved in the file system.

5.3.4. Clustering

After modelling, using each saved model file for each cluster, points will be set to clusters;

```
val groups = model.predict(training)
  .zipWithIndex()
  .map {
    line =>
      (line._2, line._1)
  }
  .join(trainingIndexes)
  .map {
    line =>
      (line._2._1, line._2._2.apply(0))
  }
  .groupBy(t => t._1);
```

This operation is identical for every clustering step (primary or secondary).

5.3.5. Routing

Each cluster is saved in a separate scheduling file. These files contain points that are in the same cluster and their planning data. The format is the same as the original input data set. The only difference is that each contains single cluster locations. After each routing for each cluster is done, results are gathered together to generate a single input schedule result.

As a routing engine used by Schröder [58].

All the operation can finish in half an hour. If the number of clusters decreased, the scheduling time will increase.

```
val files = RouterUtil.getListOfFiles(options.TXT_INSTANCE_FOLDER)

for(file<-files){
  routeOne( file)
}
```

And routeOne method;

```
def routeOne(file: File)= {
 val files = RouterUtil.getListOfFiles(GHSCExporter.getLatestExportFolder(file).getAbsolutePath)
 var totalCost = 0.0
 var totalJob = 0.0
 var lstSols = new ParHashMap[VehicleRoutingProblemSolution, Boolean]
 files.par.foreach { f =>
  val solution = route(f, null)
  totalCost += solution.getCost
  totalJob += solution.getUnassignedJobs.size()
  lstSols +=(solution, true)
 }

 println("solution ")
 val routes = new util.ArrayList[VehicleRoute]()
 val uJobs = new util.ArrayList[Job]()

 val alls = lstSols.map(_._1).toSeq.toArray
 for (s <- alls) {
  val sol = s.asInstanceOf[VehicleRoutingProblemSolution]
  RouterUtil.printSol(sol)
  uJobs.addAll(sol.getUnassignedJobs)
  for (r <- sol.getRoutes.toArray()) {
   val route = r.asInstanceOf[VehicleRoute]
   routes.add(route)
  }
 }

 println("Initial Total Cost : " + totalCost)
 println("Initial Total Unassigned Jobs : " + totalJob)

 val lastSol = new VehicleRoutingProblemSolution(routes,uJobs,totalCost)
 var sF = new File(options.TXT_INSTANCE_FOLDER + "schedules/" + file.getName + "/" );
 sF.mkdirs()
 sF = new File(sF.getAbsolutePath + "/" + file.getName+ "_" + totalCost + "km_" + totalJob + "uj" + +
(new Date()).getTime + ".sc");


 RouterUtil.printSol(lastSol,sF)
```

6. Results and Discussion

Table 3 contains the best results found so far. All the execution is run on three 2.39 GHzx4, 30GB RAM Linux machines. Results indicated show better performance than those found in the literature.

Table 3. *Results.*

Data Set	KB Vehicle	KB KM	TS Vehicle	TS KM
c1_10_6	99	43830,21	102	43319,73126
c1_10_7	97	43453,92	101	42781,84419
c1_10_8	92	44092,74	102	43309,19313
r2_10_1	19	42188,86	33	39951,00347

KB: Known Best TS: This paper.

An interesting point is that, the algorithm cannot find a best solution for every input data, but for some, it has found the best results in literature. This indicates that with a little extra optimization, very good results can be found.

7. Conclusions and Further Research

Vehicle routing algorithms are one the oldest study in the book yet there are tremendous enhancements to be done and needed [59]. On paper everything seems to solve many problems. However, in a real life, there are many constraints in real life, that makes most of the theoretical work useless [60]. Either the work is too narrow and specific to be applied, or the size of the problem becomes the bottleneck. In this paper, the attack to the behemoth is from two fronts which are the most crucial; problem size and timing. The result is a generalized and generated model for large capacitated and time windowed vehicle routing problems.

The sample dataset used here is well documented and is frequently used in literature. On the other hand, the real power of this study would be apparent with much bigger sets. Homberger and H. Gehring tried to enlarge the set used in by Solomon with statistical methods, generating one thousand points and orders, as opposed to 100 or less points and orders [29, 57]. A similar approach by Uchoa, can be taken to enlarge this set even further [61]. This way, it may give a better real-world test, albeit literature does not have this kind of benchmarks.

Proposed method also introduced a way to horizontally scale routing processing with state-of-the-art distributed processing platforms and existing routing tools. Also, as a new method, two steps clustering is introduced, albeit with a narrow implementation. Apart from that, combination of clustering algorithms, data point rearrangements, or model enhancements could possibly provide better results.

References

[1] A. J. Hoffman, J. Wolfe, R. S. Garfinkel, D. S. Johnson, C. H. Papadimitriou, P. C. Gilmore and B. L. Golden, (1986). "The traveling salesman problem: a guided tour of combinatorial optimization," J. Wiley & Sons.

[2] M. M. Flood, (1955). "The Traveling-Salesman Problem," *J. Opns. Res. Soc.Amt. 4,* pp. 61-75.

[3] L. Zambito, (2006)."The traveling salesman problem: a comprehensive survey," in *Project for CSE 4080*.

[4] Dantzig, G. B. and Ramser, J. H., (1959)."The Truck Dispatching Problem," *Management Science, 6(1),* p. 80–91.

[5] S. N. Kumar and R. Panneerselvam, (2012). "A survey on the vehicle routing problem and its variants," *Intelligent Information Management, 4(3),* p. 66.

[6] D. Cattaruzza, N. Absi, D. Feillet and J. González-Feliu, (2017)."Vehicle routing problems for city logistics," *EURO Journal on Transportation and Logistics,* vol. 6, no. 1, pp. 51-79.

[7] M. Granada, (2016). "Literature Review On The Vehicle Routing Problem In The Green Transportation Context," *revista. luna. azúl, 42,* pp. 362-387.

[8] T. K. Ralphs, L. Kopman, W. R. Pulleyblank and L. E. Trotter, (2003)."On the capacitated vehicle routing problem," *Mathematical programming, 94(2-3),* pp. 343-359.

[9] D. Vigo, (1996). "A heuristic algorithm for the asymmetric capacitated vehicle routing problem," *European Journal of Operational Research, 89(1),* pp. 108-126.

[10] H. A. Eiselt, M. Gendreau and G. Laporte, (1995). "Arc routing problems, part II: The rural postman problem," *Operations research, 43(3),* pp. 399-414.

[11] R. Baldacci, E. Bartolini and G. Laporte, (2010)."Some applications of the generalized vehicle routing problem," *Journal of the Operational Research Society, 61(7),* pp. 1072-1077.

[12] I. Kara and T. Bektas, (2003)."Integer linear programming formulation of the generalized vehicle routing problem," in *EURO/INFORMS Joint International Meeting*, Istanbul.

[13] O. Bräysy and M. Gendreau, (2005)."Vehicle routing problem with time windows, Part I: Route construction and local search algorithms," *Transportation science, 39(1),* pp. 104-118.

[14] B. Çatay, (2010)."A new saving-based ant algorithm for the vehicle routing problem with simultaneous A. I. Savran, E. Musaoglu, M. F. Yuce and E. Yesil, (2014). "RouteArt: A new framework for vehicle routing problem with pickup and delivery using heuristic bubble algorithm," *Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium on,* pp. 91-96.

[15] A. I. Savran, E. Musaoglu, M. F. Yuce and E. Yesil, (2014). "RouteArt: A new framework for vehicle routing problem with pickup and delivery using heuristic bubble algorithm," *Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium on,* pp. 91-96.

[16] J. F. Cordeau and G. Laporte, (2003). "The dial-a-ride problem (DARP): Variants, modeling issues and algorithms," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1(2),* pp. 89-101.

[17] T. G. Crainic, G. Perboli, S. Mancini and R. Tadei, (2010)."Two-echelon vehicle routing problem: a satellite location analysis," *Procedia-Social and Behavioral Sciences, 2(3),* pp. 5944-5955.

[18] J. Gonzalez-Feliu, G. Perboli, R. Tadei and D. Vigo, (2008)."The two-echelon capacitated vehicle routing problem," [Online]. Available: https://halshs.archives-ouvertes.fr/halshs-00879447/.

[19] J. Crispim and J. Brandão, (2005)."Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls," *Journal of the Operational Research Society, 56(11),* pp. 1296-1302.

[20] J. Jemai, M. Zekri and K. Mellouli, (2012)."An NSGA-II algorithm for the green vehicle routing problem," in *In European Conference on Evolutionary Computation in Combinatorial Optimization*.

[21] G. Laporte, M. Desrochers and Y. Nobert, (1984)."Two exact algorithms for the distance ‐ constrained vehicle routing problem," *Networks, 14(1),* pp. 161-172.

[22] G. Nagy and S. Salhi, (2007)."Location-routing: Issues, models and methods," *European Journal of Operational Research, 177(2),* pp. 649-672.

[23] A. I. Savran, E. Musaoglu, C. Yildiz and M. F. Yuce, (2015)."An efficient solution to Location-Routing Problems via a two-phase heuristic bubble approach," *Advanced Logistics and Transport (ICALT), 2015 4th International Conference on,* pp. 169-174.

[24] Ç. Koç, T. Bektaş, O. Jabali and G. Laporte, (2016)."Thirty years of heterogeneous vehicle routing," *European Journal of Operational Research,* vol. 249, no. 1, pp. 1-21.

[25] A. M. J. E. J. O. &. L. G. Froger, (2018)."Modeling and solving the electric vehicle routing problem with nonlinear charging functions and capacitated charging stations.," *ODYSSEUS*.

[26] Z. J. Czech and P. Czarnas, (2002). "A parallel simulated annealing for the vehicle routing problem with time windows," in *Proc. 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, Canary Islands, Spain.

[27] M. F. Yuce, E. Musaoglu and A. Gunes, (2016)."Enhancing heuristic bubble algorithm with simulated annealing," *Cogent Business & Management,* vol. 3, no. 1.

[28] A. Sakalli, (2013)."Heuristic bubble algorithm for a linehaul routing problem: An extension of a vehicle routing problem with pickup and delivery," in *Computational Intelligence and Informatics (CINTI), 2013 IEEE 14th International Symposium on. IEEE*.

[29] M. M. Solomon, (1987). "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research, 35(2),* p. 254-265.

[30] J. Brandão, (2017)."Iterated Local Search Algorithm for the Vehicle Routing Problem with Backhauls and Soft Time Windows," *ISEG-Lisbon School of Economics and Management, REM, Universidade de Lisboa.,* no. 10.

[31] G. Desaulniers, F. Errico, S. Irnich and M. Schneider, (2016)."Exact algorithms for electric vehicle-routing problems with time windows," *Operations Research,* vol. 64, no. 6, pp. 1388-1405.

[32] F. Errico, G. Desaulniers, M. Gendreau, W. Rei and L. M. Rousseau, (2018)."The vehicle routing problem with hard time windows and stochastic service times," *EURO Journal on Transportation and Logistics,* vol. 7, no. 3, pp. 223-251.

[33] U. M. Yildirim and B. Çatay, (2012). "A time-based pheromone approach for the ant system.," *Optimization Letters 6.6,* pp. 1081-1099.

[34] J. E. Beasley, (1983). "Route first—cluster second methods for vehicle routing," *Omega, 11(4),* pp. 403-408.

[35] R. Dondo and Cerdá, (2007)."A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows," *European Journal of Operational Research, 176(3),* pp. 1478-1507.

[36] G. S. Almasi and A. Gottlieb, (1994).Highly parallel computing - 2nd ed., Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc..

[37] D. E. Culler, J. P. Singh and A. Gupta, (1999).Parallel computer architecture: a hardware/software approach., Gulf Professional Publishing.

[38] C. Lynch, (2008). "Big data: How do your data grow?," *Nature,* pp. 28-29.

[39] A. Holmes, (2014).Hadoop in practice, Second Edition, Manning Publications Co.

[40] Boyd, D. and Crawford, K., (2012). "Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon," in *Information, communication & society*, pp. 662-679.

[41] J. R. Mashey, (1997)." Big Data and the next wave of infraS-tress," in *Computer Science Division Seminar*.

[42] D. Laney, (2001)."3D data management: Controlling data volume, velocity and variety," *META Group Research Note, 6,* p. 70.

[43] K. Shvachko, (2010)."The hadoop distributed file system.," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE*.

[44] S. Ghemawat, H. Gobioff and S. T. Leung, (2003). "The Google file system," *ACM SIGOPS operating systems review,* vol. 37, no. 5, pp. 29-43.

[45] J. Dean and S. Ghemawat, (2004)."MapReduce: simplified data processing on large clusters," 28 October 2004. [Online]. Available: http://research.google.com/archive/mapreduce.html.

[46] Apache Yazılım Vakfı, (2018). "Welcome to Apache Hadoop!," [Online]. Available: http://hadoop.apache.org/. [Accessed 03 10 2018].

[47] T. White, (2015). "Hadoop: The Definitive Guide 4th Edition" , pp. 3-15.

[48] M. Cafarella and D. Cutting, (2004). "Building Nutch: Open Source," *Magazine Queue - Search Engines Volume 2 Issue 2,* p. 54.

[49] O. O'Malley, (2008)."Terabyte sort on apache hadoop". [Online]. Available: http://sortbenchmark. org/Yahoo-Hadoop.

[50] "Sorting 1PB with MapReduce," (2008). [Online]. Available: https://googleblog.blogspot.com.tr/2008/11/sorting-1pb-with-mapreduce.html. [Accessed 3 10 2018].

[51] "Distributions and Commercial Support,"(2018). [Online]. Available: https://wiki.apache.org/hadoop/Distributions%20and%20Commercial%20Support. [Accessed 3 10 2018].

[52] J. Li, S. Mehrotra and W. Zhu, (2013). "Prajna: Cloud Service and Interactive Big Data Analytics,". [Online].

[53] "Cloudera Customers," (2018). Available: http://www.cloudera.com/customers.html. [Accessed 3 10 2018].

[54] S. P. Apache Yazılım Vakfı, (2018)."Clustering - RDD-based API," [Online]. Available: http://spark.apache.org/docs/latest/mllib-clustering.html#k-means. [Accessed 3 10 2018].

[55] B. Bahmani, B. Moseley, A. Vattani, R. Kumar and S. Vassilvitskii, (2012)."Scalable k-means++," *Proceedings of the VLDB Endowment, 5(7),* pp. 622-633.

[56] D. Arthur and S. Vassilvitskii, (2007)."k-means++: The advantages of careful seeding," *In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics.,* pp. 1027-1035.

[57] J. Homberger and H. Gehring, (2018). "Extended Solomon's VRPTW instances.," 26 09 2018. [Online]. Available: https://www.sintef.no/globalassets/project/top/vrptw/homberger/1000/homberger_1000_customer_instances.zip.

[58] S. Schröder, (2018). "jsprit-project," [Online]. Available: http://jsprit.github.io/. [Accessed 03 10 2018].

[59] J. Schuijbroek, R. C. Hampshire and W. J. Van Hoeve, (2017). "Inventory rebalancing and vehicle routing in bike sharing systems," *European Journal of Operational Research,* vol. 257, no. 3, pp. 992-1004.

[60] B. T., D. E. and L. G., (2016). "Green Vehicle Routing," *Green transportation logistics Springer, Cham.,* pp. 243-265.

[61] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal and A. Subramanian, (2017). " New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research,* vol. 257, no. 3, pp. 845-858.