Car2Car framework based on DDGP3

Walter Balzano, Vinicio Barbieri, Giovanni Riccardi
Dip. Ing. Elettrica e Tecnologie dell'Informazione
Università di Napoli, Federico II
Napoli, Italy
e-mail: wbalzano@unina.it, vinicio.barbieri@gmail.com, ing.giovanni.riccardi@gmail.com

Abstract— **The purpose of this paper is to provide an algorithm for the detection of free parking stalls within a multilevel garage. Obviously, we are in the condition where the parking is very busy. Using devices of the cars On Board Units (OBU) and Road Side Units (RSU) is possible to determine, with a certain error, the matrix of distances between all sensors. The way of information exchange between cars is the VANETs. Starting from the known position of the RSU and the mutual distance between all adjacent cars OBU is possible to obtain the position of all cars applying a Distance Geometry Problem (DGP) algorithm schema. Unfortunately, the complexity of these algorithms is NP-hard. Under some conditions, the DGP algorithm can switch from continuous to discrete and it can be solved with a sort of branch and pruning algorithm. We are interested in a DDGP3 that is a Discretizable Distance Geometry Problem in R^3 variant to be used as a starting point for our work. The resolution of the algorithm is equivalent to the resolution of a problem of intersection between three spheres. This problem is non-linear and, in some conditions, it is possible to obtain an approximate solution with linear techniques.**

Keywords – DGP; DDGP; V2V; WiFi positioning; car parking.

I. INTRODUCTION

The DGP consists in seeking the coordinates of a set of points (vertices) in three-dimensional space starting from the distances between them.

Let us denote by $G = (V, E, d)$ a weighted graph, where each vertex in V content corresponds to a point in space and there is an edge between two vertices if and only if the distance between them is known. The graph G represents a DGP type problem, which in turn is the problem of finding a function

$$x : V \to R^3 \qquad (1)$$

such that for each arc belonging to E (and thus for every pair of vertices u, v connected by an arch) it is true that

$$\|x(u) - x(v)\| = d_{uv} \qquad (2)$$

In its basic form this is a "constraint satisfaction problem" the solution of which can be represented as follows:

$$X = \{x_v : v \in V\} \qquad (3)$$

II. RELATED WORKS

The different approach used in [2] and [3] for the solution of this problem is to treat it as a problem of continuous global optimization in which the set of constraints is replaced by an error function *Largest Distance Error* (LDE) that measures the difference between the calculated distance and the one known:

$$LDE(\{x_1, x_2, ..., x_n\}) = \frac{1}{m} \sum_{\{u,v\}} \frac{\left| \|x_u - x_v\| - d_{uv} \right|}{d_{uv}} \qquad (4)$$

where *m* is the number of known distances.

The DGP solution can be obtained by minimizing this function, which is not convex and contains many local minima. One of the approaches used to solve the problem is to approximate the function using a sequence of uniformly convergent functions; therefore a collection *X* is a solution if and only if the *LDE* error function is 0.

The DGP is applied for the solution of problems of location in wireless networks in which you know the distance between sensors (in our case OBU), but do not know their location, except that for some fixed named anchor (in our case RSU), then used to solve the problem.

One area in which the DGP is heavily used is Biology "Molecular Distance Geometry Problem" (MDGP).

The DGP can be treated as discrete as long as certain conditions are respected, and in particular, given a graph $G = (V, E, d)$ and a *Total Order of all vertices*, we must consider the following two axioms:

1. We assume that $(1,2,3) \subset V$, they must be a "clique" (fully reachable graph) and $\forall i \in V : i > 3$ must occur that these 3 arcs $\{(i-3, i), (i-2, i), (i-1, i)\} \subset E$
2. $\forall i \in V : i > 2$ It must apply strictly the triangle inequality $d_{i-2,i} < d_{i-2,i-1} + d_{i-1,i}$

If these conditions are verified then the cosine of the angle of torsion of each quadruple of consecutive vertices can be calculated

10.18293/DMSVLSS2017-018

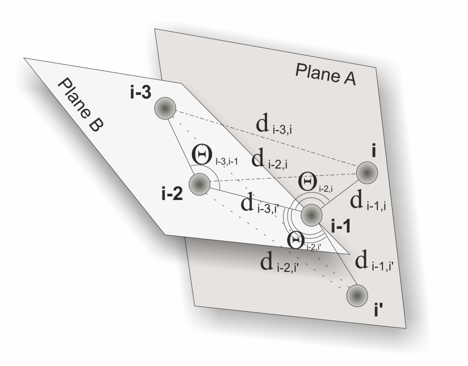

Figure 1: Angles of torsion of quadruple of consecutive vertices (through the intersection of two plans)

A position can be calculated for each one of two corners.

If these assumptions are verified the two possible positions can be calculated as the intersection of three spheres

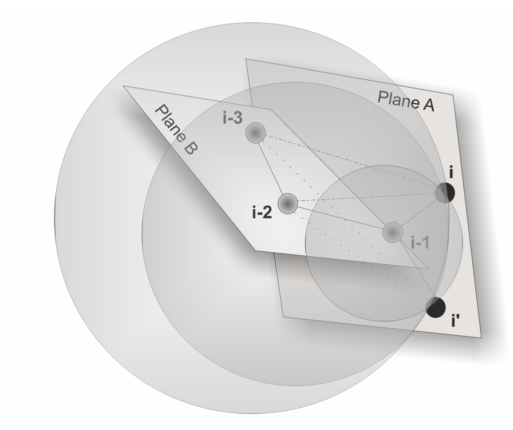

Figure 2: The intersection of the three spheres with center i-3, 1-2, i-1

- S_{-1} is the sphere with center in x_{i-1} and radius $d_{i-1,i}$
- S_{-2} is the sphere with center in x_{i-2} and radius $d_{i-2,i}$
- S_{-3} is the sphere with center in x_{i-3} and radius $d_{i-3,i}$

The intersection of the three spheres can be:

1. one point
2. two points
3. a circle
4. empty

The first hypothesis has probability 0, the third hypothesis is impossible (because of the strict triangular inequalities) and the fourth hypothesis is impossible (because the parking is very busy).

The only possibility, therefore, would be the second.

When these two assumptions are verified the LDE error function can be reduced to a discrete set and solved by the algorithm BP.

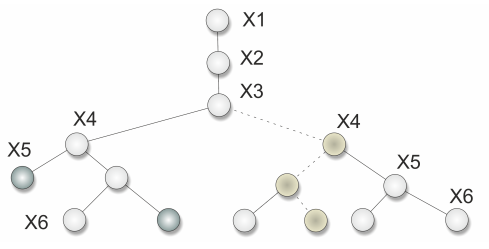

Figure 3: The algorithm BP branch and pruning

To switch from continuous to discrete domain this algorithm must be based on a real instance and these conditions must be verified:

- All distances required for the discretization (Axiom 1) are obtained from the OBU and RSU, so they are independent from the instance and stored in the VANETs;
- Distances between pairs (i, i + 1) and (i, i + 2) are know;
- Distances between pairs (i, i + 3) may be represented by intervals:
    1. $d_{i,i+3}$ is 0: it means that this is a duplicated car position, there is no branching because it can only take the same position of its previous copy;
    2. $d_{i,i+3}$ is exact: the standard discretization process is applied, and hence two possible positions for the current car position are computed;
    3. $d_{i,i+3}$ is represented by an interval: D sample distances are taken from the interval and the discretization process is applied for any chosen sample distance; 2×D car positions are generated.

III. C2C Framework

Our goal is to provide a complete procedure to find out a map highlighting the free stalls in a congested multilevel parking.

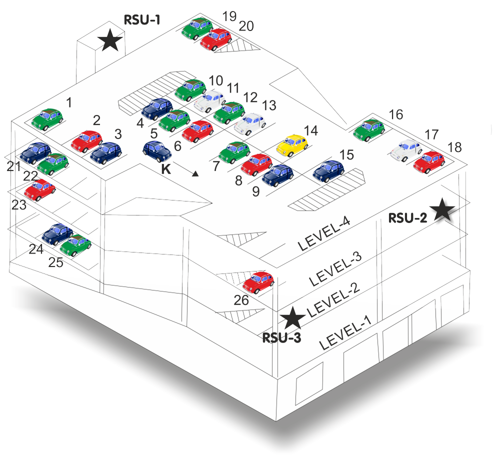

Figure 4: Parking Scheme with RSU and OBU

To achieve it we make the following assumptions:

1) *All vehicles are equipped with a sensor (OBU)*;
2) *Since each point of observation there are at least 3 fixed sensors (RSU), whose coordinates are known*;
3) *It is known the structure of the car park and are known the coordinates of all the parking stalls.*

The following flow chart shows the procedures used to obtain the map of free stalls. At each step the main input data are passed and the processing results constitute the input for the next step.

First of all, the algorithm load the map of parking and the position of Road Side Units (RSU), after that the set of vehicles registered on VANETs are loaded on memory.

All the information related to the mutual distance between the car are shared, but the On Board Units (OBU) load just the nearest.

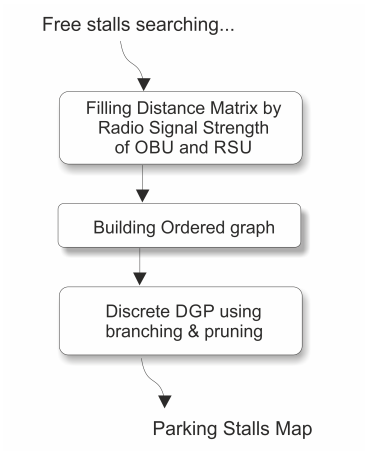
Free stalls searching...
Filling Distance Matrix by Radio Signal Strength of OBU and RSU
Building Ordered graph
Discrete DGP using branching & pruning
Parking Stalls Map

Figure 5: Workflow to search free stalls

A. *Filling Distance Matrix by Radio Signal Strength of OBU and RSU*

The model proposed in [1] allows proper accurate positioning where there are several vehicles in a small area, using a smart combination of RSS values transformed in a distance matrix provided by the V2V/V2I system.

To achieve the goal, we decide to use an external cloud[19] where to collect the distance vectors calculated by each vehicle. Such information is collected, and then these constitute the matrix of the distances which, by exploiting the cloud[20] computing capacity allows obtaining with a DGP algorithm map of the parking lot of vacancies.

Our proposal is not to use a cloud, but only assume a memory buffer on board each RSU able to accommodate distance carriers sent from the vehicles within a certain distance. Of course, knowing the structure of the parking lot and the location of the MSW is easy sizing the required memory.

Contrary to what was proposed in [1] we haven't the entire matrix of distances but only a part of it in each RSU.

When a vehicle enters the car park, in addition to calculating its distance vector, it requires RSU neighbours of distance carriers known to them and the map of the park, including the RSU coordinates and those of each parking stall.

The vehicle asks regularly update the data until it finds a parking stall and stops the vehicle. The map of the parking, being static information, once is transferred, as well as vectors of distances that do not change between a request and the next.

In this way, each vehicle has, at a certain instant, the partial matrix of the distances, the coordinates of the MSW and the parking map.

All this information will be processed by a DDGP algorithm to obtain the map of the parking lot with a list of free and occupied stalls.

B. *Building Ordered graph*

The Distance Geometry Problem (DGP) consists in finding the coordinates of a given set of points $\{x_1, x_2, ..., x_n\}$ in a three-dimensional space when some of the distances between pairs of such points are known. Our DGP instance is the set of vectors of distances received from neighboring RSU.

A vector of distances is considered significant if it contains more than three distances, sensors having less than three neighboring sensor could be initially removed from the network, and the localization problem may be solved for a sub-network (however, if a sensor has at least three sensors next it is likely that parking in that area is not congested and you do not need an algorithm to find a free place).

The graph to which we refer is G = (V, E, d), a weighted undirected graph associated to an instance of the DGP.

V = set of vertices, where each vertex in V corresponds to an c_i, in our case position of the vehicles and the RSU

E = set of arcs between vertices, there is an edge between two vertices only if it is known the distance between them (the weight associated to the edge)

d = set of distances between two vertices

We want to solve our problem by using a variant DDGP3 proposed in [3], this algorithm, depending on orders vertex and edge density.

To apply the DDGP3 algorithm it is necessary that the following condition is verified.

- (The three anchor sensors) {1, 2, 3} included in V are a "clique" (graph fully accessible), and for each parking stall occupied x_i belonging to V with rank i > 3, there are 3 vertices j, k , h such that:
    - $j < i,\ k < i, h < i,$
    - $(j, i), (k, i), (h, i) \in E$
    - $d_{jh} < d_{jk} + d_{kh}$.

To check the validity of this condition [3] suggests the following sorting algorithm:

High-level algorithm: Reordering Graph Vertices

```
Input:   $V_u$: Unordered vertices
Output:  $V_o$: Ordered vertices
 1: while( a valid ordering is not found ) do
 2:     find a 3-clique C in G($V_u$, E, d)
 3:     place the vertices of C at beginning of new
          order: G($V_o$, E, d) = C;
 4:     while($V_u$ - $V_o \neq \emptyset$) do
 5:       find the vertex $v$ in $V_u$ - $V_o$ with the largest number
                    of adjacent vertices in $V_o$;
 6:       if (l < 3) then
 7:           break the while loop: there are no possible
            ordering for this choice of C;
 8:     end if
 9:     $V_o = V_o + \{v\}$;
10:   end while
13: end while
14: return $V_o$
```

If the sorting algorithm found out a solution then we can move to the next step. We verified that the algorithm can find solutions if parking is congested (there are few free stalls).

C. Discrete DGP using Branching & pruning

Only after the sort, if the conditions are met, you can apply the DDGP3 algorithm. However, in order to avoid considering equivalent solutions that can be obtained from a given solution by translations or rotations, the first three points can be fixed. So that the final binary tree has 2n-3 positions. The first three positions are those of the RSUs closest to the viewer. At this point, we proceed to the calculation of the position and to the examination of the solution.

High-level algorithm: B&P

```
Input:   k, n, d.
Output:  Position of all vertices.
 1: for (i=1, 2) do
 2:     compute the $i^{th}$ position for the vertex k: $x_k^{(i)}$ ;
 3:     check the feasibility of the position $x_k^{(i)}$ :
 4:     if (the position $x_k^{(i)}$ is feasible) then
 5:         if (k=n) then one solution is found;
 7:         else
 8:           It calls itself with these parameters (k+1, n, d)
 9:         end if
10:     else  the current branch is pruned
12:     end if
13: end for
14: return  Position of all vertices
```

The key points of the algorithm are two: the calculation of the intersection between the three spheres and check the feasibility of the 2 solutions found.

For the calculation of intersection points we are considering whether to use the algorithm proposed by [2] MD-jeep, or the more general technique proposed by [4]. Both approaches provide solution within a reasonable elaboration time.

What really makes a difference to the convergence of branch and prune algorithm is the feasibility check.

The idea is to exploit the condition 3), in fact, knowing the coordinates of the "center" of each stall, we can say that a point (intersection of three spheres) is acceptable if its coordinates fall in turn within a of spheres of radius R (R-value to be defined) which has the center coordinates (a priori known) of the stalls.

Assuming C = $\{(X_{c1}, Y_{c1}, Z_{c1}) .. (X_{cn}, Y_{cn}, Z_{cn})\}$ the set of coordinates of all “stall’s center”, the check to apply to each (X_x, Y_x, Z_x) coordinate is:

$d((X_x, Y_x, Z_x), (X_{ci}, Y_{ci}, Z_{ci})) < R$ for a (X_{ci}, Y_{ci}, Z_{ci}) in C

where:
d (A,B): distance from the points A and B
R: radius of the sphere to be fixed (i.e. 0.5 meters)

We chose the sphere only for simplicity of calculation, you can also think of a box (more realistic).

In addition we suggest eliminating the solutions that certainly do not make sense, such as those having the Z coordinate unacceptable. Only the points for which the z coordinate (height) is compatible with the heights of the various parking levels are acceptable.

Assuming H = $\{h_1 .. h_n\}$ the set of heights of the parking floors, the check to apply to each Z_x coordinate is:

$(h_i + h_{min}) < Z_x < (h_i + h_{max})$ for a h_i in H

where:
h_{min} : minimum height of a vehicle (i.e. 0.2 meters)
h_{max} : maximum height of a vehicle (i.e. 2 meters)

Knowing the map and the coordinate of each individual stall, the algorithm replaces the calculated value of coordinates with the nearest known one.

In this way, the result is much more precise and the application is more suitable.

IV. Experimental Results

In order to facilitate the implementation of the simulation procedure, used to test the algorithm, we have used a mathematical model [21; 22] under the following simplifying hypotheses:

1. We are considering uniform configurations of parking stalls on similar floors in order to simplify the simulation. Multi-level car park with 4 floors with 250 stalls per floor and.
2. In the real case the sensors are at a distance from the floor of the parking variable from a few centimeters up to a little more than one meter. In the simulations we assume to have all sensors exactly at the level of the membership plan.
3. A further simplification we did say we have the sensors in the center of each occupied stall.
4. The distance measurement even if in reality will be affected by the error simulations we considered accurate.

We have implemented the procedure in java and performed tests on a notebook with i5 processor and 8 GB of RAM and Ubuntu 17.04 operating system.

The problem consists in verifying if a stall is free or is occupied by a sensor (vehicle).

The procedure takes as input a matrix of distances which we assume it has been acquired by the sensors, also knowing the map and the coordinates of each single stall we have the possibility of replacing the calculated value of the stall coordinates with the known coordinates of the nearest stall (within a radius R) to improve the approximation of the calculated position up to reduce to zero the error

Below is a table summarizing the results obtained in four test. As will be noted, in the table we do not report the error columns because, at each step of problem resolution of the intersection of three spheres (on which is based the whole algorithm) we obtain the coordinates of two points and choose the one feasible with respect to the parking structure.

The algorithm identifies a point feasible substitute its coordinates with coordinates known a priori.

The tests are all performed in cases of high congestion of the parking lot, and as you can see the convergence of the times are quite stable and low, in fact, are between 0.11 and 0.19 seconds. These times are encouraging for us and make us think of a future implementation on mobile devices.

TABLE I. RESULTS OF CAR2CAR ALGORITHM

Tests	**Boundary Conditions**		
	Number of stalls	***Number of OBU per floor (occupied stalls)***	***Duration***
80% occupied stalls evenly on all floors	1000	200 - 200 - 200 - 200	0.11 sec
80% occupied stalls with higher density on the lower floors	1000	244 - 222 - 195 - 139	0.12 sec
90% occupied stalls evenly on all floors	1000	225 - 225 - 225 - 225	0.15 sec
90% stalls occupied with higher density on the lower floors	1000	249 - 242 - 226 - 183	0.19 sec

V. CONCLUSIONS AND FUTURE WORKS

We started from a real problem, the search for a free place within a multi-level parking lot congested. Taking advantage of the sensors of the RSUs fixed and mobile units OBUs we calculated the distances between them, resulting in a matrix of the partial distances that we used as DGP instance. Placing particular conditions we solved the problem by using a DDGP3 algorithm. Finally we used a feasibility check that allowed us to have a rapid convergence of the algorithm.

The network model is a wireless network and each node of type OBU is able to communicate directly with any other node of same type and with one of RSU type.

The elaborations are made directly on the OBU in-car, and then the results are putted in a shared area memory through the VANETs.

In the future, we will try to improve both the calculation of the distances and the DGP algorithm. Our goal is to make the entire procedure usable with the sensors and the ability of calculation of smart phones.

REFERENCES

[1] Walter Balzano, Fabio Vitale. "DiG-Park: a smart parking availability searching method using V2V/V2I and DGP-class problem." 31st International Conference on Advanced Information Networking and Applications Workshops 2017 - DOI 10.1109/WAINA.2017.104

[2] Mucherino, Antonio, Leo Liberti, and Carlile Lavor. "MD-jeep: an implementation of a branch and prune algorithm for distance geometry problems." International Congress on Mathematical Software. Springer Berlin Heidelberg, 2010.

[3] Mucherino, Antonio, Carlile Lavor, and Leo Liberti. "The discretizable distance geometry problem." Optimization Letters (2012): 1-16

[4] Coope, I. D. "Reliable computation of the points of intersection of n spheres in n." ANZIAM Journal 42 (2000): 461-477.

[5] Balzano, Walter, Maria Rosaria Del Sorbo, and Silvia Stranieri. "A logic framework for c2c network management." Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on. IEEE, 2016.

[6] Lavor, Carlile, et al. "Discretization orders for distance geometry problems." Optimization Letters 6.4 (2012): 783-796.

[7] Balzano, Walter, et al. "A Logic-based Clustering Approach for Cooperative Traffic Control Systems." International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. Springer International Publishing, 2016.

[8] Balzano, Walter, Aniello Murano, and Fabio Vitale. "V2V-EN–Vehicle-2-Vehicle Elastic Network." Procedia Computer Science 98 (2016): 497-502.

[9] Lavor, Carlile, et al. "On a discretizable subclass of instances of the molecular distance geometry problem." Proceedings of the 2009 ACM symposium on Applied Computing. ACM, 2009.

[10] Abdelhamid, Sherin, Hossam S. Hassanein, and Glen Takahara. "Vehicle as a mobile sensor." Procedia Computer Science 34 (2014): 286-295.

[11] Sładkowski, Aleksander, and Wiesław Pamuła, eds. Intelligent Transportation Systems–Problems and Perspectives. Vol. 32. Springer, 2015.

[12] Balzano, Walter, Maria Rosaria Del Sorbo, and Domenico Del Prete. "SoCar: a Social car2car framework to refine routes information based on road events and GPS." Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on. IEEE, 2015.

[13] Y. Allouche, M. Segal, “Cluster-based beaconing process for VANET”, Vehicular Communications Volume 2, Issue 2, April 2015, Pages 80–94.

[14] Allouche, Yair, and Michael Segal. "Cluster-based beaconing process for VANET." Vehicular Communications 2.2 (2015): 80-94.

[15] Huang, Chi-Fu, Yuan-Feng Chan, and Ren-Hung Hwang. "A Comprehensive Real-Time Traffic Map for Geographic Routing in VANETs." Applied Sciences 7.2 (2017): 129.

[16] Sanguesa, Julio A., et al. "RTAD: A real-time adaptive dissemination system for VANETs." Computer Communications 60 (2015): 53-70.

[17] Milojevic, Milos, and Veselin Rakocevic. "Distributed road traffic congestion quantification using cooperative VANETs." Ad Hoc Networking Workshop (MED-HOC-NET), 2014 13th Annual Mediterranean. IEEE, 2014.

[18] Monteil, Julien, et al. "Distributed and centralized approaches for cooperative road traffic dynamics." Procedia-Social and Behavioral Sciences 48 (2012): 3198-3208.

[19] Amato, F., Moscato, F. Exploiting Cloud and Workflow Patterns for the Analysis of Composite Cloud Services (2017) Future Generation Computer Systems, 67, pp. 255-265. DOI: 10.1016/j.future.2016.06.035

[20] Amato, F., Moscato, F. Pattern-based orchestration and automatic verification of composite cloud services (2016) Computers and Electrical Engineering, 56, pp. 842-853. DOI: 10.1016/j.compeleceng.2016.08.006

[21] Amato, F., Moscato, F. Model transformations of MapReduce Design Patterns for automatic development and verification (2016) Journal of Parallel and Distributed Computing. DOI: 10.1016/j.jpdc.2016.12.017

[22] Amato, F., Moscato, F. A model driven approach to data privacy verification in e-health systems (2015) Transactions on Data Privacy, 8 (3), pp. 273-296.

[23] Oka, Hiroaki, and Hiroaki Higaki. "Multihop data message transmission with inter-vehicle communication and store-carry-forward in sparse vehicle Ad-hoc networks (VANET)." New Technologies, Mobility and Security, 2008. NTMS'08.. IEEE, 2008.

[24] Smith, David J. Reliability, maintainability and risk: Practical methods for engineers including reliability centred maintenance and safety-related systems. Elsevier, 2011.

[25] Li, Wenfeng, et al. "On reliability requirement for BSM broadcast for safety applications in DSRC system." Intelligent Vehicles Symposium Proceedings, 2014 IEEE. IEEE, 2014.

[26] Monteil, Julien, et al. "Distributed and centralized approaches for cooperative road traffic dynamics." Procedia-Social and Behavioral Sciences 48 (2012): 3198-3208.

[27] Sanguesa, Julio A., et al. "RTAD: A real-time adaptive dissemination system for VANETs." Computer Communications 60 (2015): 53-70.