A Decentralised Approach to Intersection Traffic Management

Huan Vu[1,2], Samir Aknine[1] and Sarvapali Ramchurn[3]
[1] Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, Lyon 69622, France
[2] University of Transport and Communications, Hanoi, Vietnam
[3] University of Southampton, Southampton, United Kingdom
huan.vu@liris.cnrs.fr, samir.aknine@univ-lyon1.fr, sdr1@soton.ac.uk

Abstract

Traffic congestion has a significant impact on quality of life and the economy. This paper presents a decentralised traffic management mechanism for intersections using a distributed constraint optimisation approach (DCOP). Our solution outperforms the state of the art solution both for stable traffic conditions (about 60% reduced waiting time) and robustness to unpredictable events.

1 Introduction

With the growth in urbanisation and ownership of cars, most major cities around the world suffer from high rates of traffic congestion, with significant impact on the economy and human wellbeing. In the US alone, urban congestion costs 6.9 billion hours of travel delay, and 3.1 billion gallons of wasted fuel per year [Texas A&M Transportation Institute, 2015]. In addition, pollution due to petrol and diesel cars at stand still at major traffic intersections can rise to more than 29 times than in normal free flow traffic conditions [Goel and Kumar, 2015]. With the rise of autonomous and electric vehicles, it is believed that cars will be able to coordinate at intersections to minimise delays and thus reduce the time and energy wasted. However, a number of challenges need to be addressed before such autonomous coordination can be implemented in the real-world. First, cars may arrive at any time at an intersection, each with its own urgency or priority to reach a destination. For example, emergency vehicles need to be prioritised over normal leisurely journeys, and commercial traffic may need to be prioritised at business hours. Secondly, the solution to this coordination problem needs to guarantee the safety of passengers by placing safeguards to avoid collisions at the intersection. This also requires that a coordination algorithm needs to be computationally efficient and return solutions that are safe. Third, and most importantly, intersection management algorithms need to be robust to sudden surges in demand across the intersection from vehicles with varying degrees of priority.

Now, various traffic control methods have been developed to optimise traffic flow at intersections, focusing on the control of the right-of-way. Dresner and Stone proposed a right-of-way reservation mechanism for autonomous vehicles [Dresner and Stone, 2008]. It relies on a FCFS (First Come First Served) policy, granting the right-of-way to each vehicle requesting, as quickly as possible. This mechanism takes into account human drivers by using a classical traffic light policy for human drivers, and giving the right-of-way on red lights to autonomous vehicles using the FCFS policy. However, as we show in this paper, the FCFS policy, while being computationally efficient, results in chaotic behaviours when sudden changes happen in traffic conditions while, generally producing poor solutions when large numbers of vehicles converge on an intersection. In turn, Vasirani and Ossowski proposed a market-based system where drivers have to purchase reservations from the intersection managers in order to cross intersections [Vasirani and Ossowski, 2012]. However, this tends to result in longer waiting times than for FCFS at single intersections as they focus on economic efficiency rather average travel time. [Carlino *et al.*, 2013] proposed an auction based intersection management mechanism where drivers continuously bid for reservations. This paper has shown that auctions can be applied to control autonomous vehicles, but did not propose an optimisation to a vehicle's bidding strategy.

Against this background, we propose a novel right-of-way assignment mechanism at a single intersection with the goal of minimising the average travel time across the intersection. We focus on approximate solutions that can return good solutions in real-time and account for the individual position, direction, and speed of each vehicle. Hence, we formulate the right-of-way allocation problem as a Distributed Constraint Optimisation Problem (DCOP) which has been show to be effective in task allocation and meeting scheduling problems [Macarthur *et al.*, 2011; Farinelli *et al.*, 2008; Modi and Veloso, 2004]. This decentralised approach has the added benefit of distributing some of the computation across all available computational nodes (e.g., in cars or at junctions) to find solutions quickly and is adaptable to additions or departures of cars at all times. In a similar vein, [Junges and Bazzan, 2008] demonstrated the performance of DCOP solvers to the traffic light optimisation problem. However, due to the nature of microscopic traffic regulation, where the state constantly changes and calculation time is a crucial part, these approaches cannot be directly applied to solve the problem. Instead, our approach focuses on discretisation of the intersection and traffic flow that is more computationally manageable while still satisfying the constraints of the right-of-way allocation problem. More specifically, this paper advances

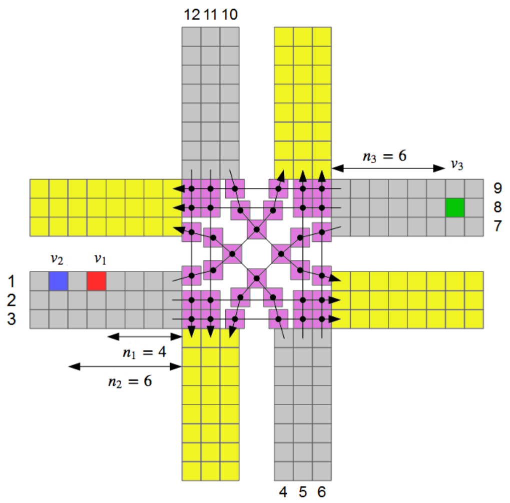

Figure 1: Intersection with 12 incoming lanes (in gray), 12 outgoing lanes (in yellow) and a conflict zone (in purple), all divided in cells. The incoming lanes are numbered from 1 to 12. The conflict zone is crossed by various trajectories. The cells belonging to several trajectories (every cell of the conflict zone in this case) are conflict spots. There are 3 vehicles v_1 (red rectangle), v_2 (blue rectangle) and v_3 (green rectangle). v_1 and v_2 are heading north, while v_3 is heading west.

the state of the art in the following ways. First, we propose a novel DCOP formulation of the right-of-way allocation problem. Second, we show how to solve the DCOP approximately using the max-sum algorithm [Farinelli *et al.*, 2008; Macarthur *et al.*, 2011]. Third, we empirically show that our algorithm outperforms the state of the art in terms of reductions in waiting time and robustness to dynamic events.

2 Problem Statement

We model an intersection using a cellular automaton model (cf. Figure 1). This model is widely used in literature because it retains the main properties of a network while being relatively simple to use [Brockfeld *et al.*, 2001; Maerivoet and Moor, 2005]. An intersection is composed of several *incoming lanes*, several *outgoing lanes*, and a central zone called *conflict zone*. The path of a vehicle across the intersection is called a *trajectory*. Each incoming lane and trajectory is a succession of cells. A cell inside the conflict zone is called a *conflict spot*.

The main objective of the system is to minimise travel time. The travel time of a vehicle consists of the time it needs to travel through its journey at its speed, and a waiting time. Thus, to minimise the travel time, we must minimise the waiting time of vehicles. Our objective is to assign to each vehicle an admission time to the conflict zone. A vehicle's admission time is the time that this vehicle can begin crossing the intersection, similar to an individual traffic light system. We define, for each time step t a *configuration* Φ_t as the set of admission time of vehicles in front of the conflict zone.

This configuration must satisfy the following rules: (i) the configuration must ensure that vehicles can cross the intersection at their admission time safely and without stopping inside the conflict zone, (ii) a vehicle must have only one admission time at a time, (iii) the current configuration must be accessible by all vehicles so they share the same agreement any time. In order to build this configuration, we model the right-of-way allocation problem as follows.

Definition 1. Let t be the current time step and V_t the set of all vehicles approaching the intersection. A configuration is a set $\Phi_t = \{\varphi_1, ..., \varphi_k\}$ where each φ_i is the admission time in the conflict zone assigned to each $v_i \in V_t$.

Let L be the set of incoming lanes and $l_k \in L$ for lane k. For each $v_i \in V_t$, let $l_{v_i} \in L$ be the lane in which vehicle v_i is present, n_i be the distance (in number of cells) between v_i and the conflict zone, and τ_i be v_i's trajectory inside the conflict zone. Let e be one of the cells in trajectory τ_i, $pos(e, \tau_i)$ is the distance, in number of cells, between the cell e and the first cell of τ_i. The position of the first cell of τ_i is 0. Let s_i be the speed of the vehicle v_i in cells per time step.

We aim to build, for each time step t, a configuration Φ_t for all vehicles in V_t that minimises their total waiting time. The input is the set of vehicles V_t presented in the system at the current time step and the configuration at the last time step Φ_{t-1}. Let w_i be the waiting time of vehicle v_i and Φ be the set of all possible configurations (this waiting time can be changed to weighted waiting time to take into account a vehicle's priority). Thus our goal is to search for a minimisation:

$$f : (t, V_t, \Phi_{t-1}) \mapsto \arg\min_{\Phi_t \in \Phi} \sum_{v_i \in V_t} w_i \tag{1}$$

To ensure that the configuration Φ_t satisfies the rules described above, the admission times of vehicles in the configuration must follow some structural constraints.

c1. Distance constraint A vehicle has to cross the distance separating it from the conflict zone before entering it:

$$\forall v_i \in V, \varphi_i > t + \frac{n_i}{s_i} \tag{2}$$

c2. Anteriority constraint In our model, we consider that no overtaking is possible when vehicles are close to the intersection. Thus a vehicle v_j cannot enter the conflict zone before the vehicles v_j preceding it on its lane. This constraint should be modified in a more complex model that takes into account overtaking. We have:

$$\forall v_i, v_j \in V_t^2, l_{v_i} = l_{v_j}, n_i < n_j \Rightarrow \varphi_i < \varphi_j \tag{3}$$

c3.a Simple conflict constraint Two vehicles cannot be in the same cell at the same time in the conflict zone. If the vehicles belong to the same lane, the anteriority constraint covers this case. However, if two vehicles v_i and v_j coming from different lanes, having a conflict spot in their trajectories, their admission times must ensure that they are not present in the conflict spot at the same moment. Thus, we have:

$$\begin{gathered}\forall v_i, v_j \in V_t^2, \forall e \in \tau_i, e \in \tau_j \Rightarrow \\ (\varphi_i + \tfrac{pos(e,\tau_i)}{s_i}) \neq (\varphi_j + \tfrac{pos(e,\tau_j)}{s_j})\end{gathered} \tag{4}$$

c3.b Conflict constraint with safety lapse We can further restrict constraint c3.a for safety reasons. Indeed, adding a time lapse t_{safe} between the passing of a vehicle v_i on a cell c and the passing of another vehicle v_j, in a conflicting trajectory on this cell, enhances the drivers' safety. Thus, v_j can only occupy this cell after a t_{safe} duration of v_i's occupation. The simple conflict constraint can be replaced by the following:

$$\forall v_i, v_j \in V_t^2, \forall e \in \tau_i, e \in \tau_j \Rightarrow \\ |(\varphi_i + \frac{pos(e,\tau_i)}{s_i}) - (\varphi_j + \frac{pos(e,\tau_j)}{s_j})| > t_{safe} \quad (5)$$

Example 1. Consider the scenario presented in Figure 1. Assuming the speed of all vehicles is 1 cell/time step, the structural constraints can be described as:

c1: v_1 has 4 cells to travel before entering the conflict zone, thus $\varphi_1 > 4$. By the same logic, $\varphi_2 > 6$; $\varphi_3 > 6$.

c2: v_2 cannot overtake v_1, therefore $\varphi_2 > \varphi_1$.

c3.b: There is a conflict spot between the trajectory of v_1 and that of v_3. The conflict spot is the cell number 4 in v_1's trajectory and the cell number 2 in v_3's trajectory. Let the safety lapse be 1 time step, we have: $|(\varphi_1+4)-(\varphi_3+2)| > 1$. v_2 has the same conflict spot with v_3, we also have $|(\varphi_2+4)-(\varphi_3+2)| > 1$.

We next formalise the right-of-way allocation problem as a distributed constraint optimisation problem.

3 DCOPs for Intersection Management

Centralised solutions to traffic regulation result in high computational requirements for one agent. Moreover, centralised approaches create a single point of failure and have a lack of scalability and adaptability to dynamic events such as accidents or the arrival of an emergency vehicle. In such a dynamic context, using a decentralised approach allows to be proactive to any change in traffic control. This is particularly relevant in the light of connected vehicle capable of advanced computations. In this paper, we present a decentralised formalisation of traffic regulation model using a DCOP. This formalisation allows every agent to coordinate by exchanging messages with their neighbours, thus reduces the computational requirements for each agent.

A Distributed Constraint Optimisation Problem (or DCOP) is a tuple $\{\mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}\}$, where: $\mathcal{A} = \{a_1, ..., a_n\}$ is a set of n agents; $\mathcal{X} = \{x_1, ..., x_n\}$ are variables owned by the agents, where variable x_i is owned by agent a_i; $\mathcal{D} = \{\mathcal{D}_{x_1}, ..., \mathcal{D}_{x_n}\}$ is a set of finite-discrete domains. A variable x_i takes values in $\mathcal{D}_{x_i} = v_1, ..., v_k$; $\mathcal{C} = \{c_1, ..., c_m\}$ is a set of constraints, where each c_i defines a cost $\in \mathbb{R} \cup \{\infty\}$. A solution to the DCOP is an assignment to all variables that minimise $\sum_i c_i$.

There are several ways to formalise our problem as a DCOP, depending on what agents, variables and constraints represent. Here we present two approaches to formalise the traffic regulation as a DCOP, a fully decentralised vehicle-based approach and a semi-decentralised lane-based approach to show the effect of different levels of decentralisation on the quality of solution and the computational time. We evaluate and show the performance of each approach which may be suitable for different traffic conditions.

3.1 Vehicle-based Approach

The vehicle-based approach consists of modelling all the vehicles as agents. The number of agents is also the number of vehicles arriving at the intersection. Each agent holds a variable which corresponds to the vehicle's admission time to the intersection. The domain of the variables varies from $t + \frac{n_i+1}{s_i}$, which is the earliest possible admission time of this vehicle taking into account its distance to the conflict zone, to $t + \frac{n_i+1}{s_i} + p$. p is the time window for the waiting time of each vehicle. A small window may limit the search and makes it impossible to find a solution, while a large window adds unnecessary complexity to the problem. The value of this time window will be detailed in Section 5.2. Since the domain of the variables already takes into account the distance constraint described in Equation 2, we map the other structural constraints described in Equation 3 and Equation 5 as follows:

Anteriority constraint

$$c_1(\varphi_i, \varphi_j) = \begin{cases} \infty & \text{if } l_{v_i} = l_{v_j}, n_i < n_j \text{ and } \varphi_i > \varphi_j \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Conflict constraint with safety lapse

$$c_2(\varphi_i, \varphi_j) = \begin{cases} \infty & \text{if } \exists e \in \tau_i, e \in \tau_j \text{ and} \\ & |(\varphi_i + \frac{pos(e,\tau_i)}{s_i}) - (\varphi_j + \frac{pos(e,\tau_j)}{s_j})| \le t_{safe} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

In order to formalise our objective (Equation 1) as a DCOP, each vehicle holds a cost constraint, which directly links to its waiting time. Thus, we also have:

Waiting constraint

$$c_3(\varphi_i) = \varphi_i - (t + \frac{n_i + 1}{s_i}) \quad (8)$$

The objective of a DCOP is to minimise $\sum_{c_i(.) \in \mathcal{C}} c_i(.)$. This optimisation represents the goal of the system (minimise the global waiting time of vehicles without violating any structural constraint).

3.2 Lane-based Approach

Instead of considering each vehicle as an agent, we can consider each incoming lane as an agent. The lane agents can either be a part of the traffic control system, or be one of the vehicles in the lane that has the highest computational capability. We consider that there is an agent per incoming lane that has the knowledge of all vehicles in it. As a lane agent, it holds an array variable ϕ_l that contains the admission time of every vehicle in the lane l. By having the knowledge on all these vehicles, the lane agent can build its own domain, respecting both distance constraints and anteriority constraints. These are defined as follows:

Conflict constraint

$$c_2(\phi_i, \phi_j) = \begin{cases} \infty & \text{if } \exists \varphi_k \in \phi_i, \exists \varphi_m \in x_j, \\ & \exists e \in \tau_k, e \in \tau_m \text{ and} \\ & |(\varphi_k + \frac{pos(e,\tau_k)}{s_k}) - (\varphi_m + \\ & \frac{pos(e,\tau_m)}{s_m})| \le t_{safe} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Waiting constraint

$$c_3(\phi_i) = \sum_{\varphi_j \in \phi_i} \varphi_j - (t + \frac{n_j + 1}{s_j}) \quad (10)$$

Now that we have formalised the problem as a DCOP, it is also necessary to discuss the continuity of the solution to deal with the continuous flow of vehicles.

4 Continuity of the Solution

Since vehicles continuously approach the intersection, at each time step, we must define the vehicles that take part in the DCOP, the vehicles for which the DCOP will provide an admission time, and the conditions under which an admission time of a vehicle can be revised.

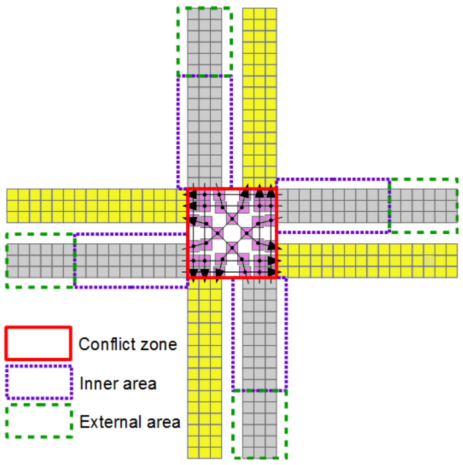

Figure 2: Inner and external areas of the intersection.

We propose several policies to manage the continuity problem. First, we distinguish two areas on the approaches of the intersection: the inner area, where all the vehicles are about to reach the conflict zone in a short term, and the external area, where the vehicles will reach the conflict zone in a slightly longer term (cf. Figure 2). The size of each area depends on the intersection. At each time step, the set V_t of the incoming vehicles is divided into two subsets: V_t^{in} the vehicles inside the inner area and V_t^{ext} the vehicles in the external area. $V_t = V_t^{in} \cup V_t^{ext}, V_t^{in} \cap V_t^{ext} = \emptyset$. Let V_t^{par} be the subset of vehicles participating in DCOP at the current time step, i.e. vehicles whose admission time can be revised. The intersection can choose to apply several policies as follows:

Iterated Policy (IP) Each vehicle in V_t^{in} participates once and only once in finding the solution. Once an admission time is chosen, it cannot be changed in the next time steps. Thus we have $V_t^{par} = V_t^{in} \backslash V_{t-1}^{in}$. This policy continues to iterate and to produce new admission times for the next vehicles in the inner area without revising those of the vehicles that already were in it.

Continuous Policy (CP) All vehicles in V_t^{in} participate in the DCOP and the admission time of every vehicle can be revised at any time step. Thus $V^{par} = V^{in}$. For safety reasons, we also note that it is risky to change the admission time of a vehicle at the last moment because of the delay in the reaction of the drivers. To avoid this, we define a safety threshold t_{low}. An admission time lower than t_{low} cannot be modified. Let V^{low} be the set of vehicles v_i having $\varphi_i - t \leq t_{low}$, we have $V^{par} = V^{in} \backslash V^{low}$.

Compared to the CP, the IP has fewer vehicles whose admission time will be assigned or modified. This leads to a lower number of agents to take part in the DCOP algorithm, reducing its computational and communication complexity. In addition, CP revises the admission time of all the vehicles, which results in a larger search space. Therefore, we expect a better quality of the solution provided using the CP (as we show later in Section 6).

5 A Max-sum Solution for the Traffic Management Problem

To solve the DCOP presented above, we use the max-sum algorithm, an *incomplete* DCOP algorithm based on the exchange of messages between agents. Despite the fact that our formalisation is compatible with any *complete* or *incomplete* DCOP algorithm, we chose to use max-sum as it is one of the fastest and most efficient algorithms in many multi-agent domains [Macarthur *et al.*, 2011; Ramchurn *et al.*, 2010; Stranders *et al.*, 2009].

In more detail, max-sum operates on a factor graph: a bipartite, undirected graph, that contains a variable node x_i for each variable, a factor node c_j for each constraint, and an edge connecting a variable node x_i with a factor node c_j if and only if x_i is involved in c_j. Each agent in max-sum takes the role of the variable node which represent its own variable. The function node's role is taken by one of the agents whose variable is involved in the constraint. Figure 3 and Figure 4 respectively show the factor graphs of the vehicle-based approach and the lane-based approach of the scenario presented in Figure 1.

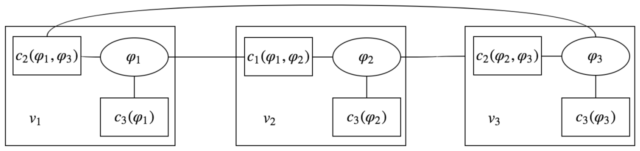

Figure 3: Vehicle-based factor graph for the scenario presented in Figure 1. There are 3 agents (v_1, v_2, v_3), each holds an admission time as a variable node $(\varphi_1, \varphi_2, \varphi_3)$, 1 anteriority factor node $c_1(\varphi_1, \varphi_2)$, 2 conflict factor nodes ($c_2(\varphi_1, \varphi_3)$ and $c_2(\varphi_2, \varphi_3)$), and 3 waiting time factor nodes ($c_3(\varphi_1)$, $c_3(\varphi_2)$, and $c_3(\varphi_3)$).

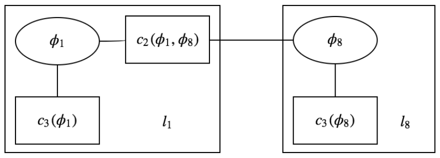

Figure 4: Lane-based factor graph for the scenario presented in Figure 1. Only two lanes have vehicles approaching the intersection so there are only two agents (l_1 and l_8). Each agent holds an array variable node, ϕ_1 contains φ_1 and φ_2, ϕ_8 contains φ_3. There are 2 waiting time factor nodes ($c_3(\phi_1)$ and $c_3(\phi_8)$), and 1 conflict factor nodes ($c_2(\phi_1, \phi_8)$).

The main routine of max-sum is the repetition of computing and exchanging messages between variable nodes and factor nodes. At each iteration i of the process, a message is sent from each variable node x to a factor node c, including for each value $d \in \mathcal{D}_x$, the sum of the costs for this value she received from all factor node neighbours apart from c in iteration $i-1$. Formally, for each value $d \in \mathcal{D}_x$ the message

$R^i_{x\to c}(d)$ includes: $\sum_{c'\in C_x\setminus c} cost(f'.d) - \alpha$, where C_x is the set of factor neighbours of variable x and $cost(c'.d)$ is the cost for value d included in the message received from c' in iteration $i-1$. α represents a scalar to prevent the message to increase endlessly in cyclic factor graphs.

To search for minimisation, the message sent from a factor node c to a variable node x contains for each possible value $d \in \mathcal{D}_x$ the minimum cost that can be achieved from any combination of other variables involved in c. Formally, for each value $d \in \mathcal{D}_x$, the message $Q^i_{f\to x}(d)$ includes $min_{PA_{-x}} cost(\langle x, d\rangle, PA_{-x})$ where PA_{-x} is a possible combination of assignments to all variables involved in c except x. The cost of an assignment $a = (\langle x, d\rangle, PA_{-x})$ is $c(a) + \sum_{x'\in X_f\setminus x} cost(x'.d')$. $c(a)$ is the original cost in the constraint c for the assignment a and $cost(x', d')$ is the cost which was received from variable node x' during iteration $i-1$, for the value d' which is assigned to x' in a.

Example 2. To give an example of the messages sent, consider the factor graph presented in Figure 3. Let $\mathcal{D}_{\varphi_1} = \{5, 6, 7\}$, $\mathcal{D}_{\varphi_2} = \{7, 8\}$. The message that the variable node φ_1 sends to the factor $c_1(\varphi_1, \varphi_2)$ at iteration i for the value d = 5 is the following: $R^i_{\varphi_1\to c_1(\varphi_1,\varphi_2)}(5) = Q^{i-1}_{c_2(\varphi_1,\varphi_3)\to\varphi_1}(5) + Q^{i-1}_{c_3(\varphi_1)\to\varphi_1}(5)$. The message sent from the factor node $c_1(\varphi_1, \varphi_2)$ to the variable node φ_1 at iteration i is the following: $Q^i_{c_1(\varphi_1,\varphi_2)\to\varphi_1}(5) = min(cost(\{5, 7\}), cost(\{5, 8\}))$, where $cost(\{5, k\}) = c_1(\{5, k\}) + R^{i-1}_{\varphi_2\to c_1(\varphi_1,\varphi_2)}(k)$.

During the propagation of messages, an agent is able to calculate locally its admission time that minimises the sum of the cost over all neighbour functions. Standard max-sum often terminates after the solution converges, or after a fixed number of iterations per agent. We have to note that the factor graph of the problem is not cycle free. Therefore, there is no guarantee of convergence with max-sum but extensive empirical evidence demonstrates that the algorithm generates good approximate solutions [Kschischang *et al.*, 2001]. In our model, the time complexity is also an issue because a solution that is found after the end of the time step is not useful. Thus, we have to optimise our the algorithm to reduce computation.

Clearly, the lane-based approach has a lower number of variables and factors compared to the vehicle-based approach. The lane-based approach considers, as an agent, a lane which contains at least 1 vehicle, thus the worst-case number of agents is the maximum number of incoming lanes ($\mathcal{O}(|L|)$), while the number of agents in vehicle-based approach grows with the number of vehicles ($\mathcal{O}(|V|)$). The number of factors is also reduced from $\mathcal{O}(\frac{|V|\times(|V|-1)}{2})$ for the vehicle-based approach to $\mathcal{O}(\frac{|L|\times(|L|-1)}{2})$ for the lane-based approach. This reduction leads to a smaller number of messages, in exchange for a growth in the average size of messages due to a larger domain. For the vehicle-based approach the domains grow at $\mathcal{O}(p)$, while for the lane-based approach the domains grow at $\mathcal{O}(p^k)$ where k is the number of vehicles presented in the lane. However, since the computational complexity of standard max-sum is exponential in the number of variables (due to combinations that factors iterate through), we expect a better performance using max-sum on the lane-based approach in dense traffic conditions.

For safety reasons, we need to ensure each vehicle is assigned an admission time before entering the intersection. However, the DCOP solver may not provide a solution in time. In the next section, we propose the role of the intersection agent which guarantees configurations.

5.1 Guaranteeing Safe Configurations

As the traffic conditions change dynamically, we have to ensure that every vehicle that enters the inner area at time step t is assigned an admission time before time step $t + 1$. To deal with this problem, we can implement an intersection agent. This agent has two roles: to hold the current configuration so that the vehicles are synchronised every time there is a change, and to assign to each vehicle that enters the intersection at the beginning of the time step a precalculated admission time. This admission time can be calculated easily by giving to each vehicle (in a random order) the earliest possible admission time, respecting all the other vehicles' admission time, including those whose admission time was just assigned. Despite not being the optimal solution for the system, this solution has two advantages: (i) it helps ensure that no vehicle in the inner area is found without an admission time at any time step, even if the DCOP solver fails to terminate in time (ii) it gives the DCOP algorithm an upper bound UB (i.e. the total waiting time of vehicles on the precalculated solution) to run a pruning algorithm as a preprocessing step.

Example 3. Consider the scenario presented in Figure 1. Let $t = 0$. At the time step $t - 1$, consider having only v_1 in the inner area. The configuration of $t - 1$ is $\{\varphi_1 = 5\}$. At the beginning of the current time step, v_2 and v_3 enter the inner area. The precalculated admission times for v_2 and v_3 are: (1) For v_2 the earliest admission time respecting $\{\varphi_1 = 5\}$ is $\varphi_2 = 7$ (2) For v_3 the earliest admission time respecting $\{\varphi_1 = 5, \varphi_2 = 7\}$ is $\varphi_3 = 11$. Therefore, we have UB = 4.

5.2 Pruning the Domains

The complexity of max-sum is known to be exponential in the number of agents where the base is the domain size and the exponent is the number of variables involved [Macarthur *et al.*, 2011]. Thus, one solution to reduce the calculation time of max-sum is to prune the search space. The pruning technique was implemented using a modified version of the preprocessing method proposed in [Stranders *et al.*, 2009] to reduce the size of the variables' domains by detecting values that are dominated. The values are detected as follows:

1. The intersection agent notifies other agents about UB.
2. The variable nodes calculate the lower bound (LB) of the cost of the value assignment, for each value assignment in their domains.
3. The variable nodes remove dominated values. A dominated value is one whose LB is higher than UB.
4. The variable nodes propagate their updated domains to the factor nodes. The factor nodes recalculate the costs and propagate them further.

5. Repeat step 2,3,4 until no more elimination found.

We note that the total cost of the solution cannot exceed UB. As mentioned in Section 4, depending on the policy, there are vehicles whose admission time cannot be changed. Let V_t^u be the set of these vehicles. Thus the cost of the admission time for each vehicle in V_t^{par} cannot exceed $UB - \sum_{v_i \in V_t^u} c_3(\varphi_i)$. Therefore, the value of p which is the range of each domain before pruning can be predetermined as $p = UB - \sum_{v_i \in V_t^u} c_3(\varphi_i)$.

Example 4. Following the scenario presented in Example 3. Let the precalculated configuration for $\{v_1, v_2, v_3\}$ be $\phi = \{5, 7, 11\}$. Thus we have $UB = 4$ and $p = 4$, initially we have $\mathcal{D}_{\varphi_1} = \{5,6,7,8,9\}, \mathcal{D}_{\varphi_2} = \{7,8,9,10,11\}, \mathcal{D}_{\varphi_3} = \{7,8,9,10,11\}$. After completing the pruning process, we obtain the following pruned domains: $\mathcal{D}_{\varphi_1} = \{5,6,7\}, \mathcal{D}_{\varphi_2} = \{7,8,9,10,11\}, \mathcal{D}_{\varphi_3} = \{7,9,11\}$.

After running max-sum on the vehicle-based scenario presented above, we obtain the following solutions: (1) With IP (the admission time of v_1 cannot be revised): $\Phi_t = \{5, 9, 9\}$, total waiting time of all vehicles: 4s. (2) With CP (any admission time can be revised): $\Phi_t = \{7, 8, 7\}$, total waiting time of all vehicles: 3s.

6 Empirical Evaluation

In this section, we evaluate the performance of our method using max-sum algorithm. All experiments were performed using an Intel Core i5-4690 3.5 GHz, 8 GB RAM, under Ubuntu 16.04. Max-sum algorithm is implemented using Frodo [Léauté *et al.*, 2009]. All compared values are averages over at least 50 simulations, with 95% confidence interval as error bars. All algorithms are evaluated according to the insertion rate of vehicles. The insertion rate varies from 0.1 (off-peak) to 0.5 (rush hour) [Junges and Bazzan, 2008]. An insertion rate of 0.5 consists of adding 5 vehicles to a lane every 10 time steps. We ran our experiments in the vehicle-based and lane-based approaches, using both IP and CP.

6.1 Benchmarking

First we compare our methods with the state of the art FCFS algorithm [Dresner and Stone, 2008] where each vehicle sends a request for an admission date and the intersection handles these requests using a First come First served policy to test the quality of the solution, the computational time and the number of messages exchanged between agents.

In terms of quality of the solution, the IP did not provide a significantly better solution compared to the FCFS policy. On the other hand, CP performs better than all the other policies, reducing the waiting time by about 60% in rush hours (cf. Figure 5a and Figure 5b). We also note that IP consumed more resources than FCFS, but less than CP. In rush hour, vehicle-based IP exchanged in average 2200 times more messages, while lane-based IP exchanged about 660 times more messages than FCFS. Lane-based CP used even more resources, exchanging 7880 times more messages than FCFS (cf. Figure 5c and 5d).

To compare the vehicle-based approach and lane-based approach, we note that both provided the same solution quality. The lane-based approach reduced the number of messages and calculation time, as its level of decentralisation is lower. In our experiments, due to the limit in computational capability of our system, we fix the time out of the DCOP solver at 6000 ms. When using the max-sum algorithm, as the time complexity grows exponentially with the number of agents, the vehicle-based approach failed very soon to provide a solution in time, while the lane-based approach using CP continued to generate solutions at the insertion rate of 0.5.

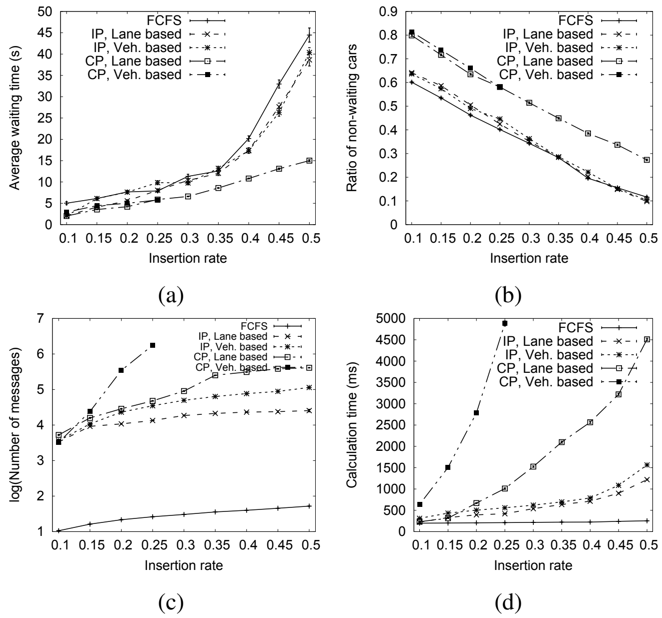

Figure 5: Empirical results. Figures (a) and (b) show the quality of the solutions where CP performed better than IP and FCFS. Figures (c) and (d) show the communication and the computational complexites. FCFS used least resources than IP and CP as expected. At the rate of 0.2, the vehicle-based CP approach failed to give a solution before the time-out.

6.2 Pruning Efficiency

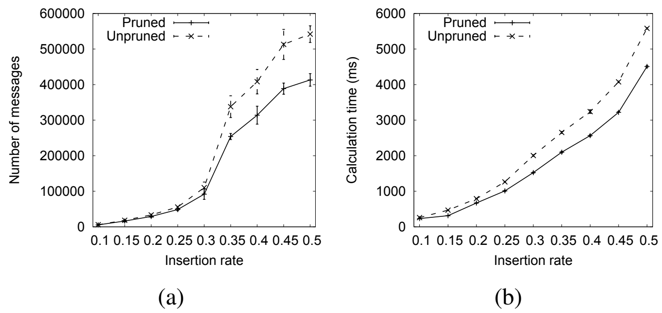

Figure 6: Performance of the pruning algorithm.

To measure the performance of the pruning algorithm, Figure 6 shows the difference in number of messages exchanged and calculation time (in milliseconds) between the pruned and unpruned versions of the lane-based approach using CP. For the unpruned version, we just fix $p = UB - \sum_{v_i \in V_t^u} c_3(\varphi_i)$ at every time step. We note that the pruned algorithm reduces about 25% - 30% of the messages exchanged and calculation time.

6.3 Dynamic Events

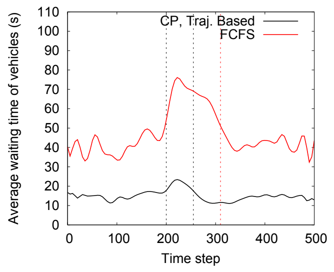

Figure 7: Average waiting time of vehicles at each time step. The emergency vehicle arrives at $t = 200$. The CP stabilises at $t \approx 250$ while FCFS stabilises at $t \approx 310$.

The DCOP formalisation of microscopic traffic regulation is also adaptable to dynamic events. We have done 20 simulations and got the average results to compare how the CP, lane-based approach and FCFS react on the arrival of emergency vehicles. We simulate the traffic over 500 time steps, with an emergency vehicle added to a random lane on time step 200. The emergency vehicle is defined in the system as a vehicle with an extremely high cost per second of waiting time. This forces the DCOP solver to look for a solution which minimises the waiting time of the emergency vehicle. This solution often leads to the immediate evacuation of the vehicles in front of the emergency vehicle in its lane. We observe that the arrival of the emergency vehicle leads to a high average waiting time on the other vehicles. FCFS succeeds to give the emergency vehicle a very low waiting time (2.7 seconds) by prioritising the emergency vehicle's lane, but this policy takes in average 110 time steps to evacuate the other lanes to return to a stable state. The DCOP approach stabilises after 50 time steps (about half the amount of time compared to FCFS) and returns to the normal condition, giving the emergency vehicle a waiting time of 2.4 seconds.

7 Conclusions

In this paper we have modelled the traffic management problem at an intersection using constraints. We then provided a DCOP formalisation of the problem and showed how we can use the Max-Sum algorithm to solve it. Our solution outperforms the state of the art solution in terms of reductions in waiting time and robustness to dynamic events.

While our work has shown the potential of DCOPs to solve traffic management problems, in future, we aim to extend the approach to consider networks and a broader set of dynamic events (e.g., closed lanes, vehicles of different lengths and sizes).

References

[Brockfeld *et al.*, 2001] Elmar Brockfeld, Robert Barlovic, Andreas Schadschneider, and Michael Schreckenberg. Optimizing traffic lights in a cellular automaton model for city traffic. *Phys. Rev. E*, 64:056132, Oct 2001.

[Carlino *et al.*, 2013] Dustin Carlino, Stephen D. Boyles, and Peter Stone. Auction-based autonomous intersection management. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 529–534, Oct 2013.

[Dresner and Stone, 2008] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *JAIR*, 31(1):591–656, March 2008.

[Farinelli *et al.*, 2008] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS '08*, pages 639–646, 2008.

[Goel and Kumar, 2015] Anju Goel and Prashant Kumar. Characterisation of nanoparticle emissions and exposure at traffic intersections through fast–response mobile and sequential measurements. *Atmospheric Environment*, 107:374 – 390, 2015.

[Junges and Bazzan, 2008] Robert Junges and Ana L. C. Bazzan. Evaluating the performance of dcop algorithms in a real world, dynamic problem. In *AAMAS '08*, pages 599–606, 2008.

[Kschischang *et al.*, 2001] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, pages 498–519, Feb 2001.

[Léauté *et al.*, 2009] Thomas Léauté, Brammert Ottens, and Radoslaw Szymanek. FRODO 2.0: An open-source framework for distributed constraint optimization. In *IJCAI'09 (DCR)*, pages 160–164, 2009.

[Macarthur *et al.*, 2011] Kathryn Macarthur, Ruben Stranders, Sarvapali Ramchurn, and Nicholas R. Jennings. A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In *AAAI '11*, pages 701–706, August 2011.

[Maerivoet and Moor, 2005] Sven Maerivoet and Bart De Moor. Cellular automata models of road traffic. *Physics Reports*, 419(1):1 – 64, 2005.

[Modi and Veloso, 2004] Pragnesh Jay Modi and Manuela Veloso. Multiagent meeting scheduling with rescheduling. In *DCR '04*, 2004.

[Ramchurn *et al.*, 2010] Sarvapali Ramchurn, Alessandro Farinelli, Kathryn Macarthur, and Nicholas R. Jennings. Decentralized coordination in robocup rescue. *Comput. J.*, pages 1447–1461, November 2010.

[Stranders *et al.*, 2009] Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nicholas R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *IJCAI'09*, pages 299–304, 2009.

[Texas A&M Transportation Institute, 2015] Texas A&M Transportation Institute. *2015 Urban mobility scorecard*, 2015.

[Vasirani and Ossowski, 2012] Matteo Vasirani and Sascha Ossowski. A market-inspired approach for intersection management in urban road traffic networks. *JAIR*, 43:621–659, 2012.