A Road Intersection Control in Urban Intelligent Transportation Systems

Julien Rouyer, Alain Ninet, Hacène Fouchal, Amor Keziou

A Road Intersection Control in Urban Intelligent Transportation Systems

Julien Rouyer, Alain Ninet, Hacène Fouchal and Amor Keziou
Université de Reims Champagne-Ardenne, France
{julien.rouyer, alain.ninet, hacene.fouchal, amor.keziou}@univ-reims.fr

***Abstract*—The implementation of optimized management process for road intersections in Cooperative Intelligent Transport Systems (C-ITS) is highly recommended in order to reduce traffic jams and fuel consumption. Most of solutions are implemented on light controllers and they use various sensors in order to optimize intersection management. The use of vehicle communications, mainly I2V (Infrastructure to Vehicle), enhances the solutions since the vehicles continuously provide messages about their status (position, timestamp, speed, heading, etc.). In this paper, we suppose that a manager is embedded on a hot-spot located at a road intersection. It will know the status of each road (thanks to requests sent by vehicles approaching the intersection) and will give the right to the most relevant vehicle to pass through the intersection (many vehicles could drive in the same time when they use concurrent roads). Our algorithm measures the weight of each lane and allows to non-crossing ones having the highest weight to traverse the intersection. The calculation of the weight is based on many parameters: the waiting time, the length of the road queue and some others. Our algorithm considers that the intersection topology as stable. The algorithm has been implemented on a simulator on three urban scenarios and our results show very promising performances, they show considerable enhancement in terms of computing time.**

***Index Terms*—Road intersection, I2V, optimisation, C-ITS, Road Traffic.**

I. Introduction

In order to be ready to accept autonomous vehicles in cities, the road infrastructure should be ready to work properly. One of the most challenging issue with autonomous vehicles is intersection crossing which is constrained by the concurrency of roads on the intersection together with waiting time of vehicles which should be handled with fairness. We model in this paper the intersection manager as a supervisor which is able to watch all roads on the intersection.

We suppose that each vehicle which intends to pass through the intersection will send a request to the manager. This request contains both the entry point and the output point of the intersection. In addition to that, it should contain some features of the vehicle as its type, its dimensions. All requests are handled by the manager and its duty is to minimize the average waiting time together with the maximal waiting time. In parallel, the manager should take care of the fluidity of the traffic. All vehicles have to respect the manager decision and have to follow its recommendations in order to reach their destinations. We propose a methodology which is able to consider these requests in an optimal manner: it will minimize the waiting time of vehicles through relevant synchronised movements of vehicles.

The rest of paper is organized as follows: section II gives an overview of some recent works about the management of intersection using C-ITS. In section III, the system model that we have chosen. Section IV is dedicated to our detailed algorithm. Section V gives the results of our evaluation. Section VI gives some ideas about future work and improvement in our model. Finally, section VII concludes the paper.

II. Related work

In this section, we give details of some works done about the smart management of intersections.

[2] proposes an approach for controlling the traffic at isolated intersections where vehicles are equipped with on-board units (ITS station). A vehicle is allowed to cross the intersection if the green color is displayed to the an on-board screen. The control aims to smooth the traffic through the sequence of vehicles authorized to traverse the intersection. The main challenge raised with the assumption is that the sequence must be dynamically formed by a real time application. They have proposed a model based on Timed Petri Nets with Multipliers (TPNM) which allows to propose the control policy through the structural analysis. The resulting switching rules are very simplistic and efficient for isolated intersections.

In [3], the authors have proposed a novel approach to traffic control at intersections. Via vehicle to vehicle or vehicle to infrastructure communications, vehicles can compete for the privilege of passing the intersection, i.e., traffic is controlled via coordination among vehicles. They have modeled the problem as a new variant of the classic mutual exclusion problem. They evaluate the performance and their results show that the approach is efficient and outperforms a reference algorithm based on optimal traffic light scheduling.

[4] proposes a new algorithm to realize intersection control via vehicular ad hoc networking. As in [3], the approach adopts a mutual exclusion algorithm, which can let vehicles at an intersection compete for the privilege of passing via message exchange. In this work, they adopt a group based privilege competition which allows only group head handling requests from other lanes.

The solution proposed in [5], denoted as cooperative intersection control (CIC), considers vehicle dynamics and is based on the concept of virtual platooning. Virtual platooning allows to form platoons of vehicles that are in different lanes of the intersection and have different directional intentions. Safe passage of the vehicles through the intersection and a high intersection throughput (due to close "virtual" vehicle following) can be applied. [7] proposes general solutions to manage autonomous driving in urban areas by using a collective perception algorithm embedded on all vehicles. This collective perception enhances the performances of the road management by vehicles.

In [8], the authors consider vehicle heterogeneity and integrate a priority scheme into perimeter control. This is achieved by installing priority lanes at some of the perimeter intersections. Contrary to other works that provide priority to certain traffic modes, they dynamically identify the groups of vehicles that should be prioritized. Then, they develop a predictive control model approach able to optimize the toll for using the priority lanes and the traffic signal timings at the perimeter intersections.

[9] introduces an Internet of Agents (IoA) framework for connected vehicles where agents make their own decisions to improve the effectiveness of any C-ITS through V2V communications with other agents. A case study on distributed traffic control system without traffic signal is presented. In particular, they consider traffic control at intersection problem as a group mutual exclusion problem where only connected vehicles in non-conflict relationship are able to enter the core of intersection simultaneously. They extend the Ricart–Agrawala based-logical clock algorithm to deal with this problem and they prove the efficiency of proposal via simulations.

[10] presents a coordination method for intersection management in a connected vehicle environment. The road network is divided into three logical sections, namely, buffer area, core area and free driving area. In addition, a buffer-assignment mechanism is developed to cooperatively assign a specific crossing span for an vehicle and guide each vehicle to adjust its entry time and corresponding speed in the core area.

In all these studies, we noticed that there is a lack of considering all lanes and their constraints in order to schedule the intersection passing. For this reason, we propose a model which is managed by a central manager and which is able to handle all requests coming from vehicles approaching the intersection. Our proposal is generic enough and able to handle usual intersection as well as roundabout intersections.

III. System Model

Our model is based on the work of S. Bai and X. Bai in [1]. In this paper, the authors consider any intersection as a chord model: inputs and outputs are represented on a circle and each input is connected to all possible outputs by a dashed line. Each output is connected as well to all inputs which could link to the output. Then, each couple of points (input, output) represents a possible trajectory through the intersection. Each trajectory is represented by a segment connecting two points on the circle. Two crossing trajectories are conflict trajectories, two non crossing ones are concurrent. Fig. 3 is an example of a chord model.

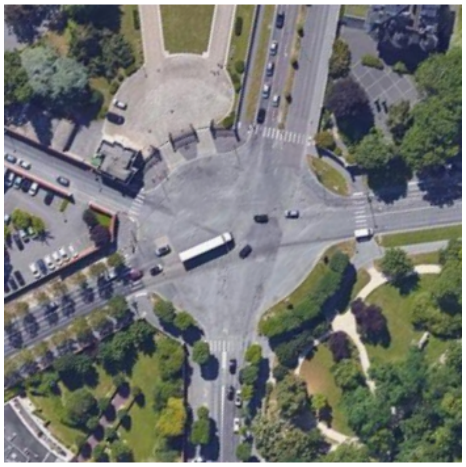


Fig. 1: A scenario example : satellite view of Pommery intersection, Reims, France.

From this scheme, we obtain a dual representation of the chord model (Fig. 4 for example)

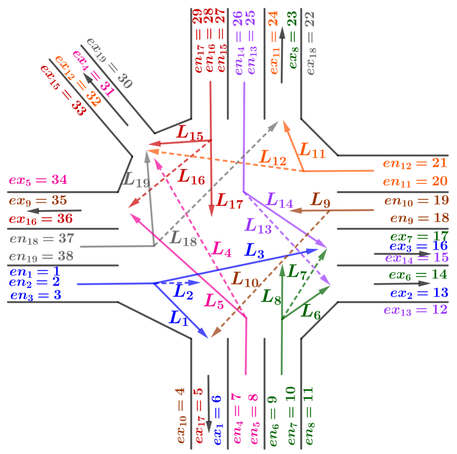
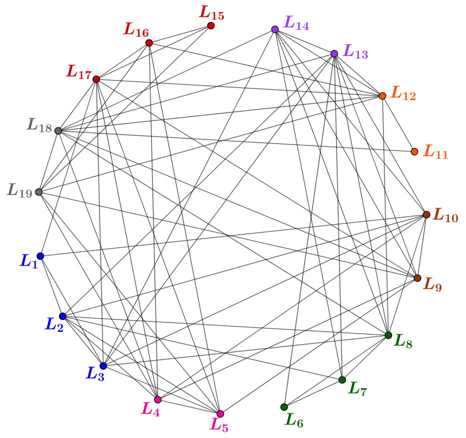


Fig. 2: A scenario example : schematic representation of Pommery intersection.



L_{15} L_{16} L_{14} L_{13} L_{17} L_{12} L_{18} L_{11} L_{19} L_{10} L_1 L_9 L_2 L_8 L_3 L_7 L_4 L_5 L_6

Fig. 4: A scenario example : the conflict graph of Pommery intersection.

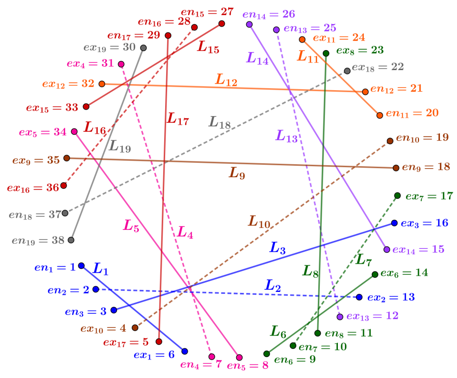


Fig. 3: A scenario example : the chord model of Pommery intersection.

denoted conflict graph where vertices are trajectories and edges are conflict between trajectories. In this model, two trajectories are linked if they are conflicting.

These representations are useful for understanding the issue but are also interesting to solve the problem.

An intersection could be modeled by a set of successive points on the circle. Each point is either an input or an output. From any number of entries and exits, S. Bai and X. Bai propose to create virtual ones so that there is an equal number of entry and exit points and that any point of entry is associated with a unique point of exit. As shown in 3, the entry and exit points are numbered successively in the trigonometric order. in The trajectory will be called a lane. If the intersection have a number n_e of entry points and n_s of exits, we obtain a maximum of n entry and exit points. They then propose the MWIS algorithm to calculate a maximum weight for an optimized set of non crossing lanes.

A difficulty is the construction of this model, that requires to multiply the number of entries and exits by creating virtual ones, such as there are as many entries as circulation lanes in the intersection. In the intersection we will use in our simulation, near Château Pommery in Reims, France (see Fig. 2 and 1), we have 8 entry points and 6 exits for a total number of 19 possible circulation lanes : this model represents this situation by 19 entries and as much exits.

IV. Our proposal

We consider a road intersection in an urban environment having a fixed topology (entries, exits and driving lanes do not change continuously, which is a reasonable assumption). We denote by SL= $(L_1, \dots L_n)$ the set of all the lanes of the intersection. Some of these lanes or trajectories are conflicting (they can not be crossed simultaneously by vehicles) and some are concurrent ones (they can be crossed simultaneously by vehicles).

A MCL, for Maximal Concurrent Lanes, is a set of concurrent lanes such that there is no other lane that is concurrent with all the lanes of the set. We propose to first determine SMCL, the set of all the MCL. Once this is done, this information remains reachable to the manager of the intersection in a dedicated memory. The number of MCL mainly depend on the local topology of the intersection. Despite the fact that theoretical abstract intersections can lead to a huge number of MCL (see VII, where the Catalan numbers are evocated), in real and concrete situations, the number of MCL seems, experimentally, very reasonable. We can see some examples - the last one being our scenario model, the other two from two other intersections from Reims - in table I.

Number of lanes	Number of sets of MCL
4	2
12	20
19	55

TABLE I: Examples of configurations

In order to compute the SMCL (see Algorithm 1), one begins by creating the set called CCL of all the couples of concurrent lanes $[L_i, L_j]$. If we note $L_i = (en_i, ex_i)$ and $L_j = (en_j, ex_j)$, en_i being the entry number of L_i, ex_i its exit number, these lanes are not concurrent if and only if $en_i < en_j < ex_i < ex_j$ or $en_j < en_i < ex_j < ex_i$.

Once this is done, we choose a lane L_i and we will determine SCL_i, the list of all the MCL containing L_i and lanes L_j such that $i < j \leqslant n$. The first step is to create the set CL_i of lanes L_j, $i < j$, such that L_i and L_j are concurrent. One can remark that $L_i \notin CL_i$ and that all the lanes in this set are concurrent to L_i, but any couple of lanes of this set are not necessarily concurrent.

In the second step, we aim to keep only the subsets of CL_i in which any couple of lanes are concurrent. We then add L_i to these subset and add it to a list SCL_i. The sets in SCL_i are not all MCL, because they are not necessarily maximal.

In the last step, one takes the sets in SCL_i one by one and verifies if it is or not included in another element of SCL_i. If it is the case, we remove this set from SCL_i. At the end of the step, SCL_i contains only MCL.

This information, SMCL, is known by the supervisor, as well as the entry and exit points of each vehicle waiting at the intersection. The supervisor is supposed to know the waiting time of each vehicle as well as the number of vehicles in each queue. With this information he can determine the weight wL of each lane L at any given time. In our simulations (see V), the weight is, in one case, the length of the queue, and in the other the waiting time of the first vehicle of each queue.

We set the time t at 0; the supervisor determine all the weights wL, then the weight wS of each MCL S by adding the weights of the lanes in S. It then selects Sa a MCL with the maximum weight - this MCL is not necessarily unique. The vehicles in the lanes of this MCL will be authorized to cross the intersection as long as the time t is inferior to a fixed value $Tmax$. When $t = Tmax$ or if the lanes of the set are empty before $t = Tmax$, the supervisor set the time t to 0 and restart the process, as described in Algorithm 2.

Algorithm 1 Algorithm for generating the sets of MCL

```
1: receive $SL$
2: $n \leftarrow |SL|$
3: **for all** $L_i$ & $L_j \in SL$ **do**
4:     **if** $L_i$ & $L_j$ concurrent **then**
5:         add $[L_i, L_j]$ to $CCL$
6:     **end if**
7: **end for**
8: **for all** $L_i \in SL$ **do**
9:     $SL_i \leftarrow L_i$
10:    **for all** $C \in CCL$ **do**
11:        **if** $C = [L_i, L_j]$ OR $C = [L_j, L_i]$ **then**
12:            add $L_j$ to $CL_i$
13:        **end if**
14:    **end for**
15:    **if** $|CL_i| = 0$ **then**
16:        add $L_i$ to $CL_i$
17:        add $L_i$ to $SCL_i$
18:    **else**
19:        $SCL_i \leftarrow$ subsets of $CL_i$
20:        **for all** $S_k \in SCL_i$ **do**
21:            **if** $\exists$ two non concurrent lanes in $S_k$ **then**
22:                remove $S_k$ from $SCL_i$
23:            **else**
24:                add $L_i$ to $S_k$
25:            **end if**
26:        **end for**
27:    **end if**
28:    **for all** $S_k$ & $S_\ell \in SCL_i$ **do**
29:        **if** $S_k \subset S_\ell$ **then**
30:            remove $S_k$ from $SCL_i$
31:        **else if** $S_\ell \subset S_k$ **then**
32:            remove $S_\ell$ from $SCL_i$
33:        **end if**
34:    **end for**
35:    add $SCL_i$ to $SMCL$
36: **end for**
37: return $SMCL$
```

We have obtained similar results as MWIS process [1] but with lower computation time, as shown in the next section.

V. Simulations

We have used our algorithms with three configurations, two from [1] and one from the Pommery

Algorithm 2 Algorithm used to determine the set of MCL authorized to cross

```
1:  receive $SL$
2:  receive $SMCL$
3:  $t$ receive 0
4:  for all $L \in SL$ do
5:      determine $wL$
6:  end for
7:  for all $S \in SMCL$ do
8:      $wS \leftarrow \sum_{L \in S} wL$
9:      add $wS$ to $WS$
10: end for
11: $M \leftarrow max(WS)$
12: $Sa \leftarrow S$ such that $wS = M$
13: while $t < Tmax \ \& \ (\exists \ L \in Sa, \ L \neq \emptyset)$ do
14:     authorize vehicles in lanes $L \in Sa$ to cross
15: end while
```

intersection, described before in this article (Fig 2 and Fig 1). The weights affected to each lane are random. The average results are in table II.

Number of lanes	MWIS algorithm (ms)	Generating sets of MCL (ms)	Maximal weight of MCL (μs)
4	0.8	0.6	15
12	7	9.6	41
19	14.5	900	140

TABLE II: Running time of algorithms

We see that our algorithm for traffic control is 100 times faster than MWIS from [1], which is not surprising. The sets of MCL are being generated only once and memorized. Our algorithm which generates those sets seems less efficient than MWIS, but our computation is being done only once by the supervisor and not repeatedly as with MWIS method. It must be updated only in case of problems on the road (road works, accidents, ...) and does it fast enough for this issue.

Let see now if our model works. We simulated the traffic, using our algorithms, in the Pommery intersection in Reims. To be precise, we will determine the average waiting time of the vehicles having crossed the intersection every minute for 3 hours. In our scenario, we suppose that the rate of arrival of vehicles at the intersection begins at 2000 vehicles per hour, then increase regularly to 5400 and then decrease to return to 2000. We can see the arrival rate in figure 5. We suppose that each vehicle can cross the intersection in at most 3 seconds, and that Tmax is 30 seconds. The arrival of vehicles is simulated by a Poisson process which mean is given by the arrival rate of 6 divided by 120 - the mean is the number of vehicles every 30s.

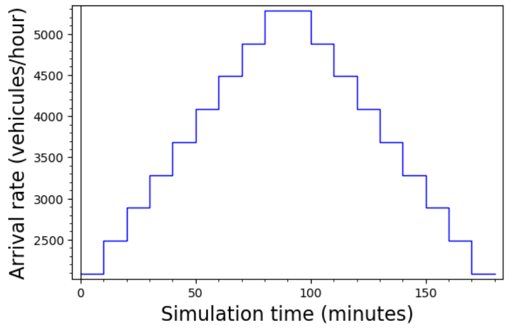


Fig. 5: Vehicle arrival rate at Pommery intersection.

We had at first to make a model of this intersection, that will, instead of 8 entries and 6 exits, have 19 of each and as much lanes.

We used two kinds of weights, one that uses only the waiting time of the first vehicle in each lane, and another that uses only the queue length of each lane. The results can be seen in Fig. 6.

Our results show an effective algorithm, as good as MWIS, but needing highly less calculus time. The average waiting time is stable for an arrival rate varying from 2500 to 4000 vehicles/hour, then increase with the arrival rate. One can observe that as soon as the arrival rate decrease, the waiting time drops very quickly to return to its initial value. Moreover, as observed in [1], the simulation using as weight the waiting time of the first vehicle of each queue and the one using the length of each queue don't seem to show significant differences. More simulations, with different configuration of intersection and other weights, will be needed to confirm this.

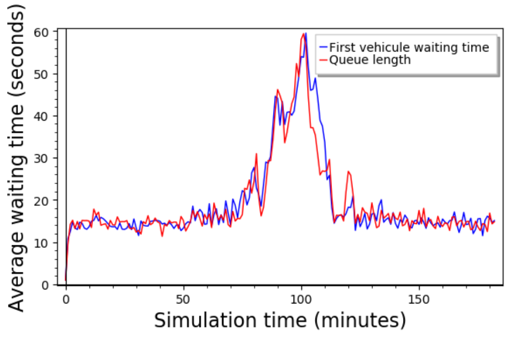


Fig. 6: Algorithm performance at Pommery intersection.

VI. Future Work and Improvement

Another point to improve is the model. It may be possible to limit the number of virtual entries and exits to create and to have a more efficient algorithm to determine the list of the sets of maximal concurrent lanes. One idea is to use DYCK words. A DYCK word is a balanced string of parenthesis. For any intersection, we will add as few as possible virtual entries and exits so that entries and exits will be alternated. We will have at most $n = 2(n_e + n_s - 1)$ such points (half entries and exits). For the Pommery intersection, we have $n_e = 8$, $n_s = 6$ but only $n = 20$ instead of $n = 38$ with MWIS method. Each entry will be followed by an exit and each exit will be followed by an entry. The point is that if all trajectories between entries and exits are authorised, a MCL can be seen as a DYCK word with size n. See the example given in Fig.7 for a 5 entries and 5 alternated exits intersection with a MCL corresponding to the DYCK word $EESESSSEESS$ (with E representing an entry and S an exit) or in an equivalent way $(()())(())$. If we note $N(n)$ the number of DYCK words with length n, we have $N(n+1) = \sum_{k=0}^{n} N(k)N(n-k)$ and we know then that, $N(n) = \frac{1}{n+1}\binom{2n}{n}$ (the nth CATALAN number, which is equivalent to $\frac{4^n}{n\sqrt{\pi n}}$) which is then an upper bound to the number of MCL. DYCK words can easily be determined. The fact is, due to the added virtual entries and exits, some of those words represent the same MCL and some represent a subset of a MCL. That does not really matter to determinate which MCL is of maximal weight, except for time efficiency. It just remains then to sort an efficient set of representatives of MCL from all DYCK words to optimize their use.

VII. Conclusion

In this article, we have shown that it is possible, for any intersection, to calculate easily an important topological information, the sets of maximal concurrent lanes, by a supervisor, and use it to calculate very efficiently a weight for each of those sets and then decide the set that will be allowed to cross the intersection.

In future work, we will try other kinds of weight, for example using the speed of increasing or decreasing in queue length, the presence of different types of vehicles (cars, trucks, priority vehicles, ...). Those parameters concern only the entry point. It will be interesting to add in the weight information from the exits, such as if a given exit is free or not.

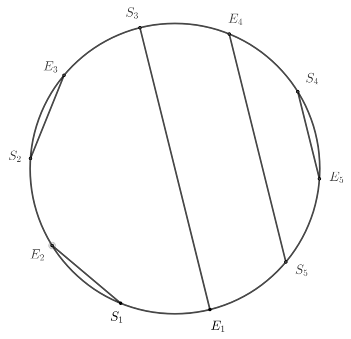


Fig. 7: A MCL corresponding to the DYCK word $(E_1(E_2S_1)(E_3S_2)S_3)(E_4(E_5S_4)S_5)$. We let indices and write both parenthesis and letters to make the associations between entries and exits easier to read.

References

[1] S. Bai, X. Bai, "A General Framework for Intersection Traffic Control With Backpressure Routing", IEEE Access, vol. 9, pp. 102125-102136, 2021. 10.1109/ACCESS.2021.3096827

[2] M. Ahmane, A. Abbas-Turki, F. Perronnet, J. Wu, A. E. Moudni, J. Buisson, and R. Zeo, "Modeling and controlling an isolated urban intersection based on cooperative vehicles," Transportation Research Part C Emerging Technologies, vol. 28, no. 3, pp. 44–62, 2013.

[3] W. Wu, J. Zhang, A. Luo, and J. Cao, "Distributed mutual exclusion algorithms for intersection traffic control," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 1, pp. 65–74, 2014.

[4] W. Ni, W. Wu, and K. Li, "A message efficient intersection control algorithm for intelligent transportation in smart cities," Future Generation Computer Systems, 2016.

[5] A. I. Morales Medina, N. van de Wouw, and H. Nijmeijer, "Cooperative intersection control based on virtual platooning," IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 6, pp. 1727–1740, 2018.

[6] D. D. Werra, C. Eisenbeis, S. Lelait, and E. St?hr, "Circular-arc graph coloring: On chords and circuits in the meeting graph," European Journal of Operational Research, vol. 136, no. 3, pp. 483–500, 2002.

[7] H. Huang, W. Fang, and H. Li, "Performance modelling of v2v based collective perceptions in connected and autonomous vehicles," in 2019 IEEE 44th Conference on Local Computer Networks (LCN), 2019, pp. 356–363.

[8] K. Yang, M. Menendez, and N. Zheng, "Heterogeneity aware urban traffic control in a connected vehicle environment: A joint framework for congestion pricing and perimeter control," Transportation Research Part C: Emerging Technologies, vol. 105, pp. 439–455, 2019.

[9] K.-H. N. Bui and J. J. Jung, "Internet of agents framework for connected vehicles: A case study on distributed traffic control system," Journal of Parallel and Distributed Computing, vol. 116, pp. 89–95, 2018.

[10] P. Lin, J. Liu, P. J. Jin, and B. Ran, "Autonomous vehicle-intersection coordination method in a connected vehicle environment", IEEE Intelligent Transportation Systems Magazine, vol. 9, no. 4, pp. 37–47, 2017.