Probability Collectives Algorithm applied to Decentralized Intersection Coordination for Connected Autonomous Vehicles

Charles Philippe[1,2], Lounis Adouane[1], Antonios Tsourdos[2], Hyo-Sang Shin[2], Benoît Thuilot[1]

Abstract— **In this paper, a multi-agent probabilistic optimization algorithm is applied to the problem of multi-vehicle coordination. The algorithm is known as "Probability Collectives" (PC) and has roots in Game Theory and Optimization theory. It is traditionally used for finding optimal solutions of NP-hard problems such as the travelling salesman problem. On the other end, the proposed PC formulation presented in this paper focuses on a minimal complexity implementation for solving the coordination problem in a time of the order of magnitude of 0.1. Besides time constraints, the emphasis in the design is put on ensuring that the algorithm always comes up with a feasible solution. Simulations show that both objectives are reached while having a decentralized algorithm, and flexible with respect to the type of situations it can deal with. Additional benefits of the PC algorithm include robustness to agent failure and the possibility to accommodate non-collaborative vehicles (market penetration of autonomous vehicles $< 100\%$).**

I. INTRODUCTION

In the last decade, autonomous vehicles have emerged as having a great potential to reduce congestion in cities and reduce casualties on the road [1]. The transition phase from only human-driven vehicles on the roads to only autonomous vehicles will be the most difficult to cope with. Some studies even underline the challenges linked to estimating the public acceptance and what passengers would be likely to accept from autonomous vehicles [2].

In the field of intersection coordination, several approaches coexist depending on the hypotheses considered by the authors. For instance, coordination can be achieved by changing the traffic lights pattern [3], by assigning slots to vehicles [4] or by direct vehicle control [5]. While some of those approaches work under the hypothesis of 100% of connected autonomous vehicles on the road [4, 6], the others focus on shorter-term hypotheses in which some vehicles present on the road would be neither connected nor autonomous [3]. In this case when not all vehicles are autonomous it is more difficult to find and enforce truly optimal solutions regarding the specific problem objectives (fuel consumption, time for crossing the intersection, ...). It is of interest to notice that even "simply" using platooning can double intersection throughput [7]. Thus, it seems that big gains are reachable despite not having a truly optimal coordination.

Some coordination techniques rely on mutual exclusion from a shared zone [4] (the centre of the intersection for example). More generally, the main difficulty in intersection coordination is to avoid conflicting motions which can lead to collisions. Mutual exclusion techniques are one way of achieving that which is compatible with human driving since it is what traffic lights achieve. This paper does not work under this hypothesis, while keeping the door open for humans to share the road. This is achieved through the use of the Probability Collectives (PC) algorithm. While the demonstrations in this paper only include autonomous vehicles, the PC algorithm has been proven to be robust to *agent failure* [8]. Its probabilistic nature also allows the insertion of probabilistic hypotheses about the behaviour of human driven vehicles without changing the way it works.

The proposed approach thus fills a gap between human-compatible intersection coordination (based on traffic light management and/or mutual exclusion of vehicles from a shared space) and optimal coordination techniques based on 100% of connected autonomous vehicles on the road. Its decentralized nature allows it to be used even without dedicated infrastructure. Other significant benefits of the algorithm are: the robustness to agent failure [8], compatibility with mixed-traffic scenarios (human drivers on the road) and its risk-averse behaviour based on its probabilistic characteristics.

In this paper the demonstrations focus on showing the useability of the proposed algorithm for road scenarios since it has never been applied this way. The examples shown are for intersection crossing but the algorithm can be used for any kind of coordination, including platooning or highway insertion. The trade-off between performance and execution time will be investigated to evaluate its potential to run in real time. The overall aim is to allow the algorithm to run a full optimization in around $0.2s$ (cf. Section III-B). This would allow reducing the distance at which the vehicles have to start synchronizing, and maybe running the optimization several times whenever a new event occurs.

The proposed method uses the Probability Collectives (PC) algorithm. It originates from the field of optimization and Game-Theory [8]. It is a decentralized agent-based algorithm based on probabilistic hypotheses and has been shown to be able to accommodate agent failure (if one of the agents is non-collaborative).

The remainder of this paper is organized as follows:

- Section II presents the state of the art on the Probability Collectives algorithm and the proposed developments to apply it to intersection coordination.
- Section III presents some experiments to show the capabilities of the proposed algorithm. The first experiment is a proof of concept with detail on the optimization variables. The second experiment is a repetition of an optimization on a fixed initial situation. The aim is

[1] Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000 Clermont-Ferrand, France `name.surname@uca.fr`
[2] Cranfield University, Cranfield, United Kingdom `name.surname@cranfield.ac.uk`

to determine how variable the result is from the PC algorithm (since it is based on a Monte-Carlo sampling, it has an element of randomness). The third experiment serves to study the effect of the search space sampling.

II. PROPOSED INTERSECTION COORDINATION ALGORITHM

A. Probability Collectives algorithm

The Probability Collectives (PC) algorithm is extensively presented in [9, 10, 8]. It is a multiagent optimization method in which different agents are playing a game iteratively. For each agent, the game consists of finding its own action (among a set of possible actions) that maximizes the overall expected utility. In mathematical uses of the PC algorithm, agents are variables of a problem and “actions” are possible values of the variables. In the proposed approach and in [12], agents are the actual vehicles trying to solve a conflicting situation and the actions are possible trajectories. The performance of the PC algorithm has been shown to be superior to classic Genetic Algoritms (GA) [11]. It can also accommodate agent failure and non-collaborative agents [10].

The PC algorithm has the following main characteristics:

- It is **probabilistic**. Each agent computes the expected utility for each of its possible actions. To do so it gets (or estimates) the probability of the actions of the other agents.
- It does not directly output a specific action, but rather a **probability distribution** $\mathbf{q}^i(\mathbf{X}^i)$ for its set $\mathbf{X^i} = (X_1^i, ..., X_N^i)$ of possible actions (for the vehicle i). An action X_k^i more likely to be the best choice will have a high probability number.
- It is **collaborative**: the probability distribution is communicated to other agents (cf. first bullet point). It has been shown that the algorithm properties and agents behaviour (rational players) make the algorithm converge to a Nash equilibrium, which is at least a local minimum of the utility function [12].
- The probabilistic aspect is coupled with **Simulated Annealing** (SA) to allow good exploration properties when starting up the algorithm and exploitation of promising solutions at the end. To achieve that, SA techniques use a “temperature” T to define an “entropy” that starts high and is close to zero at the end of the optimization.

Most of the applications of the PC algorithm are for NP-hard problems like the salesman problem, the circle packing problem or others. These applications consist of one-off offline optimizations and the goal is to find a high-quality solution with a high computational time (several minutes). To the knowledge of the authors, the only application of the PC algorithm close to the topic of this paper is presented in [12] for airplane conflict resolution. The PC algorithm is used as either a fully decentralized or semi-centralized intensive optimization algorithm. The implementation in [12] relies on a high volume of communications and a relatively complex problem formulation. It has been favourably compared to the *Iterative Peer-to-Peer Collision Avoidance* (IPPCA) algorithm which is a benchmark in the domain of airplane collision avoidance [12].

A coarse outline of the PC algorithm is presented below in Algorithm II-A from the point of view of one of the agents. For a more detailed description, the reader is invited to read [10], [12] or any reference mentioned above on the PC algorithm.

Algorithm 1 Basic outline of the PC optimization

```
Data: Own set of possible actions $\mathbf{X^i}$
Result: Probabilities vector $\mathbf{q}^i(\mathbf{X}^i)$
Initialize $\mathbf{q}^i(\mathbf{X}^i)$ to a uniform distribution
Initialize the SA temperature $T$
while no convergence of $\mathbf{q}^i(\mathbf{X}^i)$ do
  for vehicles $j \neq i$ do
    if $j$ is collaborative then
      Get set $\mathbf{X^j}$ from communication
      Get $\mathbf{q}^j(\mathbf{X}^j)$ from communication
    else
      Estimate $\mathbf{X^j}$
      Estimate $\mathbf{q}^j(\mathbf{X}^j)$
    end if
  end for
  for each $X_k^i$ in $\mathbf{X^i}$ do
    for $j \neq i$ do
      Randomly sample some $\mathbf{X^j}$ based on the probabilities $\mathbf{q}^j(\mathbf{X}^j)$
    end for
    Compute expected utility $E(X_k^i)$ of action $X_k^i$
    (based on the sampled strategies for other vehicles)
    Store $E(X_k^i)$ in a vector $\mathbf{E}(X^i)$
  end for
  Find $\mathbf{q}^i(\mathbf{X}^i)$ minimizing $f(\mathbf{E}(X^i), T)$ ($f$ is described in Eq. (2))
  Update $T$
end while
Apply action $X_{opt}^i = argmax(\mathbf{q}^i(\mathbf{X}^i))$
```

Usually the algorithm starts in a synchronized manner for all agents but there is no obligation to do so. In this paper the simulations have been run with a synchronized start of the algorithm on all vehicles. The effects of inserting a new vehicle in the optimization while it is running will be explored later.

B. Application to Intersection Coordination

The proposed formulation of the PC algorithm introduces several novelties. All the design choices have been oriented towards a low complexity and fast optimization. The problem is also very different from airplane collision avoidance because of the highly constrained environment, its dynamic nature and the low time available to solve the coordination problem. The proposed PC was also made specifically for dealing with any type of collaborative manoeuvre (intersection crossing, highway insertion, platooning). Thus the proposed algorithm and formulation should not be seen exclusively as an intersection coordination algorithm but more of a polyvalent collaborative planner.

Different aspects of the proposed PC formulation for intersection crossing will be detailed. The main contributions are the formulation of the search space, the specific 2-step optimization to break down complexity, the form of the utility function and the specific PC parameters to reach an interesting trade-off in terms of solution quality and execution time.

1) Problem formulation: In this paper, the path of the vehicles is considered fixed through the intersection. Thus the only degree of freedom is their speed. The problem geometry is thus entirely defined by the set of all the 2D paths of the vehicles on the road. This set of 2D paths will depend on the topology of the intersection. However, the only information needed by the algorithm is the shape of the paths. As a consequence, the algorithm can cope with any type of situation as long as some paths are defined. An example of such a situation is shown in Fig. 1 for a "cross" intersection and a roundabout.

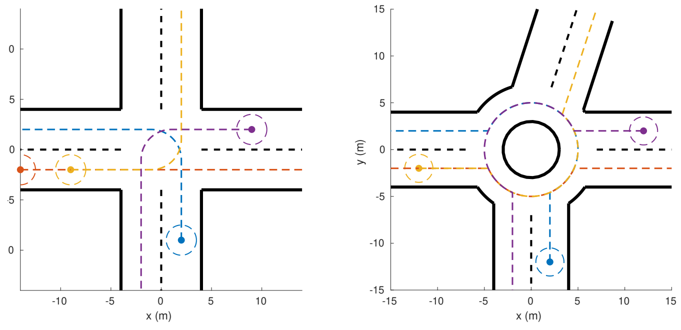

Fig. 1. Example of intersection configuration and paths of the vehicles. The vehicles are at their initial position and have to follow the dotted paths to their intersection exit at a suitable speed.

In the proposed PC formulation, an element of the search space is a preset speed profile over the whole intersection. Thus, the size of the search space does not depend on the length of the conflict zone but on the number of sampled speed profiles considered. Changing this number is an easy way to modify the complexity of the optimization. The consequences of that will be discussed later.

An illustration of the possible actions (speed profiles) of a vehicle at each step of the optimization is shown in Fig. 2. On the upper plot, there are 10 available options to the vehicle. These options are all composed of an acceleration/deceleration of constant magnitude to a fixed speed. In this formulation the speed profiles could be summed up as a couple (a_{max}, v_{end}) (acceleration, end speed). However, it shall be noted that this set could contain randomly shaped speed profiles and is not limited to such simple shapes. Any speed profile of the form $v = f(t)$ (where f is any continuous function that respects the vehicle's dynamics) could be considered. On the bottom plot, the available options are 10 speed profiles that all start following the profile chosen at phase 1 and have reaccelerations to a nominal speed at different times. The available options could have any shape that the designer sees fit.

For the intersection application, the optimization is done in two steps (Fig. 2). First of all the agents are looking for a speed profile with a fixed end speed ("Phase 1" subplot) that allows them to avoid any collision. At this step, the algorithm is guaranteed to find a feasible solution if it is started when the vehicles are far enough upstream of the intersection. This way, the vehicles have the option to come to a complete stop (or an arbitrary low speed, here $0.1m/s$) before the shared zone. However their aim is to find a better speed profile that allows them to clear the intersection without coming to a stop and without colliding with any other vehicle. The second step is a reacceleration to a speed that allows the vehicles to clear the intersection as fast as possible while maintaining the collision-free characteristic of the solution.

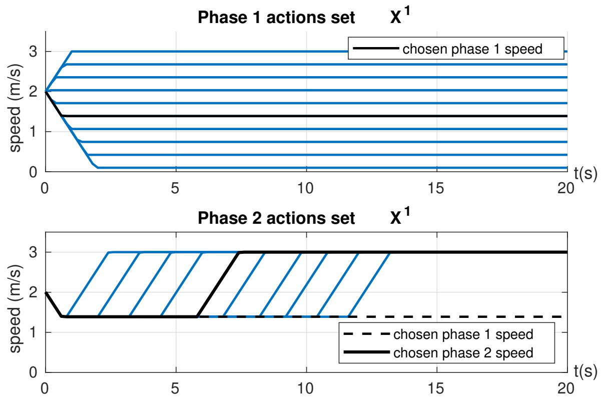

Fig. 2. Search space representation with two-step optimization

To ensure a sampling of good quality during the first step, the final speeds have to span:

- A wide range, that needs to include $0m/s$ (a complete stop, to ensure the availability of a safe solution) and v_{max} (the maximal speed in the intersection for fast clearing)
- Closely "enough" spaced speeds so that the vehicle can "squeeze" in slots between other vehicles.

If the speed profiles sampling is too coarse, some solutions could be unattainable. For example, a car willing to insert itself on a lane between two other cars could be unable to find a suitable speed profile if its options are too coarse. One action will be too fast and the next slightly slower variation will be too slow. Thus, it will lead to a suboptimal solution through suboptimal use of space.

2) Objective Function: In this paper, the proposed local utility to minimize is as follows for an agent i:

$$\begin{aligned} J(X) = & W_{sep} \sum_{i_v \neq i} \sum_{k=1}^{k_{max}} \frac{1}{d_k(i_v, i_{\text{self}})^2} \\ & + W_{speed}(v_{max} - v_{avg})^2 \\ & + W_{control} \sum_{k=1}^{k_{max}} |v_{i_{\text{self}}}(k) - v_{i_{\text{self}}}(0)| \end{aligned} \quad (1)$$

Where $d_k(i_v, i_{\text{self}})$ is the distance between the ego vehicle and the vehicle i_v at time step k. The sum over k is for all the time steps. The first term penalizes low separation distances between the vehicles. The second term penalizes slow average speeds through the intersection, and thus favours a fast crossing. The last term penalizes the control effort, that is the deviation from the initial speed of the vehicle when it approaches the intersection.

Constraints are imposed on the separation distance. For simplicity the vehicles are represented as discs and thus the separation constraint is a distance between the vehicles centres.

As usual in the PC framework, the vehicle i will not directly optimize this function. Instead, it will find a probability distribution $\mathbf{q}^i(\mathbf{X}^i) = (q^i(X_1^i), ..., q^i(X_N^i))$ for its set of N actions $\mathbf{X}^i = (X_1^i, ..., X_N^i)$ such that:

$$\mathbf{q}^i(\mathbf{X}^i) = argmin\left(\sum_{k=1}^{N} q^i(X_k^i)E(J(X_k^i)) - TS(\mathbf{q}^i(\mathbf{X}^i))\right) \quad (2)$$

Where $E(J(X_k^i))$ is the expectancy of the utility function J for the strategy X_k^i of the vehicle i. It is computed by randomly sampling the other vehicles' strategies and computing the obtained utility J. The obtained utility is averaged over several samplings of the other vehicles' strategies. This random sampling is done according to the latest probability distribution of the actions of the other vehicles', as they communicated it to vehicle i. This distribution is denoted $\mathbf{q}^j(\mathbf{X}^j)$ (for $j \neq i$). The parameter T is specific for the Simulated Annealing (SA) and is called the temperature. The function S is an entropy defined as:

$$S(\mathbf{q}^i(\mathbf{X}^i)) = -\sum_{k=1}^{N} q^i(X_k^i)\ln(q^i(X_k^i)) \quad (3)$$

At the beginning of the PC optimization, the parameter $T \in \mathbb{R}$ is big, which weighs the entropy term more. If T is infinite, the optimal $\mathbf{q}^i(\mathbf{X}^i)$ is uniform and the $q^i(X_k^i)$ are all equal to $1/N$. This favors the exploration of all possible solutions. At the end, T is brought close to zero to weight the expectancy term more.

As the speed profiles span the whole duration of the intersection crossing, the PC optimization can be run just once. This comes from the fact that the shared road space has finite dimensions and well-defined bounds. One run of the PC algorithm corresponds to several iterations in which the probabilities of each action are exchanged at each iteration between the vehicles as they update their probability distribution.

It shall be noted that the probability distributions $\mathbf{q}^j(\mathbf{X}^j)$ of other vehicles' actions may not be available. For instance, some vehicles may be human-driven. In that case, an estimation of the probability distribution should be carried out based on sensory information: use of blinkers, speed of entry...

C. Comparison with existing work using PC

Several key differences exist between our approach and the approach for airplane collision avoidance presented in [12]. In this airplane collision avoidance approach, the search space is a succession of heading changes over a rolling horizon. The headings are determined one by one at regular time intervals. In our approach, an element of the search space is a whole speed profile over a fixed time horizon, greatly limiting the complexity of the search space. In [12] the environment is not dense (airspace) so a solution can always be found. The minimal separation distance is implemented as a soft constraint. The emphasis is not on finding a feasible solution (easy to do in this case) but on finding a really optimal solution. In our approach the space is very constrained: there are a lot of vehicles in a small area. The emphasis is on finding a feasible collision in a very short time that corresponds to the time constants found in ground traffic.

In [12], objective functions are shared and agreed upon so they are *homogeneous* through the range of all agents. It is interesting to note that the optimization runs at regular interval when the horizon has moved. Thus, the future collisions (or separation distance violations) are less penalized than the imminent collisions in the cost function. Because of the heavily constrained space for traffic management, the horizon of the optimization has been chosen to span the whole intersection. This is possible for ground traffic because the boundaries of the shared zone are usually clearly identified.

Finally, a typical optimization in [12] will see 800MB of data exchanged between airplanes in the fully decentralized version. The time spent for optimization is up to 120s for a round of PC optimization with 25 aircraft. There is a linear complexity dependence for the decentralized PC implementation under some realistic hypotheses (broadcasting messages about partial costs and predictions). The semi-centralized PC operation takes longer, at 610s for the same conflict. The fully decentralized approach is thus better at scaling. In the proposed PC formulation and implementation, the data exchange is intended to be minimal and a solving time of $0.2s$ is targeted ($0.8s$ have been achieved so far for 4 vehicles on non-optimized Matlab code). The typical data exchange would be in the range of several MB (detailed in section III-A).

III. EXPERIMENTS

This section presents experiments done with a first implementation in Matlab. Simulations show a cross intersection scenario, but the work is generalizable to any type of collaborative maneuver.

The algorithm has been implemented in its fully decentralized version for better scaling [12].

A. Proof of concept

This section details a simulation done on a given initial situation with 4 vehicles at a cross intersection. Table I shows the main parameters chosen for the PC algorithm.

The sampling time is used to represent the speed profiles and to integrate the positions of the vehicles over the time horizon. This information is used to compute the cost J. In this simulation, the cost function contains only the terms on the average crossing time (altruistic objective) and on the separation distance. If a solution violates the separation constraint (represented by the dotted circles in Fig. 3), an additional cost J_{cons} is added for each constraint violated. The algorithm stops if the global solution does not change for N_{stop} iterations. The weight on the control effort has been left at zero for the moment to focus on the "altruistic" terms of the utility function. The $M1$ column is the regular mode of the PC algorithm for fast optimization, and the $M2$ mode has been used as a longer optimization to find an optimal solution. The annealing schedule is much slower and starts at a higher temperature to allow the algorithm to explore more possibilities. The number of possible actions is also higher. This mode will be used in section III-B.

A snapshot of the results of the optimization is shown in Fig. 3. The vehicles are represented as circles so far for the sake of simplicity. The solution seems intuitive, because it allows the two vehicles with the highest entry speed (yellow and orange) to carry their speed through the intersection. The purple (resp. blue) vehicle lowers its speed just enough to yield to the yellow (resp. orange) vehicle. Thus during this fully decentralized optimization, the algorithm has arrived at a relevant solution without any constraint violation.

At the beginning of the simulation, each vehicle broadcasts its set of strategies to other vehicles. Each strategy is a *float* vector of size 200 ($40s$ horizon and $0.2s$ sampling time) and the set has 10 strategies. That is a total of 2000 floats exchanged per vehicle, or 8kB (kilobytes). This is done again at the beginning of the phase 2 (reacceleration, cf. subsection II-B.1). Then for each iteration the vehicle broadcasts its updated probability vector $\mathbf{q}^i(\mathbf{X}^i)$ of 10 floats. The optimization in the $M1$ mode runs in around 20 iterations so it will be a total broadcasted per vehicle of 200 floats (0.8kB). For 4 vehicles, the data volume broadcasted will be around 16.8kB per vehicle and 67.2kB in total. If the objective of doing the optimization in $0.2s$ is respected, it means the required network thoughput should be in the order of magnitude of 0.4MB/s. Of course this does not consider the overhead due to the communication protocols, and considers that messages can be broadcasted. Even with a pessimistic hypothesis of a requirement ten times superior, it would stay well within what is physically possible (4MB/s required).

B. Analysis of consistency on fixed initial situation

For this series of simulations, the initial situation has been fixed to be the same as in section III-A. The algorithm has been run several times to check consistency. For reference, the performance metric distribution has been compared with the performance found with a run of the algorithm in the $M2$ mode (slower and more "optimal" solution). The results are shown in Table II. It shall be noted that the algorithm provided solutions with no constraint violation in 100% of the cases.

TABLE I
MAIN PC PARAMETERS

Parameter	Notation	Value	
		M1	*M2*
Number of strategies	N_s	10	20
Sampling time	T_s	$0.2s$	
Weight on control effort	$W_{control}$	0	
Weight on separation dist.	W_{sep}	1	
Weight on avg. crossing time	W_{avg}	10	
Penalty for constraint violation	J_{cons}	10^5	
Samples to get expected utility	$N_{samples}$	10	20
Stopping criteria	N_{stop}	4	10
SA start temperature	T_{init}	1	10
SA end temperature	T_{end}	0	0
SA temperature step	T_{step}	0.2	0.66

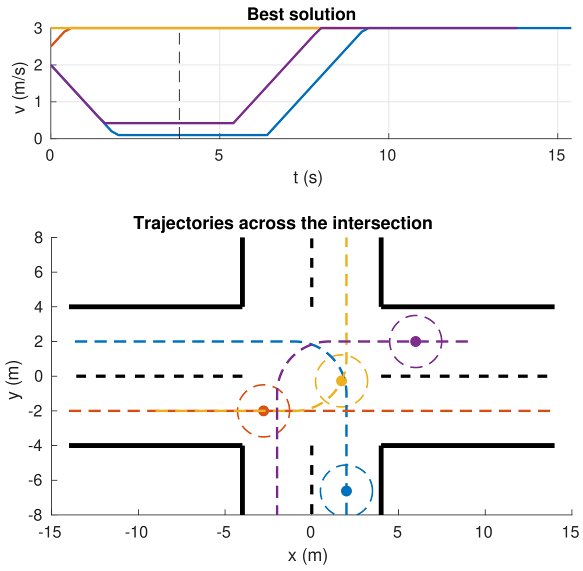

Fig. 3. Snapshot of the application of the best solution when $t = 4s$. The purple vehicle is entering from the right lane and will exit at the bottom. It slowed down just enough to yield for the yellow vehicle (yellow vehicle came from the right, exits at the top). The red vehicle (coming from the left, exits to the right) accelerated up to its maximal allowed speed so that the blue one has to wait the minimal amount of time before entering the intersection (blue enters from the bottom). Full video available at `https://youtu.be/XTgY-4RUFz0`

TABLE II
RESULTS OF THE PC CONSISTENCY TEST (100 RUNS)

Parameter	Notation	Distribution
$M1$ mode		
Average crossing time	$t_{avg,M1}$	$8.4s \pm 0.7s$
Max crossing time	$t_{max,M1}$	$12.5s \pm 1.9s$
Execution time	$t_{exec,M1}$	$3.3s \pm 0.4s$
$M2$ mode		
Average crossing time	$t_{avg,M2}$	$7.8s \pm 0.2s$
Max crossing time	$t_{max,M2}$	$10.6s \pm 0.2s$
Execution time	$t_{exec,M2}$	$14.6s \pm 1.0s$

The distribution is shown in Fig. 4. The distribution in the histograms seems discrete for the average time and max time because of the discrete nature of the search space. The algorithm used in *Mode 1* found most of the time a single solution yielding a crossing time of $8.5s$, and sometimes solutions yielding either $t_{avg} = 7.5s$ or $t_{avg} = 9.5s$.

When running the algorithm in the more precise mode $M2$, the crossing times for the vehicles are slightly improved in terms of average value and highly improved in terms of standard deviation. This comes however with a 4.4 times higher computation time. The fast mode (and low complexity) of the PC optimization thus seems to give a good balance between the quality of the solution and the execution time. The overall consistency is quite good and even the suboptimality is not critical. The main interest of the M2 mode is the increased consistency. In both cases, the best solution the algorithm can output (at $7.2s$ of average crossing time) has only been found a handful of times.

It shall be noted that computation time is given for a Matlab simulation where the code for all the vehicles is not parallelized. The execution time *per vehicle* would be 4 times less for these conditions with the same code. It means that would be around $0.8s$ in this case. It is expected that the code could run much faster when implemented in C++. With the current figure or a vehicle going at $14m/s$ ($50km/h$), it means that the synchronization should start around $11m$ before the point where any braking should start. For reference, an emergency braking to a full stop at this speed is around $14m$ on modern cars and dry roads. Thus, the distance needed for synchronization is of an acceptable magnitude for the considered application. The authors expect to bring the optimization time down to $0.2s$ ($3m$ travelled at $50km/h$).

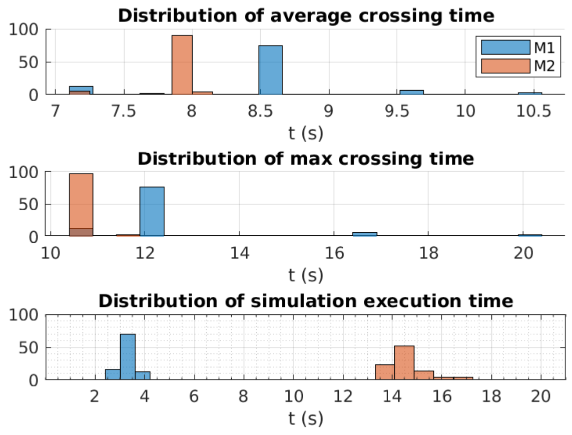

Fig. 4. Results of 100 runs of the PC optimization for the same initial conditions. Mode 1 in blue and Mode 2 in orange.

C. Influence of search space sampling

In this experiment the effect of changing the number of available actions is tested. The *available actions* refer here to several speed profiles that the vehicle can choose from to cross the intersection. Instinctively, if a vehicle has fewer actions to chose from, it should show a decrease in performance. The simulations are done on the same settings as before on a cross intersection and with 4 vehicles. Optimization parameters are the same as shown in Table I (Mode 1). The number of available actions is changed between 4 and 14.

The results are shown in Fig. 5. The average performance does not really decrease (average crossing time, max crossing time) but more and more outliers appear on the middle plot (distribution of the crossing time of the last vehicle). This means that the algorithm struggles to guarantee consistent performance. The outliers most likely correspond to situations where an additional intermediate action should have been undertaken to squeeze between two (or several) vehicles.

However, the execution time plot shows a significant decrease in execution time. For a search space of size $N_{strats} = 4$, the average execution time is $1.7s$. This corresponds to $0.4s$ per vehicle, and $5.6m$ travelled at $14m/s$ ($50km/h$). With this setting, the algorithm starts to show real time capabilities (considering the code is not yet optimized) but fails to propose refined enough solutions in some cases. Ideally, a very low number of available options could be kept if a post-processing of the speed profile is applied: it is easy to detect if a vehicle waits more than is necessary because of not enough available options. It would keep the execution time low but would not solve the situations where a vehicle failed to find a solution to “squeeze” between other vehicles.

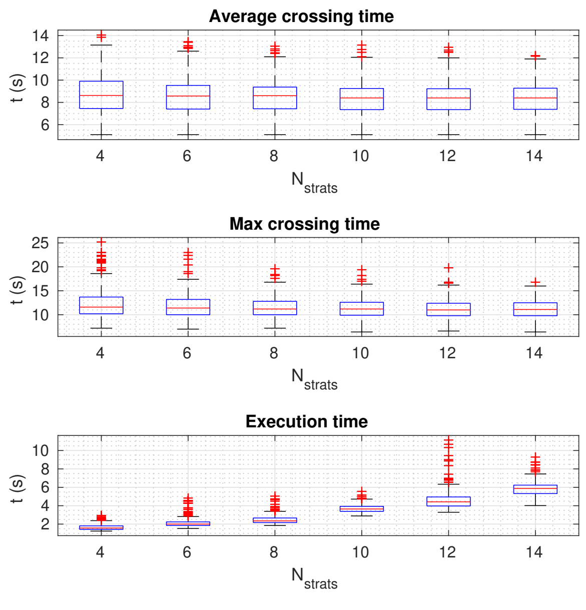

Fig. 5. Effect of the size of the search space on the performance of the coordination algorithm.

IV. CONCLUSIONS

A novel formulation of the Probability Collectives (PC) algorithm has been presented to apply it to ground traffic coordination. The main interests of the proposed algorithm are its low complexity compared to usual PC applications and its flexibility to any kind of road scenario. Its capabilities have been demonstrated for an intersection crossing in which the space is highly constrained. The algorithm exhibits good exploration properties to find relevant solutions in a very short time ($0.8s$ in average for the demonstrated scenarios). It is also fully decentralized and can be applied to collaborative manoeuvre. Trade-offs between performance, consistency and execution time have been highlighted. The current performance hints towards true real-time implementation of the proposed algorithm. In order to further minimize the execution time, it is planned in the near future to switch to a ROS-based C++ implementation.

Further work will focus on practical implementation problems. For example the algorithm should re-run if a new vehicle wants to join the collaboration. In this case some continuity of the already computed speed profiles shall be enforced. Furthermore, the algorithm will be tested in situations where the cost functions of the vehicles are non-homogeneous. Non-collaborative vehicles or human-driven vehicles will also be inserted to check the robustness of the proposed algorithm to such elements.

ACKNOWLEDGMENT

This work has been sponsored by the French government research program Investissements d'avenir through the RobotEx Equipment of Excellence (ANR-10-EQPX-44) and the IMoBS^3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program Regional competitiveness and employment 2014-2020 (FEDER - AURA region) and by the AURA region.

REFERENCES

[1] Daniel J. Fagnant and Kara Kockelman. "Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations". In: *Transportation Research Part A: Policy and Practice* 77 (2015), pp. 167–181.

[2] Scott Le Vine, Alireza Zolfaghari, and John Polak. "Autonomous cars: The tension between occupant experience and intersection capacity". In: *Transportation Research Part C: Emerging Technologies* 52 (Mar. 2015), pp. 1–14.

[3] Hironori Suzuki and Yoshitaka Marumo. "A New Approach to Green Light Optimal Speed Advisory (GLOSA) Systems for High-Density Traffic Flow *". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018), pp. 1–6.

[4] Robert Hult, Gabriel R. Campos, Paolo Falcone, and Henk Wymeersch. "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections". In: *Proceedings of the American Control Conference*. Vol. 2015-July. IEEE, July 2015, pp. 763–768.

[5] Stefanie Manzinger and Matthias Althoff. "Tactical Decision Making for Cooperative Vehicles Using Reachable Sets". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018).

[6] Andreas A. Malikopoulos, Christos G. Cassandras, and Yue J. Zhang. "A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections". In: *Automatica* 93 (2018), pp. 244–256. arXiv: `1602.03786`.

[7] Jennie Lioris, Ramtin Pedarsani, Fatma Yildiz Tascikaraoglu, and Pravin Varaiya. "Platoons of connected vehicles can double throughput in urban roads". In: *Transportation Research Part C: Emerging Technologies* 77 (2017), pp. 292–305. arXiv: `1511.00775`.

[8] Anand J Kulkarni and Kang Tai. "A Probability Collectives Approach for Multi-Agent Distributed and Cooperative Optimization with Tolerance for Agent Failure". In: *Agent-Based Optimization*. Ed. by Ireneusz Czarnowski, Piotr J\kedrzejowicz, and Janusz Kacprzyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 175–201.

[9] Michalis Smyrnakis and David S Leslie. "Sequentially updated Probability Collectives". In: *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference* (2009), pp. 5774–5779.

[10] Anand J. Kulkarni and K. Tai. "Probability Collectives: A multi-agent approach for solving combinatorial optimization problems". In: *Applied Soft Computing Journal* 10.3 (June 2010), pp. 759–771.

[11] Chien-feng Huang, Los Alamos, Aero Astro, David H Wolpert, Moffett Field, and Charlie E M Strauss. "A Comparative Study of Probability Collectives Based Multi-agent Systems and Genetic Algorithms". In: (2005), pp. 751–752.

[12] David Šišlák, Pemysl Volf, Michal Pěchouček, and Niranjan Suri. "Automated conflict resolution utilizing probability collectives optimizer". In: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 41.3 (May 2011), pp. 365–375.