GPS2GR:Optimized Urban Green Routes based on GPS Trajectories

Antonio M R Almeida[*]
Federal University of Ceara
Brazil
manoel.ribeiro@lsbd.ufc.br

Jose L A Leite[†]
Federal University of Ceara
Brazil
luis.alves@lsbd.ufc.br

Jose A F Macedo[‡]
Federal University of Ceara
Brazil
jose.macedo@dc.ufc.br

Javam C Machado[§]
Federal University of Ceara
Brazil
javam.machado@dc.ufc.br

ABSTRACT

This work proposes an automated method to encourage the formation of convoys of vehicles on the main highways of a city through the global integration of traffic lights. To do this, a work planner is used to generate a green wave movement in the most frequent flows of the city. The discovery of hot routes uses the principle of minimum conservation of the flow traffic of by a window time of allowing different behaviors throughout the day. Street segment estimation is discovered by mapping the GPS coordinates (a process known as map matching) to its right place segment and using the low sampling approach, which allows for greater precision in this process. This whole process considers time windows of the trajectories that are grouped according to the density in time. We prove that the proposed model optimizes the flow of vehicles in urban centers to maximize the flow up to the saturation limit of the roads, without the need of using intelligent agents or computer vision.

CCS CONCEPTS

• **Information systems** →**Spatial-temporal systems;** *Global positioning systems;*

KEYWORDS

Trajectory Pattern Mining, Hot Routes, Green Wave, Traffic-Light Scheduler

[*]PhD Candidate in Computing Science Department
[†]Scholarship of scientific initiation
[‡]Researcher in Computing Science Department
[§]Researcher in Computing Science Department

ACM Reference format:
Antonio M R Almeida, Jose L A Leite, Jose A F Macedo, and Javam C Machado. 2017. GPS2GR:Optimized Urban Green Routes based on GPS Trajectories. In *Proceedings of 8th ACM SIGSPATIAL Workshop on GeoStreaming, Los Angeles Area, CA, USA, November 7–10, 2017 (IWGS'17),* 10 pages.
DOI: 10.1145/3148160.3148167

1 INTRODUCTION

The past few years have seen a dramatic increase in the number of handheld travel systems and GPS-embedded smartphones. The proliferation of these devices has enabled the collection of a huge amount of GPS trajectories. Many applications, such as route planners, route finders and georeferenced social media, have started to use information from GPS data to achieve a better quality of service.

On the other hand, traffic congestion is one of the leading causes of productivity loss and decline of the standard of living in urban settings. The data generated by smartphones with GPS provides datasets rich in information that can be processed to provide solutions to urban traffic problems.

Our problem is how to create an optimization method for semaphore programming that is able to encourage the formation of convoys of vehicles following a green wave through hot routes. Using as input the flow variation estimated by GPS trajectories, a challenge in this is how to deal with trajectory with sampling failures, since clouds or other obstacles, such as buildings, can commonly keep GPS devices from acessing the sattelites, causing failures [11].

Those failures generate trajectories with gaps that need to be filled in to create a complete path. To solve this problem, we apply a map matching algorithm which, with the help of an A* [1] algorithm, is able to fill in the missing gaps in the trajectories and significantly increase the accuracy of the match. [1]

In practice, a huge amount of low-sampling-rate GPS trajectories is generated [11]. Unfortunately, most of the current map-matching approaches only deal with high-sampling-rate

[1]A* is a computer algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently directed path between multiple points, called nodes

GPS data – typically one point every 10-30s, and become less effective for low-sampling-rate points as the uncertainty in data increases. Small trajectory failures can be corrected with this approach. Another challenge is how to identify hot routes from a set of unsupervised paths. We propose to estimate the intensity of traffic, based on the density of trajectories of GPS, grouped by the time window. We must also consider the benefits of clustering time scale by using discretization techniques to allow specific behaviors to be captured when one divides paths in the cluster. There are techniques that allow discretizing time and group trajectories within each cluster. Some algorithms may not be able to cope with GPS sampling errors, generating incorrect and meaningless matched paths. Our proposal is to extend ST-matching and FlowScan algorithms in order for them to be able to deal with GPS sampling failures and using time scale defined a priori. Our solution extends the low-sampling method for the map-matching problem (ST-Matching) [23], the method for detection of hot routes (FlowScan) [10] and uses concepts from the technique for time discretization (CSTH) [22]. Finally, this work provides an unprecedented optimization solution for vehicle flow in convoys, reducing the original integer programming problem to a combinatorial optimization problem.

The main contributions of this paper are:

(1) A method for creating time windows for trajectories based on the analysis of their densities.
(2) For each time windows, a method to estimate the density of vehicles by counting trajectories, using map-matching variation.
(3) For each time windows, the development of method for discovering hot routes using the density estimation obtained previously.
(4) For each time windows, the generation of a new method to find the optimum configuration of traffic light times on hot routes in order to maximize the flow in the green waveform over the most frequent paths, maximizing the overall traffic flow.
(5) To solve the above problem, the reduction of the classic offline optimization model of semaphore scheduler, to a Job Shop Scheduler problem (JSSP)
(6) The application of a simulation method to prove that the solution generated by JSSP is the best

2 PRELIMINARIES

Definition 1. ***(GPS trajectory)*** *A trajectory $\mathcal{T}$ is a sequence of GPS positions, each defined by a quadruple consisting of id, timestamp, latitude and longitude. Suppose a path T, such that, $\mathcal{T}$ = (a, b, c, d). A trajectory dataset $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n\}$ is a collection of trajectories. This work uses the GPX trajectories format which is the standard GPS data exchange format*

Definition 2. ***(Road Network)*** *A road network is a directed graph $G = (V, E)$, where V is a set of vertices representing the inter subsections and terminal points of the road segments , and E is a set of edges representing road segments.(pathway)*
This work uses the Road Network in OSM format which is the XML file created in the OpenStreetMap (OSM) format, which is used for saving street map information.

Definition 3. ***(Hot Route)*** *It is used to reveal the common behavior of a set of trajectories in a given time window. The hot route in this context is the sequence of adjacent edges that share the certain amount of traffic (minTraffic) from start to end, defined as a parameter.*

Definition 4. ***(Convoy)*** *Grouping of vehicles in space-time following a common flow at an average track speed held together in a green-wave cluster.*

Definition 5. ***(Density of trajectories)*** *Number of trajectories passing in a street segment in a time window*

Definition 6. ***(Green wave)*** *It within occurs when a series of traffic lights are coordinated to allow continuous traffic flow over several intersections in one main direction on one avenue.*

Definition 7. ***(Green Route)*** *A green wave of vehicles built on a hot route that can go through several distinct avenues and streets before dissipating.*

3 RELATED WORK

3.1 Time Series Discretization Problem

The discovery of knowledge in time series generally uses the density of trajectories by time interval as a criterion of grouping. In order to enable cluster discovery, the symbolic representations of time in hours, minutes and seconds are converted into numerical scales facilitating the application of several methods of discretization or grouping, besides facilitating the evaluation of the quality of the method through the analysis of the final error of each discretization solution or grouping. The discovery of patterns in trajectories can form clusters that reduce the set of solutions of the problem and group windows of times in which the same solution remains valid.

In our context, discretization would be applied to group trajectories with homogeneous density, thus facilitating the analysis and understanding of the results, and enabling the discovery of patterns in time-space series. [6]

Many studies use histograms with automatic adjustment of the bin width resulting in the smallest error in the data representation. Given an amount of data, generating a histogram representing the probability density function (PDF), which allows extracting as much information as possible, is not an easy task. Doing it empirically may be cumbersome, especially if there is more than one experiment. Thus, it is interesting to use a previously known algorithm that calculates the histogram with an optimal number of bins. For the case of data that follows the Poisson distribution, the algorithm for bin size selection presented by [17] allows this automatic calculation. The result of this algorithm is the selection of a range of classes, but implementing this efficiently is crucial for scientific processing which uses huge amounts of data.

CSTH is a clustering based spatiotemporal histogram which uses mean integrated squared error (MISE) as a metric to optimize the adjustment of the histogram classes that represent the dataset of trajectories.

Benefits of time series discretization for trajectory data include:

- A more accurate representation of variations in density of the trajectories during the day
- A more accurate representation of the hot routes and traffic changes due to the different dynamics of city traffic throughout the day

Other methods have been proposed [17] using adjustment of the histogram bin size by minimizing the MISE. However, all histogram-based methods use fixed domain width clusters, which is not very suitable for trajectory densities that vary frequently over the day. Therefore, we sought a method where the width of clusters varies according to the density of trajectories.

3.2 Map-matching Problem

Local or incremental methods try to find the local matching of [8] geometries. The incremental methods use two measures of similarity to evaluate the candidate edges, one for distance similarity and another for the similarity of orientation. The combined similarity measure is calculated as the sum of the individual scores. The complexity of time is $O(n)$ since we find adjacent edges for each sample, where n is the number of GPS points to be matched. The "adaptive cutout" method uses the Dijkstra algorithm to construct the shortest path in the local free space graph. It runs in $O(mn \log m)$, where m and n are, respectively, the number of edges in the road network and the number of GPS points.

In addition to incremental matching, [23] proposes a segment-based matching method to assign trust values to different sampling points. It first matches high-trust segments and then combines low-trust segments using previously merged borders. In general, when a new position is matched, a local or incremental method considers only a small portion of the trajectory close to the position. It runs fast and works well when the sampling frequency is too high. However, as the sampling rate decreases, the problem of "arc-skipping" becomes prominent, causing degradation of significant accuracy. In contrast, with a reasonable increase in the complexity of time, our tailored ST-Matching algorithm is more robust for decreasing the sampling rate.

3.3 Hot Routes Discovery Problem

The problem of discovering frequently followed (hot) routes has also been examined in a recent work [10], but only for the case where objects are confined to move in a known network. A hot route in that context is a sequence of edges, not necessarily adjacent, that share a given amount of traffic.

In this work, we address the problem of discovering hot routes in a road network. Informally, a hot route is a general traffic flow pattern. The set of hot routes offers direct insight into a city's traffic patterns. Urban planning officers can use them to improve traffic flow; store owners and advertisers can use them to place their properties at the best locations; police officers can use them to maximize patrol coverage. [20]

An important feature of real-world traffic is its amount of complexity. Instead of natural groups, vehicles travel at different speeds and times, even when they are on the same route. For example, in a residential neighborhood, many people leave for work in the morning and commute to work using roughly the same routes. However, it is very unlikely that a group will leave at the same time and also travel together all the way back, many events (for example, semaphore) can be easily separated.

3.4 Green Wave Problem

Mobility is one of the most significant concerns in large cities. In urban traffic, vehicles are controlled by traffic lights to give priority to a road because traffic networks in large cities often exceed their capacity. The paper [3] studied the optimization of urban traffic using an Automaton Cellular traffic model. It clarified the effect of the signal control strategy on vehicle traffic. In addition, it has shown that traffic controlled by traffic lights can be reduced to a simpler single-lane problem. Other studies [15] [12] investigated the flow of urban traffic controlled by traffic lights in a single route using the ideal speed model. They differ, however, in relation to the road capacity and the mode of transition of traffic lights, but both studied the traffic of vehicles through a sequence of traffic lights on a highway and presented a dynamic model in which a vehicle is prevented from passing other vehicles.

We consider the flow of vehicles passing through a finite series of traffic lights. Each vehicle moves with the average velocity v if the path is not blocked by other vehicles and there are no non-synchronized traffic lights. We consider vehicular traffic in the one-dimensional network. It encourages the formation of vehicle convoys through the orchestration of traffic lights where vehicles tend to travel together, all at the average speed of the v route. In the synchronized strategy, all traffic lights change simultaneously from red (green) to green (red) with a fixed time period $(1 - S_p)$ R_s (S_pT_s) . The period of green is S_pT_s and the period of red is $(1 - S_p)$ T_s. The time T_s is called cycle time and the fraction S_p represents the division that indicates the ratio of green time to cycle time. In the green-wave strategy, the traffic light changes with a certain time delay t offset between the traffic light phases of two successive inter subsections. The delay time is called the offset time. The change of traffic lights propagates backwards like a green wave.

4 OUR APPROACH

Our GPS2GR - GPS to Green Route - approach provides a stream optimization solution in Hot Routes that are revealed based on GPS trips in time window created according to the variation of path density throughout the day. The solution create an algorithm (TW) that selects the best method of discretization of timestamp considering scale reduction, histogram or grouping to use k-means, create an algorithm

(FDI) to estimate road trajectory density using GPS trajectory sampling by time window and make an extension of the FlowScan algorithm to calculate the hot routes by time window, using an output of FDI. Finally, optimize the convoys of vehicles on hot routes, reducing it to a problem of Job-shop scheduler.

Summary of the proposed solution:

- TW - It creates an algorithm that uses k-means to compute the time discretization of the trajectories with a minor amount of time window.
- FDI- It creates an algorithm that uses the ST-Matching algorithm to estimate a traffic density by road segment.
- HRT- It extends the FlowScan algorithm to generate frequent routes by time window.
- GR - By using a solver for JSSP, It finds an optimal solution of scheduler traffic lights that allows all virtual convoys to follow a green wave without collision. In order to that it uses two other algorithms:
  - MC - It creates virtual trains of vehicles that will follow in green wave by the Hot Route.
  - DTL - It finds on the map all the traffic lights connected to the Ho Routes related and that need to have the optimized programming

  The figure 1 shows the framework data flow.

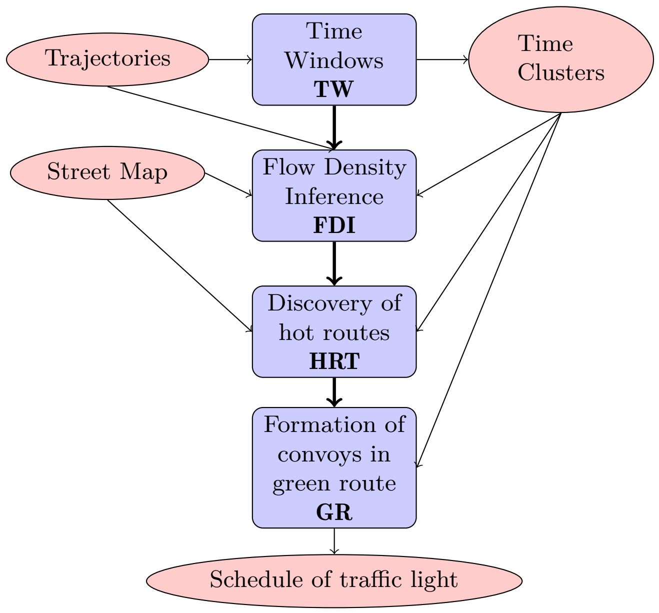
Trajectories
Time Windows TW
Time Clusters
Street Map
Flow Density Inference FDI
Discovery of hot routes HRT
Formation of convoys in green route GR
Schedule of traffic light

Figure 1: Framework GPS2GR

4.1 TW - Time Window

Traditionally the histogram is used to create time windows to group timestamp variables in order to facilitate the identification of patterns or rules. [17] Our approach alternately proposes the use of grouping to group the trajectories using the timestamp variable, comparing the error with the traditional methods of scaling and histogram, thus, obtaining time windows that will gyrate all other methods employed in this solution. We observed through exercises that the grouping technique using k-means generates grouping with less error than the histogram technique, since the grouping can generate time window with different widths and the density of the trajectories tends to vary throughout the day. The number of windows dividing the day into ideal time intervals becomes the key decision for time discretization. To run this test, we use the sum of the squared error (SSE) and define a stop criterion as a variation less than 10% in relation to the previous error.

$$SSE = {}^{n}_{i=1} X_i - \bar{X}^2. \quad (1)$$

In our approach, X_i is the timestamp of the trajectory point and $\bar{X}$ is the centroid of the generated class. We start from a dataset of vehicle trajectories from Fortaleza (Brazil) and directly apply the TW algorithm. The output is the discretization of the set of trajectories depending on the density throughout the day. It searches for the number of clusters that best fits the dataset. The adopted stopping criterion is when SSE increases or its variation is less than residual flotation K%, or SSE between SSE by the time truncation method and SSE by the histogram method. The algorithm TW considers that the distribution of trajectories throughout the day can be similar to a normal distribution and apply the techniques of histogram, Downscaling and k-means to observe which one has the lowest SSE. In the case of k-mens, a search before and after the Sturges value is used to find the best fit. In line 9 of TW, the stopping condition is defined when the error increased in relation to the last grouping or the error variation was less than 10 percent The preliminary experiments showed that in the discretization process a residual fluctuation (K) of the SSE in relation to the previous, when it falls below or equal to 10%, achieves a good classification performance, getting close to the minimum value, but with a number of small classes. Therefore, we adopted $K = 10\%$ in the final experiments. The clusters obtained through this method are wide and highly variable; some with a minutes window and others with a several hours window, varying according to the density of the trajectories dataset used.

4.2 FDI - Flow Density Inference by time windows

Traditionally the estimation of the density of vehicles on the streets is made by physically counting in them place or through the integration with sensors of traffic lights. We propose an alternative method of estimating the density, based on data from vehicle trajectories from GPS and Smartphones. In the following subsection, we apply the algorithm FDI to estimate the flow density by street and cluster. This is done to save the count where each path has passed throughout the day. As output, we get a street map with annotation by time window traffic density (clusters). In map-matching,

ALGORITHM 1: TW- Time Window

```
Input: set of trajectory points $T$
Output: set of time windows better adjusted for
        trajectories
1  $sturges \leftarrow Log_2$(size($T.nodes.count$))+1;
2  $priorSSE \leftarrow Float.MAXVALUE$;
3  $min \leftarrow sturges * 25\%$;
4  $max \leftarrow sturges * 120\%$;
5  Sort($T.nodes.timestamp$);
   /* Find the best cluster number for the dataset
      */
6  for $k$ in $min..max$ do
7      $clusters \leftarrow k - means(T.nodes.timestamp, k)$;
8      $sse \leftarrow SSE(clusters)$;
9      if $sse > priorSSE$ OR $sse > 90\% * priorSSE$ then
10         $bestClusters \leftarrow k - 1$;
11         break;
12     end
13     $priorSSE \leftarrow sse$;
14 end
   /* Defining the boundaries of each cluster    */
15 $clusters \leftarrow$k-means(T.nodes.timestamp, bestClusters);
16 $windows \leftarrow \{0\}$;
   /* Start 00:00Hs                               */
17 $k \leftarrow clusters.First$;
18 for $c$ in $clusters$ do
19     if $c <> k$ then
20         clusters.add(T.nodes[c].timestamp);
21         $k \leftarrow c$
22     end
23 end
24 $windows.add(1)$;
   /* Stop 24:00Hs                                */
25 return $windows[]$
```

each trajectory point is a position that must be mapped to a corresponding point in the road network. Considering GPS precision errors, the nearest point is not necessarily the correct one. Therefore, the whole trajectory must be considered to avoid potentially wrong matching. Thus, instead of mapping each point to its nearest node on the digital map (a method known as point-by-point matching), a set of possible paths is kept and the most likely path is chosen based on probability. This is called multiple hypothesis techniques (MTH) and it is used in some map-matching strategies [16], [14]. Initially, each trajectory point must be mapped to its set of candidate points, which are its projections on nearby road segments. This involves finding the n vertices closer to that point trajectory on the road network, as the next points are possible matches and should be taken into account. Then, we find the candidate points for each trajectory point by mapping them to their line segment projection (calculated as the distance between a point of a road segment, shown on subsection II) on any edges leaving or entering nearest neighbors. After the candidate points have been retrieved, we are faced with multiple path possibilities for the solution. From any other candidate point $c_{i,j}$ (representing the j-th candidate point on timestamp i), it is possible to move to any candidate point $c_{i+1,k}$. To determine the most likely path traveled by a trajectory, we must add probabilities to our model. To best represent this model, a candidate graph with all possible paths taken by a trajectory is used. In this graph, each candidate point is a vertex and each pair of timestamp-consecutive candidate points forms an edge. Each vertex has a score associated with it, related to the probability of a trajectory being actually on that candidate point. Likewise, each edge has a score associated with it, related to the probability of a trajectory going from its source vertex to its target vertex. These probabilities are calculated according to the ST-Matching algorithm, but the original algorithm ignores the temporal dimension. Shortest path distances between nearby vertices in the road network need to be computed as part of the transition probability calculation. Because the result of this phase is fixed for the same road network and because it can be rather costly depending on its dimension, this step is processed beforehand and stored in a distributed file system, from where it can be accessiby read afterward. After the candidate graph is built, it is possible to traverse it in order to find the optimal path (the one with the highest combined score). This step is essentially a breadth-first search starting in the vertices with the minimum timestamp.

ALGORITHM 2: FDI - Flow Density Inference

```
Input: road network $G$, set of trajectories $T$
Input: Time Windows $TW$
Output: $G^{'}$ set of maps with traffic annotation
1  $G^{'} \leftarrow \{\}$
2  for $w$ in $TW$ do
3      for $t$ in $T$ do
4          if $t.timestamp \subset w$ then
5              $G^{'}_t \leftarrow$STMatching(G, t); /* call the
                   STMatching algorithm                 */
6              for $s$ in $G^{'}_t$ do
7                  $s.density \leftarrow s.density + 1$; /* add one
                       to density on segment of Hot
                       Route                            */
8              end
9          end
10     end
11 end
12 return $G^{'}$
```

4.3 HRT - Hot route by Time windows

Then, HRT applies the algorithm (FlowScan) to identifying the hot routes using the flow density obtained from the previous step. As output, we get a map of higher density routes, one for each time window (clusters). We use this subset of lanes as input to the next step where we will apply algorithms for the generation of vehicles convoy and the optimization of traffic lights to encourage the green wave in convoys and avoiding conflicts at inter subsections. The HRT algorithm uses the neighborhood concept defined as the edges next to Eps steps, typically $Eps = 3$.

$$NEpsr = \{s \in E | ForwardNumHopsr, s \leq Eps\}, Eps \geq 0. \tag{2}$$

The HRT algorithm starts by identifying the street segments that fit into the Hot Route Start concept, ie, the street segment x where the density of vehicles passing through the $trafficr$ segment is greater than the sum of the density of the antecedent segments x $trafficx$, where this difference is greater than a minimal amount of traffic, $minTraffic$.

$$|trafficr - trafficx| \geq minTraffic, x \in \{NEpsr\} \tag{3}$$

The Hot Route is built from Hot Route Start and walks along the edges that maintain the condition of directly traffic density-reachable, see equation 4, when this condition is no longer satisfied for any of the possible routes the Hot Route is closed.

$$|trafficr \cap traffics| \geq minTraffic, s \in NEpsr \tag{4}$$

For each time window, the HRT algorithm generates a sub-map in OSM format with the routes of greater flow preservation (Hot Route). These files will be the input to the next phase of the problem that will optimize the traffic lights on the Hot Route in order to increase the flow on this way.

ALGORITHM 3: HRT - Time dependent Hot Route

```
Input: set of road network $G$ with traffic annotation
Input: Time Windows $TW$, $Eps$, $minTraffic$
Output: $R$ set of trajectories with annotation
1 $R \leftarrow \{\}$
2 for $w$ in $TW$ do
      /* repeat for each time window */
3     $r \leftarrow$ FlowScan(G[w], Eps, minTraffic);
      /* Call FlowScan algorithm */
4     $R.add(r)$;
5 end
6 return $R$
```

4.4 GR - Green Route by time windows

Since the problem of optimizing green waves on public roads is of NP-Hard complexity, the solutions have been adaptive methods [18] to find local solutions through trial and fee-feedback, similar to the methods of signal processing.

We propose an optimized method that reduces the green-wave problem to the Job-shop scheduler problem, allowing the use of Branch and Bound solver [7] to find an optimal solution to the original problem. We generate a set of car

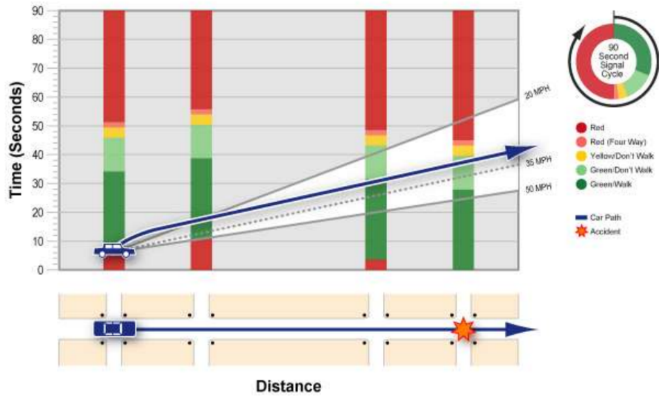

Figure 2: Example of a graph of candidate pathsSpace x time diagram for green wave

ALGORITHM 4: GR - Green Way over Hot Route

```
Input: $R$ set of trajectories with flow annotation
Output: S set of schedduler for all traffic light
1  $Processors \leftarrow \{\}$;
2  $Jobs \leftarrow \{\}$;
3  $w \leftarrow$WebsterModel(R[]);
4  $w.greentime \leftarrow w.mingreentime$;
5  do
6      $Processors \leftarrow$ DTL(R[], w);
7      $Jobs \leftarrow$ MC(R[], w);
8      $s \leftarrow$JSSP.Solver(Processors, Jobs);
9      if $s != null$ then
10         break;
11     end
12 while ($w.greentime$++ $<= w.maxgreentime$);
   /* repeats by increasing the green time, in
      search of a solution */
13 return $s$;
   /* $s$ variable contains an optimal solution */
```

ALGORITHM 5: MC-MakeConvoys

```
Input: $R$ set of trajectories with flow annotation
Input: w Webster model variables
Output: Jobs list of tasks (convoys)
1  $Convoys \leftarrow \{\}$;
2  $ConvoysSize \leftarrow w.getTotalTimeRoute$ / $w.CycleTime$;

3  while $n \neq ConvoysSize$ do
4      $c \leftarrow new\ Convoy$;
5      $c.hotRoute \leftarrow this$;
6      $c.processingTime \leftarrow w.GreenTime$;
7      $c.timeInHotRoute \leftarrow w.CycleTime$;
8      $Convoys.add(c)$;
9  end
10 return $Convoys$;
```

ALGORITHM 6: DTL-DiscoveryTrafficLights

```
Input: $R$ set of trajectories with flow annotation
Input: w Webster model variables
Output: Jobs list of processors (traffic lights)
$Processors \leftarrow \{\}$;
foreach $s \leftarrow R.Segments$ do
    foreach $n \leftarrow s.Nodes$ do
        $Found \leftarrow True$;
        if $n.TrafficSignal == True$ then
            $t \leftarrow new\ Processorn.Id$;
            foreach $p \leftarrow Processors$ do
                $Result \leftarrow$checkIfProcessorCloser(p, t);
                if $t.Id \neq p.Id \wedge Result > 30$ then
                    $Found \leftarrow False$;
                end
            end
            if $!Found$ then
                $p \leftarrow new\ Processor$;
                $Processors \leftarrow p$;
            end
        end
    end
end
```

convoy on the hot routes and look for an optimal solution for this convoy to travel on hot routes without colliding. Inspired by the work [9] we propose the formation of virtual convoys of vehicles on a hot route, optimizing the displacement of these convoys along the hot route. With this model, the sequence of block subsections traversed by a convoy is viewed as a set of machines in a job shop scheduling problem, where convoys correspond to jobs. The traversing of a block subsection by a convoy is called an operation o_i, and requires a running time p_i. [5]

4.5 Webster model

According to Webster's model for urban semaphores [19], the yellow time of an isolated semaphore is given by the equation 5. For usual situations we can consider the driver reaction time $T_r = 1s$ and an average deceleration given by $a = 2.8ms$. Therefore, the time of yellow depends on the velocity in the approaching pathway (V).

$$T_a = T_r + \frac{V}{2a} \tag{5}$$

For simplicity, the standard defined in the table 1 is adopted. The general red time t_{gr} of an approximation is a function

Approaching pathway	Yellow time
Street - 40Km/h	3s
Avenue - 60Km/h	4s
Expressway - 80Km/h	5s

Table 1: Yellow time

of the following variables:

- Velocity V of this approximation;
- Width L of the track to be crossed (of the distance to be covered by the Vehicle within approach);
- Vehicle c length;
- Clearance time T_f (minimum time elapsed between the beginning of time of the transverse track and the moment of entry of a vehicle into Speed).

Considering the usual situations, the following parameters were adopted:

- Secondary roads (streets) with a maximum width of 9 m, and speed of 40 km / h
- Preferred routes (avenues) up to 30 m wide, and speed 60 km / h;
- Length of 5 m of the vehicle;
- Clearance time $T_f = 1.2s$.

Approaching pathway	Way to be crossed	t_{gr}
Any	Street	$0s$
Street	Avenue	$2s$
Avenue	Avenue	$1s$
Expressway	Expressway or Avenue	$0.4s \rightarrow 1s$

Table 2: General red time

According to Webster's model, we must adopt sub-multiple cycles of time or programming period, to enable a change of planes without abruptty cuting in the time of green current. A T_c shorter than 30s must never be adopted and only in special cases, a T_c greater than 120s can be accepted.

$$T_c = T_g + T_y + T_r + T_{gr} \tag{6}$$

The concept of minimum green time T_{g-min} is associated with the concept of time-wasted predicted in the Webster's model, must be respected under penalty of causing major traffic jams. In table 3 shows the usual values of T_{g-min}. To define the percentage of the cycle that corresponds to

Approaching pathway	minimum green time T_{g-min}
Street - 40Km/h	12s
Avenue - 60Km/h	15s
Expressway - 80Km/h	17s

Table 3: Minimum green time

green, we adopted a proportion of traffic volume between the approach path and the secondary pathway, that is, where F_a is the approach path flow and F_b the secondary path flow, then

$$T_g = T_c - T_y - T_{gr} * T_a T_a + T_b \tag{7}$$

$$T_c = T_r + T_y + T_{gr} + T_g \tag{8}$$

The Job-Shop Scheduling (JSSP) The JSSP is an NP-hard combinatorial optimization problem and is defined as follows.

Consider a set of n tasks (jobs) and a set of m machines, where each task is composed of a set of operations whose execution order is defined. For each operation, we want to know in which machine it must be executed and the time necessary for it to be completed. There are some constraints in relation to tasks and machines.

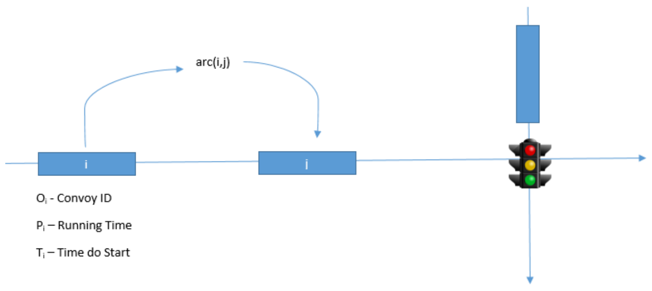

Figure 3: Job Scheduler problem for green wave

- A task cannot be assigned to the same machine twice.
- There is no order of execution in relation to different task operations.
- An operation cannot be interrupted.
- Each machine can process only one task (operation) at a time.

The purpose of JSSP is to distribute tasks between machines so that the processing time necessary to finish executing all of them (also known as makespan) is as small as possible. In other words, we want to determine the sequence of machines that minimize the makespan. In our approach, we have reduced of the problem of the green wave to a problem of Job Scheduler where the traffic lights are the processors and the convoys of vehicles are the tasks.

Mathematical model

- **Job-shop(J_m)**: In a job shop with m machines (traffic lights), each job represents a car convoy that has a particular and predetermined route to follow (hot route).
- **Processing time (p_{ij})**: The p_{ij} represents the processing time of the Job j on machine i.
- **Makespan (C_{max})**: The makespan (manufacturing time), defined as Max $(C_1, ..., C_n)$, the end time of the last job leaves the system. A minimum manufacturing time usually implies maximum use of the machine (s).

4.6 Reduction of the green wave problem to Job-shop scheduler

The process of transforming the original problem of optimization of semaphore programming to a combinatorial optimization problem (JSSP) follows the following transformations:

- Traffic lights are mapped to the processors.
- Convoys are formed to replicate the Webster cycle on the Hot Route.
- Convoys are mapped to the Jobs.
- Green time is the job run time.
- Webster model are problem constraints.
- We will use the branch and bound method to find an optimal solution that minimizes makespan.
- The solution generates a set of Jobs that can be executed without conflicts in the processors, that is, a set of convoys of vehicles that travel at the average speed of the route, passing all the open semaphores[2]
- Convoys of vehicles travel by at the average track speed for that time window.

4.7 Implementation details

The whole implementation was done in C# language because some of the baseline algorithms were originally written in this language. The implemented code is available in the GitHub repository. In Table 4 we show the inputs and outputs of each step of the process, as well as the process chaining.

Algorithm	Input	Output
TW	Trajectories(GPX)	Clusters(XML)
FDI	Trajectories,Map(OSM),Clusters	Density(OSM)
HRT	Density,Clusters	Routes(GPX)
GR	Routes,Clusters	Scheduller (XML)

Table 4: Process chaining

5 EVALUATION

5.1 Input data

For testing purposes, we used real trajectory data extracted from Taxi Simples, a taxi company based in Fortaleza, Brazil. This dataset had a very high-frequency sampling rate but was filtered down to reflect one with a low sampling character. Maps in OSM XML format were obtained from OpenStreetMap and then parsed into a road network-like graph data structure using an XML parser written in C# [23]. The dataset used has collected data paths in the period between 11/14/2015 and 09/05/2016, containing 22,791,003 GPS points.

5.2 Experiments Setup

For the experiments, we used a Intel (R) Core (TM) i7-4770S CPU @ 3.10GHz with 16GB RAM memory, 256KB Cache L1, 1.0 MB Cache L2 and 8.0 MB Cache L3. The trajectory data of the Taxi [2] was downloaded and unzipped and we randomly chose the day 6/5/2016 to perform the experiments of this article, which contains alone 749,049 GPS trajectory points. The format of the files was converted from CSV to GPX and the city map of Fortaleza was downloaded from the website Open Street Map [13]. To demonstrate the effectiveness of our algorithms, we perform the sequence of

[2]Vehicles at the ends of the convoy may eventually leave the convoy and be held at a traffic light, this is the case for vehicles with a speed less than or greater than that of the convoy.

the following steps on the raw data of GPS trajectories in the city of Fortaleza. Then the framework was executed in sequence as described in figure 1 and the results were recorded and analyzed.

5.3 Experimental Results

To evaluate GPS2GR, we compared the results of three different forms of discretization of the time scale of the trajectories and measured the number of generated classes together with the error observed using the sum of squares error (SSE) criterion:

- Discretization for time scale in hours and minutes, rounding (truncating) the course of time to just hours and minutes
- Discretization using occurrence of frequency histogram with fixed-width bin
- Proposed approach - discretization using k-means

Method	N.Cluster	SSE	Status
Down. Minute	120	167,49	Above the Sturges
Down. Hour	9	325,30	
Histogram	20	481,52	Above the Sturges
k-means[5]	5	87,07	Best choice
k-means[6]	6	100,54	
Stugers	19		

Table 5: Discretization results for 600k points

We note in the Table 5 and Figure 4 that the methods are reduced to minute and the Histograma generated a number of classes superior to the limit of Stugers. Among other methods K-means [k = 6] presented the smallest error, so it was chosen. Therefore the trajectories in this dataset were divided into six independent sets for the following steps. In figure 4 the

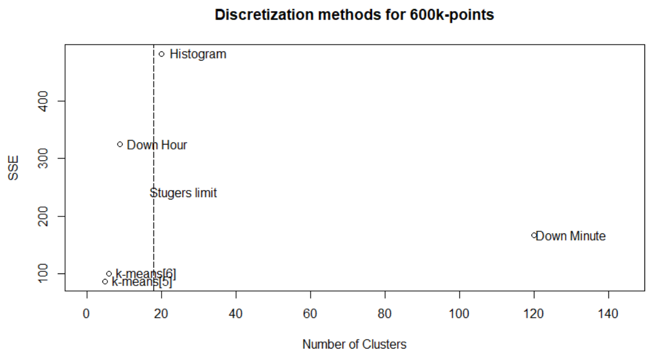

Figure 4: Analysis of number of clusters x SSE - TW

final error (SSE) of the k-means method was between the histogram method error and the time truncation method error. All the approaches used in this study showed very promising results and high applicability. Based on real-time trajectories of raw data we can obtain the city traffic behavior in each time slice where the behavior is linear. The query visualization can be used as a tool to discover the higher density flow (hot routes) and influence the implementation of public policies for urban traffic. For HRT algorithm we use the parameter minTraffic with value 3, ensuring that routes with only 3 vehicles already form a Hot Route, causing them to have more Hot Routes than less frequent routes. The results presented in table 6 show that the travel time of the GPS2GR method was between 0.3% and 0.4%, which is better than the best solution obtained by SUMO simulation, both in scenario 1 with 791 vehicles, and in scenario 2 with 6291 vehicles. Other scenarios were simulated with equivalent results.

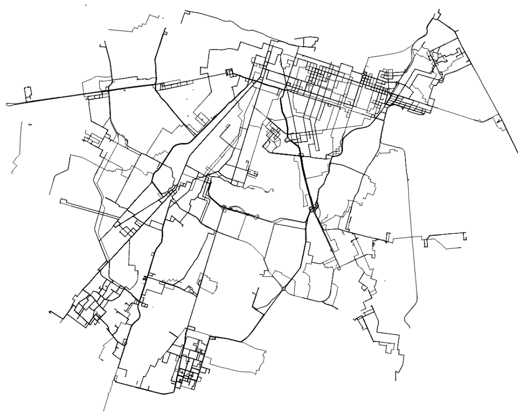

Figure 5: Result of FDI for time window[4]

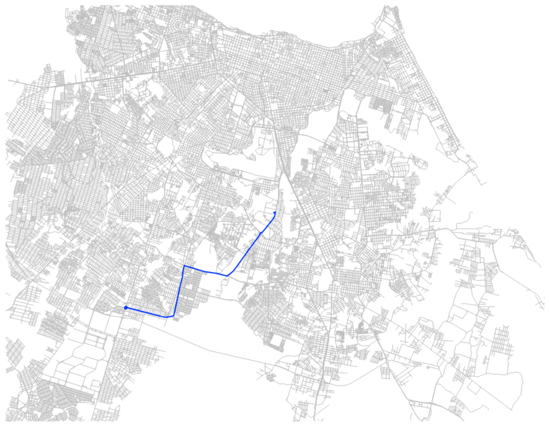

Figure 6: Hot Houte identify by HRT in time window[4]

In our experiments, we used the Simulation of Urban Mobility (SUMO) [4] software (version 0.29.0). That tool is open source and has a tremendous efficiency on large sets of data routes. Moreover, to show the effectiveness of the Green Route algorithm, three simulations were performed on a Intel(R) Core(TM) i7-4770S CPU @ 3.10GHz.

Simulation	First scenario			Second scenario		
	SUMO	GR	%	SUMO	GR	%
loaded vehicles #	791	791		6291	6291	
rot houtes #	1	1		3	3	
total lane length (km)	84,94	84,94		653,26	653,26	
Average time						
Trip[3] waiting time (s)	11,6	11,32	-2,41	28,49	28,11	-1,33
Trip duration (s)	110,95	110,53	-0,38	247,5	246,74	-0,31
Trip time loss (s)	34,2	33,8	-1,17	77,71	77	-0,91

Table 6: Simulation results

Listing 1: Part of scheduler file from GR

```
<?xml version="1.0" encoding="UTF-8"?>
<scheduler>
  <traffic-light id="0">
    <green start="00:00:00" end="00:00:17" />
    <green start="00:00:34" end="00:00:51" />
    <green start="00:01:08" end="00:01:25" />
    <green start="00:01:42" end="00:01:59" />
     ...
    <green start="23:59:03" end="23:59:00" />
  </traffic-light>
</scheduler>
```

6 CONCLUSION AND FUTURE WORK

We were able to complete an implementation of the GPS2GR solution methods and conduct experiments at all stages of the process. The temporal discretization by clustering methods used in the TW algorithm provides benefits to spatiotemporal queries. TW reduces the scale of temporal variables and facilitates visualization while allowing for the identification of patterns in sets of trajectories.

The experiments have shown that the FDI algorithm has high accuracy and scalability and can be used for trajectory data streams, quickly processing the matching to online applications that need mapping trajectories on demand in real time, we consider the FDI to be a good method to estimate the traffic density per street segment using GPS trajectories. The HRT algorithm generated long and significant Hot Routes from the point of view of taking a flow of vehicles from one point to another of the city.

Finally, the GR algorithm converged and generated optimized solutions of the opening and closing times of the traffic lights along the hot routes [21]. We prove by simulation that the solution obtained by the GR algorithm is optimal since its optimization solution is always better than the good solutions generated by heuristic methods. Optimization on hot route can solve classic vehicle containment problems at the end of a green wave, as it takes into account flow optimization, that is, green wave overflow, no matter its path.

As future work, we can consider the application of GR algorithm on real time information streaming in order to reflect real-time aspects of the flow such as accidents, road maintenance, etc.

REFERENCES

[1] Antonio MR Almeida, Maria IV Lima, Jose AF Macedo, and Javam C Machado. 2016. DMM: A Distributed Map-matching algorithm using the MapReduce Paradigm. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 1706–1711.

[2] Mirla Braga. 2016. Taxi Dataset Fortaleza, Ceará. https://s3.amazonaws.com/txsimples_tracking/index.html. (2016).

[3] Elmar Brockfeld, Robert Barlovic, Andreas Schadschneider, and Michael Schreckenberg. 2001. Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review E* 64, 5 (2001), 056132.

[4] German Aerospace Center. 2017. Simulation of Urban MObility. http://sumo.dlr.de/index.html. (2017).

[5] Francesco Corman, Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. 2009. Evaluation of green wave policy in real-time railway traffic management. *Transportation Research Part C: Emerging Technologies* 17, 6 (2009), 607–616.

[6] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. 2007. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 330–339.

[7] Google. 2017. Google Optimization Tools. https://developers.google.com/optimization/. (2017).

[8] Joshua S Greenfeld. 2002. Matching GPS observations to locations on a digital map. In *Transportation Research Board 81st Annual Meeting*.

[9] Andrew Higgins, Erhan Kozan, and Luis Ferreira. 1996. Optimal scheduling of trains on a single line track. *Transportation research part B: Methodological* 30, 2 (1996), 147–161.

[10] Xiaolei Li, Jiawei Han, Jae-Gil Lee, and Hector Gonzalez. 2007. Traffic density-based discovery of hot routes in road networks. In *International Symposium on Spatial and Temporal Databases*. Springer, 441–459.

[11] Tomio Miwa, Daisuke Kiuchi, Toshiyuki Yamamoto, and Takayuki Morikawa. 2012. Development of map matching algorithm for low frequency probe data. *Transportation Research Part C: Emerging Technologies* 22 (2012), 132–145.

[12] Takashi Nagatani. 2007. Vehicular traffic through a sequence of green-wave lights. *Physica A: Statistical Mechanics and its Applications* 380 (2007), 503–511.

[13] OpenStreetMap. 2016. OpenStreetMap. http://www.openstreetmap.com/. (2016).

[14] Jong-Sun Pyo, Dong-Ho Shin, and Tae-Kyung Sung. 2001. Development of a map matching method using the multiple hypothesis technique. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*. IEEE, 23–27.

[15] Masashi Sasaki and Takashi Nagatani. 2003. Transition and saturation of traffic flow controlled by traffic lights. *Physica A: Statistical Mechanics and its Applications* 325, 3 (2003), 531–546.

[16] Nadine Schuessler and Kay W Axhausen. 2009. Map-matching of GPS traces on high-resolution navigation networks using the Multiple Hypothesis Technique (MHT). *Arbeitsberichte Verkehrsund Raumplanung* 568 (2009).

[17] Hideaki Shimazaki and Shigeru Shinomoto. 2007. A method for selecting the bin size of a time histogram. *Neural computation* 19, 6 (2007), 1503–1527.

[18] Andreas Warberg, Jesper Larsen, and Rene Munk Jørgensen. 2008. *Green wave traffic optimization-a survey*. Technical Report. Informatics and Mathematical Modelling.

[19] FV Webster and BM Cobbe. 1966. Traffic signals. Road Research Technical Paper No. 56. *HMSQ, London* 111 (1966).

[20] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 195–203.

[21] Josh Jia-Ching Ying, Bo-Nian Shi, Kun-Chan Lan, and Vincent S Tseng. 2014. Spatial-temporal Mining for Urban Map-Matching. In *UrbComp The 3rd international workshop on Urban Computing*.

[22] Qing Zhang and Xuemin Lin. 2004. Clustering moving objects for spatio-temporal selectivity estimation. In *Proceedings of the 15th Australasian database conference-Volume 27*. Australian Computer Society, Inc., 123–130.

[23] Yu Zheng, Yin Lou, Chengyang Zhang, and Xing Xie. 2010. Map-Matching for Low-Sampling-Rate GPS Trajectories. (Feb. 25 2010). US Patent App. 12/712,857.